Reinhard Markus Lüftenegger

Algebraic Analysis of Arithmetization-Friendly Cryptographic Primitives

Doctoral Thesis

$$f(x) \qquad \sqrt{} \qquad \cong \qquad \notin$$

$$\mapsto \qquad \mathrm{LT}(h) \qquad \leq_{lex} \qquad R/I$$

$$\mathbb{F}_q \qquad i_{\mathrm{reg}} \qquad \delta(f) \qquad \mathcal{O}(\log N)$$

$$\mathrm{sig}(\boldsymbol{g}) \qquad \preceq_{rat} \qquad \exists \qquad x \odot y$$

$$\neq \qquad I_{\leq d} \qquad \Delta_{a_1,\ldots,a_n} \qquad \overset{*}{\to}_G$$

$$\mathrm{Syz}(F) \qquad \bigoplus \qquad \mathrm{HS}_{R/I} \qquad \forall$$

Reinhard Markus Lüftenegger

Algebraic Analysis of Arithmetization-Friendly Cryptographic Primitives

Doctoral Thesis

to achieve the university degree of
Doktor der Naturwissenschaften, Dr. rer.nat.,

submitted to
Graz University of Technology

Assessors: Christian Rechberger
Graz University of Technology

Anne Canteaut
Centre INRIA de Paris

Institute of Applied Information Processing and Communications
Graz University of Technology

Graz, October 2023

ABSTRACT

In recent years, the field of symmetric cryptography has seen a significant growth of research efforts focused on tailored cryptographic primitives that address the particular needs of modern protocols such as multi-party computation (MPC), (fully) homomorphic encryption (FHE), and zero-knowledge proofs of knowledge (ZKP). Especially zero-knowledge proof systems have recently attracted a tremendous amount of attention from, both, academia and industry. This is due to several promising applications in verifiable computation, identity protection (also known as self-sovereign identity), and user authentication.

Primitives tailored for MPC-, FHE-, or ZKP-protocols are commonly refered to as arithmetization-friendly primitives (AFPs). The particular design goals of AFPs require them to admit an efficient (i.e., concise, simple, low-degree) representation as a system of polynomial constraints. This efficient representation not only promotes arithmetization-friendliness, but, at the same time, it also makes AFPs potentially vulnerable to algebraic cryptanalysis. As a consequence, developing new techniques in algebraic cryptanalysis, and establishing a well-founded understanding of the performance thereof, have become important research directions in recent years. The present thesis discusses our research contributions in these directions. In particular, the thesis is based on the following scientific publications.

[ACG+19] We cryptanalyze the block cipher Jarvis and the hash function Friday using Gröbner basis techniques. Our analysis invalidates the security claims for all full-round instances made by the designers and indicates that a substantially higher number of rounds were necessary to restore full security.

[CGG+22] We develop novel theoretic upper bounds on the algebraic degree in substitution-permutation networks. The algebraic degree is an important indicator for the complexity of several means of cryptanalysis, such as higher-order differential distinguishers and interpolation analysis.

[GKL+22] Arithmetization-friendly hash functions aim to be efficient with respect to SNARK (e.g., R1CS or Plonk constraints) or STARK (e.g., AIR constraints) cost metrics. This has proven to be a competing design goal with plain performance. With our new design Reinforced Concrete we aim to overcome this shortcoming and propose a hash function that targets efficiency in both domains, plain performance and proving performance.

[HLL+22] Most often, equation systems modelling an AFP exhibit a particular structure. Dedicated Gröbner basis algorithms that are able to exploit this structure have great potential to improve the security analysis of AFPs and, thus, are of independent interest. We take a first step towards dedicated Gröbner basis algorithms and devise a new algorithm that performs particularly well for overdetermined, quadratic equation systems.

ACKNOWLEDGEMENT AND DEDICATION

To all who have accompanied me in times of need and joy. Your love, companionship, and mentoring are the true blessings of my life.

To Anja.
To Anna-Lena.
To Christian.
To Christof.
To Christoph.
To Clémence.
To Clemens.
To Djeneba.
To Felix.
To Ferdinand.
To Joanna.
To Hosein.
To Jasmin.
To Karl.
To Lorenzo.
To Mama, Papa, and my family.
To Maria.
To Mario.
To Markus.
To Masoud.
To Mohammad.
To Peter.
To Pramod & Priyanka.
To Pritam.
To Ralph & Rebeca.
To Renate.
To Samuel.
To Sebastian.
To Simon.
To Stefan.
To Therese.
To Werner.
To my assessor and examiner, Anne Canteaut.
To my colleagues at IAIK.
To my scientific collaborators.
To the children in Assouindé.
To you I owe my heartfelt gratitude.

AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present doctoral thesis.

CONTENTS

Part I

ALGEBRAIC PRIMITIVES IN SYMMETRIC CRYPTOGRAPHY

1

INTRODUCTION

In recent years, the field of symmetric cryptography has seen a significant growth of research efforts focused on cryptographic primitives that address the particular needs of modern protocols such as multi-party computation (MPC), (fully) homomorphic encryption (FHE), and zero-knowledge proofs of knowledge (ZKP). The literature calls these primitives[1] *domain-specific* and distinguishes them from *traditional*[2] primitives. As an example of traditional design goals, the standardization processes for the Advanced Encryption Standard (AES) and the Secure Hash Standard 3 (SHA-3) called for symmetric-key primitives that, among other requirements, are optimized for software and hardware implementations. In contrast, domain-specific primitives aim to optimize different cost metrics.

A usual design strategy for FHE-friendly primitives is to minimize the multiplicative depth of the arithmetic circuit representing them.[3] MPC-friendly primitives typically aim to minimize the number of non-linear operations. ZKP-friendly primitives aim for a concise (polynomial) description in terms of operations over their native domain.[4] In a faithful approximation, the common trait that all of these primitives share is the "simple" algebraic structure of their building blocks. In case of (keyed or unkeyed) cryptographic permutations, this translates to algebraically "simple" round functions. The term "simple", for example, may refer to a round function based on low-degree polynomial functions such as the cubing function $x \mapsto x^3$. Another example is the inverse fuction $x \mapsto x^{-1}$, that maps every field element to its (multiplicative) inverse, and that can be expressed via the low-degree polynomial relation[5] $x^2 \cdot y = x$. The simple algebraic structure of domain-specific primitives is also the reason why they may be susceptible to algebraic cryptanalysis. As a consequence, the increasing popularity of algebraic designs in symmetric cryptography has led to a renewed interest in algebraic cryptanalysis. We will touch upon this topic again in Section 1.2.

The focus of this thesis lies on ZKP-friendly primitives. For reasons we shall motivate in Section 1.1, the literature calls these primitives also *arithmetization-friendly*[6] primitives (AFPs). In this document, we adopt the latter denomination.

1.1 Zero-Knowledge Proofs of Knowledge

In, both, academia and industry, zero-knowledge proofs (of knowledge) have recently attracted a tremendous amount of attention. Research on practically efficient proof systems is a rapidly evolving research area with

[1] Together with primitives specialized for other domains, such as *lightweight cryptography*.
[2] Here, "traditional" refers to a timespan from standardizing the Data Encryption Standard (DES) and onwards.
[3] An important exception are Torus-based FHE protocols [CGG+20], where the depth is not the foremost concern.
[4] There are noteworthy exceptions. Recent proposals for ZKP-friendly permutations [GKL+22; GKL+23; SLS+23] utilize complex, high-degree building blocks that do *not* admit a concise polynomial description over the native domain. These building blocks are implemented using lookup tables.
[5] If $x \neq 0$, this simplifies to $x \cdot y = 1$.
[6] Sometimes also *arithmetization-oriented* or *circuit-friendly*.

several promising applications such as verifiable computation[7], identity protection (also known as "self-sovereign identity"), and user authentication. Informally speaking and omitting technical details, the rationale behind a *zero-knowledge proof* is as follows.

> "A prover convinces a verifier that a given statement is valid, without revealing any information beyond the validity of the statement itself."

The prover is the party intending to prove a claim, while the verifier is responsible for validating the claim. The following discussion elaborates on this informal rationale. Our discussion will be brief, and intentionally imprecise from a mathematical point of view. Furthermore, we use the terms "statement", "claim", "assertion" interchangeably. Part of our discussion is guided by the exposition of the same topic in the textbook [Gol01].

What is a statement? In the context of our motivational discussion, a computational statement may be regarded as an instance of a computational problem in a certain complexity class, usually NP. For representational purposes, it is advantageous to choose complete problems [Gol08, Section 2.3.5]. Examples of such problems relevant for proof systems are

- the NP-complete problem describing satisfiability of *arithmetic circuits* [BCG+14a; BCG+14b; BCI+13; Gro16],
- the NP-complete problem describing constraint satisfaction of *rank-1 constraint systems (RC1S)* [BCR+19],
- the NP-complete problem describing constraint satisfaction of *quadratic arithmetic programs (QAP)* [GGP+13], or
- the NEXP-complete problem describing constraint satisfaction of *permuted algebraic intermediate representations (PAIR)* [BBH+18b].

What is a proof? The notion of "proof" has a rather specific (i.e., technical) meaning in the context of (zero-knowledge) proof systems. To motivate this notion, let us briefly intuit how mathematical proofs establish the validity of a statement. A mathematical proof may be viewed as a *deductive argument*, i.e., its nature is a *logical* one. It consists of a sequence of propositions, beginning with the premises and ending with the claimed statement. Each proposition in this sequence is either self-evident (i.e., an axiom, a commonly agreed postulate that doesn't require proof; it is taken to be true) or derived from a previous statement via established inference rules (that, again, have a self-evident character). In this sense, a mathematical proof is *static* (i.e., monologic). Besides sharing a (written account of the) proof, there is no further element of interaction between a prover and a verifier.

In contrast, the notion of "proof" as in "(zero-knowledge) proof system" carries the connotation of a discourse (in the sense of conversation, argument, questioning, examination or interrogation) and may be viewed as a *discursive argument*. A proof in this sense consists of exchanged messages between two parties, the prover and a verifier. The prover provides answers to (random and challenging) questions from a verifier in order to prove the validity of an assertion. If the prover's answers are convincing, the verifier

[7] Also refered to as computational integrity in [BBH+18b].

is persuaded and deems the assertion to be true. This persuasive[8] notion implicitly contains a likelihood for error. This means, its nature is, in principal, an *interactive* (i.e., dialogic, rhetorical and persuasive) one. For a more comprehensive discussion, we refer to the textbook [Golo1, Section 4.1].

Adopting a more technical speech, and in a reliable simplification, the term "proof" as in "proof system" refers to a concept known as *succinct non-interactive argument of knowledge*. Here, *succinct* means, the proof is (much) smaller than the assertion itself (and can be verified quickly). An *interactive argument* is an interactive protocol between a computationally bounded (deterministic) prover and a computationally bounded probabilistic verifier that satisfies the following properties.

1. *(Perfect) Completeness.* True assertions are always accepted by the verifier.
2. *(Computational) Soundness.* Wrong assertions are rejected by the verifier with overwhelming probability (as long as the computational power of the potentially cheating prover is bounded).

Completeness expresses the prover's ability to convince a verifier of true assertions. *Perfect* completeness means, there is always a way to convince the verifier of true assertions, without error. *Soundness* captures the verifier's ability to recognize wrong assertions (with negligible error probability). *Computational* soundness says that a verifier's verdict is sound (with negligible error probability) as long as the computational power of the potentially cheating prover is bounded.[9] An *interactive argument of knowledge* is an *interactive argument* with the additional property of *knowledge-soundness*. We do not motivate this property, but refer to the comprehensive manuscript [Tha22]. A *public-coin* interactive argument, is an interactive argument where the random choices of the verifier are publicly known. Any *public-coin interactive argument* may be compiled into a *non-interactive argument* in the random oracle model using the *Fiat-Shamir transform* [FS86; CCH+19].

Two important classes of "proofs" are *SNARKs* [PHG+13; Gro16; GWC19; BCR+19; COS20][10] and *STARKs* [BBH+18b].

What is Zero-Knowledge? The notion of *zero-knowledge* pertains to the idea that the proof of a statement should yield no information (i.e., no knowledge) beyond the validity of the statement itself.[11] Within the framework of proof systems, the term "knowledge" relates to computational difficulty. We consider a *gain of knowledge* as an increase in our computational abilities. Meaning, if after interacting with a prover we can efficiently solve a computational problem that we couldn't solve before, we perceive this as a gain

[8] The term "persuasive" is meant as a contrast to "logical". It alludes to the notion of "persuasion" as in "proof by persuasion". Here, the prover (e.g., a proponent, or defendant) interactively delivers convincing points and evidence in favor of an assertion to the verifier (e.g., an opponent, or plaintiff).

[9] This sentence deserves a comment. Technically speaking, there is a dedicated meaning of the term "proof" as in "proof system". If the prover is given unbounded computational power and above soundness requirement is assserted for such unbounded provers, the literature speaks of a *proof system*. This is why argument systems are also called *computationally sound proof systems*. Especially in practical contexts, the assumption of computationally bounded provers is often made implicitly. This leads to the habit of (tacitly) using the terms "argument" and "proof" synonymously. However, formally, these are different notions.

[10] The references cited are not intended to be exhaustive or representative of the entire domain of SNARKs.

[11] The term "information" should be understood in a colloquial manner. In the context of information theory, "information" has a rather concrete meaning. This meaning is different from what we want to emphasize here.

of knowledge.[12] In contrast, if whatever we can efficiently compute after interacting with a prover is what we could efficiently compute beforehand (i.e., by ourselves, without any interaction), we acknowledge that we have not gained any knowledge. With a slight shift of perspective, we may say (the interaction with) the prover revealed *zero knowledge.*

This line of thought leads to the *simulation paradigm*. We do not elaborate further on this topic. For more information we refer to the excellent textbook [Golo1].

Arithmetization Modern argument systems transform computational statements into an algebraic form, e.g., Groth'16 [Gro16] or Pinocchio [PHG+13] uses R1CS, while STARKs [BBH+18b] use (P)AIRs. We briefly motivate this transformation, commonly known as *arithmetization*. Arithmetization refers to the reduction (or transformation, translation, representation, compilation) of a computational statement (such as "I know the preimage to a given hash value.") into a *structured algebraic statement*. The term *algebraic* conveys the conventional meaning, i.e., *polynomial*. In the context of our motivational exposition, it is accurate enough to think of an *algebraic statement* as a system of polynomial equations (or constraints) over a finite field. The term *structured* alludes to the fact, that the final equation system exhibits a specific structure that follows from a predefined format. In the following considerations, we discuss three popular arithmetization formats. If not stated otherwise, let $\mathbb{F}$ denote a finite field.

Rank-1 Constraint System (R1CS) A common arithmetization format in modern SNARKs[13] are rank-1 constraint systems (R1CS). As an intermediate step in the arithmetization process, a computation (i.e., an arithmetic circuit) is translated into a R1CS. A R1CS over a field $\mathbb{F}$ is an equation system with m constraints (i.e., equations) over $\mathbb{F}$ in n variables $x_1, \ldots, x_n$, where the constraints have the form

$$\forall i \in \{1, \ldots, m\} : \left(\sum_{j=1}^{n} A_{i,j} x_j\right) \cdot \left(\sum_{j=1}^{n} B_{i,j} x_j\right) = \sum_{j=1}^{n} C_{i,j} x_j.$$

Here, all $A_{i,j}$, $B_{i,j}$, and $C_{i,j}$ are coefficients in $\mathbb{F}$. If

$$\gamma = \begin{pmatrix} \gamma_{1,1} & \gamma_{1,2} & \cdots & \gamma_{1,n} \\ \gamma_{2,1} & \gamma_{2,2} & \cdots & \gamma_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_{m,1} & \gamma_{m,2} & \cdots & \gamma_{m,n} \end{pmatrix}, \quad \text{for } \gamma \in \{A, B, C\},$$

denotes the corresponding matrix of coefficients, then the short notation for a R1CS is

$$(A \cdot x) \odot (B \cdot x) = C \cdot x.$$

Here, the operator $\cdot$ denotes ordinary matrix-vector-multiplication, $\odot$ denotes the Hadamard-product (i.e., element-wise multiplication), and $x = (x_1, \ldots, x_n)^t$ is the vector of variables.

Plonk Constraint System Plonk [GWC19] uses *Plonk gates*[14] over $\mathbb{F}$, which are constraints (i.e., equations) over $\mathbb{F}$ in the variables x, y, z of the form

$$q_L \cdot x + q_R \cdot y + q_O \cdot z + q_M \cdot xy + q_C = 0.$$

[12] E.g., after interacting we know the prime factorization of a large integer.
[13] E.g., in Groth'16 [Gro16], or Pinocchio [PHG+13]
[14] Also called *Plonk (gate) constraints*.

The constants $q_L, q_R, q_O, q_M, q_C \in \mathbb{F}$ are also called *selectors*. Let

$$Q_\alpha = \begin{pmatrix} q_{\alpha_1} \\ \vdots \\ q_{\alpha_m} \end{pmatrix}, \quad \text{for } \alpha \in \{L, R, O, M, C\},$$

denote the corresponding vector of selectors in $\mathbb{F}^m$. Then, a system of m Plonk gates in the $3m$ variables $x_1, \ldots, x_m$, $y_1, \ldots, y_m$, $z_1, \ldots, z_m$ can be expressed as

$$Q_L \odot x + Q_R \odot y + Q_O \odot z + Q_M \odot (x \odot y) + Q_C = 0.$$

Here, $\odot$ denotes the Hadamard-product (i.e., element-wise multiplication) and, for $\beta \in \{x, y, z\}$, the vector $\beta = (\beta_1, \ldots, \beta_m)^t$ denotes the accompanying vector of variables. For relating variables to each other, Plonk uses *copy constraints* (also known as *equality constraints*). Let

$$h := (h_i)_{i \in \{1,\ldots,3m\}} := (x_1, \ldots, x_m, y_1, \ldots, y_m, z_1, \ldots, z_m)$$

denote the $3m$-tuple containing all $3m$ variables. Furthermore, let σ be a permutation of $\{1, \ldots, 3m\}$ such that $\sigma(i) = j$ captures the relation $h_i = h_j$, i.e.,

$$\sigma(i) = j \iff h_i = h_j.$$

In other words, if some variables are equal (i.e., copied), then σ permutes the indices of these variables. The indices of uncopied variables are left untouched by σ. Plonk expresses equality constraints as

$$\prod_{i=1}^{3m} (h_i + ir + s) = \prod_{i=1}^{3m} (h_i + \sigma(i)r + s),$$

for uniformly at random chosen $r, s \in \mathbb{F}$, see [GWC19, Claim A.1].

Algebraic Intermediate Representation (AIR) As part of the arithmetization process, STARKs represents computations as a sequence of computation states connected via transition functions. This representation is called *algebraic intermediate representation (AIR)*.[15] Basically, an AIR consists of

- an execution trace T of a certain size $m \times w$, for $m, w \in \mathbb{N}$,
- polynomial transition constraints in $2w$ variables, and
- boundary constraints.

At any time step $i \in \{1, \ldots, m\}$, the state of a computation is determined by the content of w registers taking values from $\mathbb{F}$. Thus, the *execution trace*[16] T is an $m \times w$ matrix over $\mathbb{F}$, where rows are indexed by time steps and columns are index by registers. In other words, the i-th row represents the state of a computation at time step i and the j-th column tracks the content of register j over time. Polynomial *transition constraints* specify the transition relations between any two consecutive states. These constraints are satisfied if and only if state i is correctly related to state $i + 1$. Stated differently, transition constraints relate to consecutive rows in T with each other. Each constraint, thus, is a multivariate polynomial equation over $\mathbb{F}$ in $2w$ variables. In total, all transition constraints are satisfied if and only if

[15] It resembles register machines [Rob20].
[16] Sometimes also called *algebraic execution trace* (AET).

the whole execution trace T is valid. For efficiency reasons, it is desirable to keep the degree of the transition constraints low. *Boundary constraints* simply enforce the values of some (or all) registers in the initial state, the final state, or in an arbitrary state. Boundary constraints, thus, can be represented by tuples of field elements in $\mathbb{F}$.

1.2 Arithmetization-Friendly Primitives and their Cryptanalysis

The main characteristics that distinguish arithmetization-friendly primitives from traditional ones are

1. their native domain, and
2. their concise description in terms of polynomial constraints.

To comment on 1., we remark that AFPs aim to be efficient not only over binary extension fields $\mathbb{F}_2^n$ but also over large prime fields $\mathbb{F}_p$ of odd characteristic (or extensions thereof). Working with large prime fields is motivated by the fact that many modern SNARKS rely on elliptic-curve based primitives, such as elliptic curve pairings [EMJ17], that directly work over these prime fields.[17] For 2., it is insightful to keep in mind the different arithmetization techniques outlined in Section 1.1. Generally speaking, for all of these arithmetization techniques, the performance of AFPs inside proof systems benefits from a concise and low-degree polynomial representation of the primitive.[18]

In the following discussion, we present a brief overview of the state-of-the-art in designing and analyzing arithmetization-friendly cryptographic primitives. Arguably, the first arithmetization-friendly primitive is MiMC, a block cipher and a corresponding sponge-based hash function published in 2016 [AGR+16].[19] As an attempt to address the performance bottleneck of traditional ciphers in SNARK applications, the designers of MiMC aimed to minimize the total number of multiplications[20] of an arithmetic circuit representing the permutation. Since its inception in 2016, MiMC has attracted a considerable amount of cryptanalysis from the research community [Bon19; ACG+19; LP19; EGL+20; BGL20; CHW+22; BBL+22; BCP23].

The MPC-oriented block cipher LowMC, published in 2015 [ARS+15a], preceeded MiMC and pursued similar design goals. However, the design of LowMC explored an (extreme) trade-off between non-linear and linear operations, trying to shift a large part of the cryptographically relevant work to linear operations. This approach is motivated by another cost metric that considers the cost of linear operations negligible compared to the cost of non-linear ones. Considering linear operations, essentially, as "free" does not necessarily translate to arithmetization-friendliness in the sense as outlined in Section 1.1. Hence, we do not include LowMC in our listing of AFPs.

[17] E.g., Plonk [GWC19] builds upon the the pairing-based KZG polynomial commitment scheme [KZG10].
[18] Here, the same remark as in Footnote 4 applies.
[19] A possible objection to counting MiMC as the first AFP could be the elliptic curve based Pedersen hash function [BHH+22, Section 5.4.1.7]. Pedersen hash is a collision-resistant hash function based on the hardness of the Discrete Logarithm Problem on the JubJub curve [BHH+22, Section 5.4.9.3].
[20] This cost metric is also known as the *multiplicative complexity*.

The block cipher Jarvis and the corresponding hash function Friday[21], published in 2018 [AD18], aimed to be specifically efficient in STARK [BBH+18b] applications. Standard primitives, such as AES, SHA-2 and SHA-3, have turned out to be inefficient in these applications. This created the need for tailored cryptographic primitives. Jarvis and Friday aimed to address this need. Our independent cryptanalysis [ACG+19], however, invalidated the security claims of the designers, leading to the first successful Gröbner basis analysis of a block cipher. As a consequence, the authors of [AD18] abandoned their initial design strategy of Jarvis and Friday, and devised the follow-up designs Vision (defined over $\mathbb{F}_{2^n}^m$) and Rescue (defined over $\mathbb{F}_p^m$, for p prime and m a natural number) [AAB+20]. Both, Vision and Rescue, instantiate sponge-based hash functions with the same names. Related cryptanalysis can be found in [BCL+20; BGL20; BCD+20b]. A version of Rescue that is optimized for the prime field $\mathbb{F}_p$ with $p = 2^{64} - 2^{32} + 1$ is suggested in [AKM+22; SAD20].

The year 2019 saw the publication of two other AFPs, GMiMC [AGP+19] and Poseidon [GKK+19; GKR+21]. The GMiMC permutation is a generalized Feistel-variant of MiMC, constructed from a range of (balanced and unbalanced) Feistel networks. The Poseidon permutation is based on the Hades design strategy[GLR+20], a substitution-permutation network (SPN) that uses, both, full S-box layers and partial ones. In other words, Poseidon uses an uneven distribution of S-boxes. The S-box functions in Poseidon are low-degree polynomial power maps.[22] For that reason, Poseidon can be regarded as an SPN-variant of MiMC. Both permutations, GMiMC and Poseidon, instantiate a hash function (with the same name) using the sponge-construction. For cryptanalysis of GMiMC, we refer to [Bon19; RAS21; BCD+20b; CHW+22]. Due to its efficiency in the target use cases and the fast adoption in industry [GKR+21, Section 3], Poseidon has received much cryptanalytic attention [BCD+20b; BCD+20a; KR21; BBL+22; ABM23]. For completeness, we mention two other designs related to Poseidon. Neptune[GOP+22] is a variant of Poseidon that explores the use of generalized Lai-Massey functions as S-box functions. Poseidon2 [GKS23] is an updated variant of Poseidon that takes into account the findings of previous cryptanalysis.

In 2021, the permutation and sponge-based hash function Grendel [Sze21] was proposed in attempt to leverage the Legendre symbol as a building block for the non-linear layer in cryptographic permutations. Soon after the publication of Grendel, the authors of [GKR+22] found a preimage attack on the Grendel hash function, and, hence, invalidated the security claims made in [Sze21].

The year of 2022 marked the publication of Reinforced Concrete [GKL+22], a permutation and sponge-based hash function that leverage lookup tables for fast native performance. At the same time, Reinforced Concrete targets efficiency in proof systems with support for lookup arguments.[23] Achieving efficiency in both domains, native performance and proof system performance, is important in applications where the same hash function is used to build a Merkle tree and to prove an opening of the Merkle tree.[24]

[21] The block cipher Jarvis is turned into a compression function using the Miyaguchi-Preneel mode of operation. The resulting compression function is then employed in a Merkle-Damgård-construction to produce the hash function Friday.

[22] Of the form $x \mapsto x^\alpha$, for $\alpha \in \mathbb{N}$, $\alpha \geq 3$.

[23] E.g., Plonkup [PFM+22], a version of Plonk [GWC19] that integrates the ideas of Plookup [GW20].

[24] As is, e.g., the case in the Fractal recursive protocol [COS20].

In addition to Reinforced Concrete, two other AFPs were published in 2022: Anemoi [BBC+23] and Griffin [GHR+23]. The designers of Anemoi propose a novel S-box construction called Flystel that exploits the CCZ-equivalence of a low-degree function with a high-degree one. In essence, the CCZ-equivalence of these functions allows, both, a high degree evaluation and a low degree verification of the S-box function. Using CCZ-equivalence as a means to achieve arithmetization-friendliness generalizes the approach taken in [AAB+20]. The authors of [AAB+20] make the following observation for a permutation F of $\mathbb{F}_q^m$, $m \in \mathbb{N}$, $m \geq 1$. If F has a high degree, whereas F^{-1} possesses a low degree, then verifying the high-degree statement $y = F(x)$ is equivalent to verifying the low-degree statement $F^{-1}(y) = x$, for any given $x, y \in \mathbb{F}_q^m$. In other words, a permutation may use high-degree round functions for security and, at the same time, admit a low-degree representation for efficient verification. The authors of [BBC+23] present a more general instantiation of this approach. Let $F, G : \mathbb{F}_q^m \longrightarrow \mathbb{F}_q^m$ be two CCZ-equivalent functions. Then, there exists an affine permutation[25]

$$\begin{aligned} \mathcal{L} : \left(\mathbb{F}_q^m\right)^2 &\to \left(\mathbb{F}_q^m\right)^2 \\ (x, y) &\mapsto (\mathcal{L}_L(x, y), \mathcal{L}_R(x, y)) \end{aligned}$$

such that

$$\mathcal{L}(\Gamma_F) = \{\mathcal{L}(x, F(x)) : x \in \mathbb{F}_q\} = \Gamma_G.$$

Here, $\Gamma_f = \{(x, f(x)) : x \in \mathbb{F}_q\}$ denotes the graph of $f \in \{F, G\}$. Due to the CCZ-equivalence of F and G, it holds for $x, y \in \mathbb{F}_q$

$$\begin{aligned} y = F(x) &\iff (x, y) \in \Gamma_F \iff \mathcal{L}(x, y) \in \mathcal{L}(\Gamma_F) = \Gamma_G \\ &\iff \mathcal{L}_R(x, y) = G(\mathcal{L}_L(x, y)). \end{aligned}$$

In other words, if F is of high degree and G is of low-degree, the relation $y = F(x)$ can be verified by the low-degree relation $\mathcal{L}_R(x, y) = G(\mathcal{L}_L(x, y))$. We remark, verifying a statement via "inverse equivalence", i.e., via the equivalence

$$y = F(x) \iff F^{-1}(y) = x$$

is a special case of verifying a statement via CCZ-equivalence

$$y = F(x) \iff \mathcal{L}_R(x, y) = G(\mathcal{L}_L(x, y))$$

of two functions F, G. Indeed, any permutation F is CCZ-equivalent to its inverse F^{-1} via the mapping $\mathcal{L}(x, y) \coloneqq (y, x)$. Hence, compared to "inverse equivalence", CCZ-equivalence offers a more general possibility to verify a high-degree evaluation via an equivalent low-degree relation. Regarding the targeted use cases, the Anemoi permutation instantiates compression and sponge-based hash functions that aim to be efficient in, both, SNARK and STARK applications.

The Griffin permutation employs a novel construction for the non-linear part in their round function. This novel construction combines a modified Feistel network called Horst with, both, low-degree and high-degree power maps into a single non-linear function. The authors of [GHR+23] propose two modes of operation for the Griffin permutation: a compression function with feed-forward and subsequent truncation, and the sponge mode of operation. Griffin targets efficiency in, both, SNARK and STARK applications.

[25] In its most general form, an affine permutation $\mathcal{L}$ of $(\mathbb{F}_q^m)^2$ has the form $\mathcal{L}(x, y) = M \cdot (x, y)^t + (a, b)^t$, for $a, b \in \mathbb{F}_q^m$ and M a non-singular $2m \times 2m$ matrix over $\mathbb{F}_q$.

Recent addings to the family of AFPs are: Monolith [GKL+23], Arion and ArionHash [RST23], Tip5 [SLS+23] (see [Sal23] for two additional instantiations called Tip4 and Tip4'), and XHash8, XHash12 [AKM23].

Let There Be Order To shed light on the current "zoo" of AFPs, we discuss and identify major design trends, and conclude with a first classification of AFPs. We advise the reader to take this (and any) classification with a grain of salt, as the whole area of arithmetization-friendly hashing is a *very* dynamic field of research at the moment. This means, design trends evolve quickly and it might be too early for a reliable classification without being too assertive when devising suitable categories.

The following classification suggests three different categories (similar, but different, to the classifications found in [RST23, Section 1], [SLS+23, Section 1]), [AD20, Section 1]. We only mention hash functions, without the underlying permutations.

- *Generation I*. MiMC, GMiMC, Poseidon, Poseidon2, Neptune
- *Generation II*. Friday, Vision, Rescue, Rescue-Prime, XHash8/12, ArionHash, Griffin, Grendel, Anemoi
- *Generation III*. ReinforcedConcrete, Monolith, Tip5, Tip4/4'

In the first generation, we consider designs that only use low-degree round functions in their permutation. This includes permutations based on, both, Feistel-networks and SPNs, and an uneven distribution of S-Boxes as in Poseidon. Designs in this category usually show a fast plain performance due to the low-degree of their round function. The downside is that permutations with low-degree round functions require a high number of rounds to prevent algebraic cryptanalysis.

The second generation consists of designs that use low-degree *and* high-degree components in their round functions. The specific trait of this category is the insight that verification of a computation needn't be done directly, but can also be achieved indirectly (or non-procedurally, as the authors of [AAB+20, Section 3.1] call it). An example is the verification of the high-degree inverse function $x \mapsto x^{q-2}$ on $\mathbb{F}_q$. Although the statement $y = x^{q-2}$ is of high degree, it can be verified by the low-degree statement[26] $x^2 \cdot y = x$. Due to the high-degree building blocks in the round functions, permutations in this category tend to be slow when evaluated plainly. However, in general, much fewer rounds are needed to resist algebraic cryptanalysis compared to Generation-I designs.

With the third generation we capture designs that leverage lookup tables for fast plain evaluation and target efficiency in, both, plain evaluation and zero-knowledge proving performance. Designs in this category use lookup tables to implement complex, high-degree round functions in an efficient manner. As a result, utilizing lookup tables for implementing complex round functions tends to further reduce the number of rounds to resist (algebraic) cryptanalysis compared to Generation-II designs. We remark, Generation-III designs primarily target proof systems that support lookup arguments in their arithmetization step.

Concluding Remarks The particular design goals of AFPs require them to admit an efficient (i.e., simple, low-degree) representation as a system

[26] It holds $x^q = x$, for all $x \in \mathbb{F}_q$.

of polynomial constraints. This efficient representation not only benefits the arithmetization in proof systems, but, at the same time, also makes AFPs potentially vulnerable to algebraic cryptanalysis. For example, MiMC exhibits a particularly slow growth of the algebraic degree, as observed in [BCP23] and exploited in [EGL+20]. Another example is our cryptanalysis of Jarvis and Friday in [ACG+19]. More recently, the authors of [ABM23] claim to have found some caveats in the security analysis of non-standard instances of Poseidon. The emergence of AFPs, thus, has led to a renewed interest in algebraic cryptanalysis. As a consequence, developing new techniques in algebraic cryptanalysis and establishing a well-founded understanding of the performance thereof have become important research directions in the last years. We present our contributions in this direction in the next section.[27]

1.3 Our Contributions

Below, we give an overview of the scientific articles forming the foundation of this thesis. Besides a brief motivation for each article, we provide a summary of the author's original contribution to the joint work.

I Martin R. Albrecht, Carlos Cid, Lorenzo Grassi, Dmitry Khovratovich, Reinhard Lüftenegger, Christian Rechberger, and Markus Schofnegger. "Algebraic Cryptanalysis of STARK-Friendly Designs: Application to MARVELLous and MiMC." in: *Advances in Cryptology - ASIACRYPT 2019*. Ed. by Steven D. Galbraith and Shiho Moriai. Vol. 11923. Lecture Notes in Computer Science. Springer, 2019, pp. 371–397. DOI: 10.1007/978-3-030-34618-8_13

In [ACG+19], we cryptanalyze the block cipher Jarvis and hash function Friday using Gröbner basis techniques. Jarvis and Friday aimed to be efficient in STARK applications and, hence, were constructed from building blocks that admit a low-degree and simple polynomial representation. We show that the particular combination of the building blocks in Jarvis leads to a successful Gröbner basis analysis of, both, Jarvis and Friday. Our analysis invalidates the security claims for all full-round instances made by the designers and indicates that a substantially higher number of rounds were necessary to restore full security. The author devised the algebraic models and implemented Gröbner basis experiments on toy versions of Jarvis and Friday. Furthermore, the author was involved in the development of the ideas underpinning the algebraic cryptanalysis and wrote the majority of the article [ACG+19].

II Carlos Cid, Lorenzo Grassi, Aldo Gunsing, Reinhard Lüftenegger, Christian Rechberger, and Markus Schofnegger. "Influence of the Linear Layer on the Algebraic Degree in SP-Networks." In: *IACR Transactions on Symmetric Cryptology* 2022.1 (2022), pp. 110–137. DOI: 10.46586/tosc.v2022.i1.110-137

Most cryptographic permutations iterate a simple round function many times to construct a cryptograpically secure permutation. For designs defined over $\mathbb{F}_2^n$, the degree of the n-variate polynomial representation over $\mathbb{F}_2$ is an important indicator for the complexity of several means of cryptanalysis, such as higher-order differential distinguishers and interpolation analysis. The literature calls this degree also the algebraic degree.

[27] All contributions list author names in alphabetical order; see https://www.ams.org/profession/leaders/CultureStatement04.pdf.

Establishing explicit and exact formulae for the algebraic degree as the number of rounds increases is, in general, a difficult task. Hence, obtaining estimates and working with (tight) upper bounds on the algebraic degree is an important research direction. In [CGG+22], we develop novel theoretic upper bounds on the algebraic degree in SPNs. As a key contribution, we show how the (word-level) degree of the affine part of the round function influences the growth of the algebraic degree. For low-degree round functions, our new bound provides considerably tighter estimates than the best known theoretic bounds in [BCC11]. The author contributed the main idea behind the bound on the algebraic degree and developed most of the underlying proofs. Furthermore, the author supervised the practical tests and wrote the majority of the article [CGG+22].

III Lorenzo Grassi, Dmitry Khovratovich, Reinhard Lüftenegger, Christian Rechberger, Markus Schofnegger, and Roman Walch. "Reinforced Concrete: A Fast Hash Function for Verifiable Computation." In: *SIGSAC Computer and Communications Security - CCS 2022*. Ed. by Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi. ACM, 2022, pp. 1323–1335. DOI: 10.1145/3548606.3560686
Arithmetization-friendly hash functions aim to be efficient with respect to SNARK (e.g., R1CS or Plonk constraints) or STARK (e.g., AIR constraints) cost metrics. This has proven to be a competing design goal with plain performance. Arithmetization-friendly hash functions tend to be slow when evaluated plainly,[28] and, vice versa, more "traditional" hash functions[29] with fast plain performance have revealed themselves to show poor performance inside zero-knowledge proof systems. With Reinforced Concrete we aim to overcome this shortcoming and propose a hash function that targets efficiency in both domains, plain performance and proving performance. Most notably, Reinforced Concrete leverages lookup tables for an efficient implementation of algebraically complex transformations of large prime field elements. This approach primarily targets proof systems which offer lookup gates in their underlying arithmetization (e.g., Plonkup). The author was responsible for the algebraic analysis of Reinforced Concrete. This included experiments on toy instances of Reinforced Concrete using interpolation and Gröbner basis techniques. Furthermore, the author made significant contributions to proving the completeness and soundness of the table constraints, and he devised the proof for the bijectivity of the Bar function.

IV Manuel Hauke, Lukas Lamster, Reinhard Lüftenegger, and Christian Rechberger. "A Signature-Based Gröbner Basis Algorithm with Tail-Reduced Reductors (M5GB)." in: *IACR Cryptology ePrint Archive* (2022), p. 987. URL: https://eprint.iacr.org/2022/987
Gröbner basis cryptanalysis has received much attention in the last years from the research community in symmetric cryptography. This is motivated by the fact that Gröbner bases play a key role in the security analysis of "algebraic" cryptographic primitives that aim to be efficient in MPC-, FHE-, or ZKP-applications (see Section 1.2 for an overview about ZKP-friendly designs). For these "algebraic" designs, Gröbner basis techniques are an important means to assess and argue for their security. Most often, equation systems modelling a given cryptographic primitive do not

[28] We refer to our classification at the end of Section 1.2 for a more nuanced discussion of this point.
[29] E.g., SHA2/SHA3 or Blake2b/s.

behave like random systems and exhibit a particular structure (e.g., a multi-homogeneous structure [MS87; LLB03; MM07]). Dedicated Gröbner basis algorithms that are able to exploit this structure have great potential to improve the security analysis of "algebraic" primitives and, thus, are of independent interest. In [HLL+22], we take a first step towards dedicated Gröbner basis algorithms. We devise a new Gröbner basis algorithm that performs particularly well for overdetermined, quadratic equation systems. Our new algorithm is called `M5GB` and merges ideas from state-of-the-art signature-based Gröbner basis algorithms (such as `F5` [Fau02]) with the efficient reduction routine of `M4GB` [MS17]. The author developed the idea behind `M5GB` together with Manuel Hauke and made significant contributions to the algorithmical formulation of `M5GB`. Furthermore, the author supervised the implementation of `M5GB`, aided in proving termination and correctness and wrote most of the article [HLL+22].

2

LIST OF PUBLICATIONS

[ACG+19] Martin R. Albrecht, Carlos Cid, Lorenzo Grassi, Dmitry Khovratovich, Reinhard Lüftenegger, Christian Rechberger, and Markus Schofnegger. "Algebraic Cryptanalysis of STARK-Friendly Designs: Application to MARVELlous and MiMC." in: *Advances in Cryptology - ASIACRYPT 2019*. Ed. by Steven D. Galbraith and Shiho Moriai. Vol. 11923. Lecture Notes in Computer Science. Springer, 2019, pp. 371–397. DOI: 10.1007/978-3-030-34618-8_13

[GLR+20] Lorenzo Grassi, Reinhard Lüftenegger, Christian Rechberger, Dragos Rotaru, and Markus Schofnegger. "On a Generalization of Substitution Permutation Networks: The HADES Design Strategy." In: *Advances in Cryptology - EUROCRYPT 2020*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12106. Lecture Notes in Computer Science. Springer, 2020, pp. 674–704. DOI: 10.1007/978-3-030-45724-2_23

[EGL+20] Maria Eichlseder, Lorenzo Grassi, Reinhard Lüftenegger, Morten Øygarden, Christian Rechberger, Markus Schofnegger, and Qingju Wang. "An Algebraic Attack on Ciphers with Low-Degree Round Functions: Application to Full MiMC." in: *Advances in Cryptology - ASIACRYPT 2020*. Ed. by Shiho Moriai and Huaxiong Wang. Vol. 12491. Lecture Notes in Computer Science. Springer, 2020, pp. 477–506. DOI: 10.1007/978-3-030-64837-4_16

[CGG+22] Carlos Cid, Lorenzo Grassi, Aldo Gunsing, Reinhard Lüftenegger, Christian Rechberger, and Markus Schofnegger. "Influence of the Linear Layer on the Algebraic Degree in SP-Networks." In: *IACR Transactions on Symmetric Cryptology* 2022.1 (2022), pp. 110–137. DOI: 10.46586/tosc.v2022.i1.110-137

[GKL+22] Lorenzo Grassi, Dmitry Khovratovich, Reinhard Lüftenegger, Christian Rechberger, Markus Schofnegger, and Roman Walch. "Reinforced Concrete: A Fast Hash Function for Verifiable Computation." In: *SIGSAC Computer and Communications Security - CCS 2022*. Ed. by Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi. ACM, 2022, pp. 1323–1335. DOI: 10.1145/3548606.3560686

Preprints

[HLL+22] Manuel Hauke, Lukas Lamster, Reinhard Lüftenegger, and Christian Rechberger. "A Signature-Based Gröbner Basis Algorithm with Tail-Reduced Reductors (M5GB)." in: *IACR Cryptology ePrint Archive* (2022), p. 987. URL: https://eprint.iacr.org/2022/987

[GKL+23] Lorenzo Grassi, Dmitry Khovratovich, Reinhard Lüftenegger, Christian Rechberger, Markus Schofnegger, and Roman Walch. "Monolith: Circuit-Friendly Hash Functions with New Nonlinear Layers for Fast and Constant-Time Implementations." In: *IACR Cryptology ePrint Archive* (2023), p. 1025. URL: https://eprint.iacr.org/2023/1025

3

SECURITY ESSENTIALS OF CRYPTOGRAPHIC PERMUTATIONS AND BLOCK CIPHERS

In this thesis, we focus on the security of two important classes of cryptographic primitives: cryptographic permutations and block ciphers. To be more precise, we emphasize block ciphers since we employ the same design principles known from block cipher design to construct cryptographic permutations.[1]

The use of cryptographic permutations gained popularity during the SHA-3 competition [Sha]. In particular, the selection of the permutation-based Keccak sponge-function [Dwo15; BDP+11] as the new SHA-3 standard popularized the sponge construction within the cryptographic community. The sponge construction [GJM+11; BDP+11] is a very versatile construction and is nowadays widely used to, e.g., construct hash functions based on cryptographic permutations.

Definition 1. Let $\mathcal{M}$ be a finite set. A *cryptographic permutation of* $\mathcal{M}$ is a bijective function

$$P : \mathcal{M} \longrightarrow \mathcal{M},$$

such that P (and, if necessary, its inverse P^{-1}) is efficiently[2] computable.

Sometimes we call a cryptographic permutation also an *unkeyed* permutation to distinguish it from a keyed permutation, i.e., a block cipher. Unlike for block ciphers with their pseudo-random permutation claim (see Section 3.2), a formal security notion for cryptographic permutations is harder to formulate. After all, a cryptographic permutation admits trivial distinguishers since it is a fixed, known permutation.[3] Hence, the authors of [GJM+11, Section 8.1.4] use the notion of *structural distinguishers* to set them apart from trivial distinguishers. Structural distinguishers include linear and differential distinguishers, integral distinguishers, higher-order differential distinguishers, and distinguishers based on any non-random property in the polynomial representation of a cryptographic permutation. For a more detailed treatment of the security of cryptographic permutations we refer to [GJM+11, Section 8.2].

An important problem for assessing the security of cryptographic permutations is the *Constrained Input Constrained Output Problem* (CICO Problem) [GJM+11, Section 8.2.4]. For a cryptographic permutation it should not be possible to solve the CICO Problem faster than any generic means [GJM+11, Section 8.2.4].

Definition 2 (CICO Problem)**.** Let $\mathcal{M}$ be a finite set and let $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{M}$ be two subsets of $\mathcal{M}$. The CICO Problem for a cryptographic permutation P of $\mathcal{M}$ is phrased as:

$$\text{Find } x \in \mathcal{X} \text{ and } y \in \mathcal{Y} \text{ such that } y = P(x).$$

[1] Indeed, cryptographic permutations are typically constructed as fixed-key block ciphers. This means, the construction of cryptographic permutations boils down to the iteration of simple (and well-chosen) round functions.

[2] Here, the same remark as in Footnote 4 applies.

[3] For a given input-output pair $(x, y) \in \mathcal{M}^2$ of a cryptographic permutation P, it is trivial to check whether $y = P(x)$ and, thus, to distinguish P from a random permutation.

For the rest of this section, we establish the following convention. We say an element s is *randomly chosen* from a finite set S, if s is sampled from S according to a uniform distribution. In other words, each element in S is equally likely and is assigned the probability $|S|^{-1}$.

3.1 Block Ciphers

Block ciphers are a fundamental cryptographic primitive and, at the same time, a building block for other cryptographic primitives, such as *compression functions*, *hash functions*, *pseudo-random number generators*, *message authentication codes*, and *stream ciphers*.

Informally speaking, a block cipher is a function which takes as input a secret key and a plaintext (or message) of *fixed length*, and outputs a ciphertext (or encrypted message) of the same length. Knowing the secret key, it is easy to recover the plaintext from the ciphertext. There are two immediate security goals a block cipher should address.

- Without the secret key k it should be (computationally) infeasible to learn anything about the plaintext.
- Additionally, given a set of plaintext-ciphertext-pairs it should be (computationally) infeasible to recover (part of) the secret key from these pairs.

In short, a block cipher protects the confidentiality of, both, the plaintext encrypted with a secret key, and the secret key itself. The following definition captures the notion of a block cipher more formally.

Definition 3. Let $\mathcal{M}$ and $\mathcal{K}$ be finite sets. A *block cipher* is a family of permutations of $\mathcal{M}$

$$\mathcal{E}(\mathcal{M},\mathcal{K}) := \{E_k : \mathcal{M} \longrightarrow \mathcal{M},\ k \in \mathcal{K},\ E_k \text{ a permutation}\},$$

parametrized by $k \in \mathcal{K}$, such that E_k and E_k^{-1} are efficiently[4] computable for all $k \in \mathcal{K}$. Here, E_k^{-1} is the (compositional) inverse of E_k, i.e., it holds

$$E_k \circ E_k^{-1} = E_k^{-1} \circ E_k = Id_{\mathcal{M}}, \text{ for all } k \in \mathcal{K}.$$

Furthermore, we use the following denominations.

- An instance $E_k \in \mathcal{E}(\mathcal{M},\mathcal{K})$, for $k \in \mathcal{K}$, is commonly called a block cipher as well.[5]
- Instead of block cipher we also use the term *keyed cryptographic permutation*.
- The set $\mathcal{M}$ is called the *message space*, and $\mathcal{K}$ is called the *key space*.
- The number $\log_2(|\mathcal{M}|)$ is called the *block size* or *state size*.
- The number $\log_2(|\mathcal{K}|)$ is called the *key size* or *size of the keyspace*.

[4] "Efficiently" in this context means, given $k \in \mathcal{K}$ and $m \in \mathcal{M}$ it is easy compute $E_k(m)$, and $E_k^{-1}(m)$, respectively. Since $\mathcal{K}, \mathcal{M}$ are fixed finite sets, any notion of efficiency in terms of asymptotic complexity is not fruitful here. We deliberately do not give a precise definition of "easy" and hope, the reader will understand what is meant here.

[5] We believe, this causes no confusion, as any concrete block cipher E_k is understood to be a member of a family permutations parametrized by some set $\mathcal{K}$.

- Let $c = E_k(p)$. The message p is called *plaintext*, and the encrypted message c is called *ciphertext*.

If the sets $\mathcal{M}$ and $\mathcal{K}$ are clear from the context, we also just write $\mathcal{E}$ instead of $\mathcal{E}(\mathcal{M}, \mathcal{K})$. We emphasize, Definition 3 does not say anything about what properties a "good" block cipher should have. The family $\mathcal{E}(\mathcal{M}, \mathcal{K})$ could be a single permutation, e.g., the identity on $\mathcal{M}$, that is repeated $|\mathcal{K}|$ times. Of course, we do not regard such a block cipher as a "good" one. We shall see later in Section 3.2 what properties we require from a block cipher to consider it "good", i.e., secure.

The block size determines the (theoretical) number of all possible permutations that a block cipher might cover. The key size determines the number of permutations that are actually covered. It is interesting to ask, how large the key size would have to be, to cover all possible permutations of $\mathcal{M}$. The number of permutations of $\mathcal{M}$ is $|\mathcal{M}|!$. According to Stirling's formula, this is approximated by

$$|\mathcal{M}|! \approx \sqrt{2\pi|\mathcal{M}|}\left(\frac{|\mathcal{M}|}{e}\right)^{|\mathcal{M}|}, \quad \text{for } |\mathcal{M}| \text{ large.}$$

Since

$$\sqrt{2\pi|M|}\left(\frac{|\mathcal{M}|}{e}\right)^{|\mathcal{M}|} < \left(\frac{|\mathcal{M}|}{2}\right)^{|\mathcal{M}|}, \quad \text{for } |\mathcal{M}| \text{ large,}$$

with a key size of

$$\log_2(|\mathcal{K}|) = |\mathcal{M}| \cdot \log_2(|\mathcal{M}|/2),$$

bits, one could cover all permutations of $\mathcal{M}$. For a typical block size of $\log_2(|\mathcal{M}|) = 128$ bits, this would imply an exorbitantly large key size, which is far beyond any practical usability. In practice, much smaller key sizes suffice, e.g.,

$$\log_2(|\mathcal{K}|) = 256.$$

In this example, a block cipher family "only" covers at most 2^{256} different permutations. However, this is more than enough to thwart any exhaustive search of the key space $\mathcal{K}$ with current (and foreseeable) computing resources. Nevertheless, it is a basic design principle of block ciphers that for a randomly chosen key $k \in \mathcal{K}$ the permutation E_k should appear to be a randomly chosen permutation from all permutations of $\mathcal{M}$. We discuss this aspect further in the following section.

3.2 Security of Block Ciphers

When designing and analyzing symmetric cryptographic primitives, the literature commonly distinguishes between the roles of *designer* and *cryptanalyst*. The designer proposes a cryptosystem, in conjunction with concrete security claims. The cryptanalyst tries to "break" a cryptosystem, i.e., to invalidate the stated security claims. Often, the cryptanalyst is seen as an *adversary*, and, thus, is also called *attacker*. We adopt a more neutral diction, which is why we use the term cryptanalyst, or short *analyst*, instead of adversary or attacker.

When analyzing a block cipher $\mathcal{E}(\mathcal{M}, \mathcal{K})$, it is customary to work with the following assumptions.

- All keys in $\mathcal{K}$ are equally likely to be chosen.

- For every $k \in \mathcal{K}$, the cryptanalyst knows all details of E_k and E_k^{-1}, except for the secret key k itself.

The second assumption is also known as *Kerckhoff's principle* [Ker83a; Ker83b] and its aim is to facilitate public analysis of a block cipher, while its security rests entirely upon the secret key k. Open-sourcing the design[6] of a block cipher, i.e., considering it as public knowledge, reflects the assumption that any cryptanalyst will gain full familiarity with the design anyway. By one way or another, and no matter if kept secret. In addition, if everyone knows the details of a block cipher and is free to assess its security, the confidence in this block cipher increases.

When we speak of the power of the cryptanalyst we mean three types of resources: time, data, and memory. Time and memory, in terms of computational resources, are usually measured in a way that is agnostic to the underlying hardware.[7] "Data" refers to a set of plaintext-ciphertext-pairs $(p, c) \in \mathcal{M}^2$, all encrypted under the same fixed, but unknown, key $k \in \mathcal{K}$. The usual model for collecting plaintext-ciphertext pairs is the following: a cryptanalyst A interacts with an *oracle* $\mathcal{O}_F$ for a function $F : \mathcal{M} \longrightarrow \mathcal{M}$, such that A sends $x \in \mathcal{M}$ to $\mathcal{O}_F$ and receives back $F(x)$. The function F either implements E_k or E_k^{-1} for a fixed, but unknown, key $k \in \mathcal{K}$. Only the corresponding oracle is able to generate valid plaintext-ciphertext-pairs. However, due to Kerckhoffs's principle, A knows the whole family $\mathcal{E}$ and may implement $E_{k'}$ or $E_{k'}^{-1}$, for any self-chosen key $k' \in \mathcal{K}$.

We call the amount of time, data, and memory a cryptanalyst requires also the time, data and memory *complexity*. We emphasize, in contrast to any asymptotic theory of computational complexity, in this context the term "complexity" carries a rather concrete meaning. The cryptographic literature knows the following commonly used meanings.

1. *Time complexity*. A measure for the time needed to carry out a certain analysis. One time unit, usually, is one evaluation of E_k (or E_k^{-1}). Another common time unit is one arithmetic operation over $\mathcal{M}$.

2. *Data complexity*. The amount of plaintext-ciphertext pairs that are necessary for a certain analysis. In other words, the number of queries to an oracle of E_k (or E_k^{-1}). One unit is one plaintext-ciphertext pair.

3. *Memory complexity*. A measure for the memory needed to carry out a certain analysis. Typically, one unit is one bit or, also, the bit-length of an element in $\mathcal{M}$, i.e., $\log_2(|\mathcal{M}|)$.

From a conceptual point of view, it is meaningful to classify the security of block ciphers according to the *goal* of the cryptanalyst and the data access that is granted to the cryptanalyst, i.e., the *model* of cryptanalysis.

Goals of Cryptanalysis We align ourselves with the classification in [Knu94b; VTJ14] and formulate four basic goals for the cryptanalysis of block ciphers. In the following, let $E_k \in \mathcal{E}(\mathcal{M}, \mathcal{K})$ be an arbitrary block cipher.

1. *Key-Recovery*. The cryptanalyst recovers the secret key k from a set of ciphertexts, or pairs of plaintexts and corresponding ciphertexts.

[6] "Design" in this context refers to the internal details of a block cipher, much like a construction plan.

[7] Implicitly, often a RAM model of computation is assumed.

2. *Global Deduction*. The cryptanalyst constructs a function F that is functionally equivalent[8] to E_k or E_k^{-1}, *without* knowing the secret-key k.

3. *Local Deduction*. Same goal as a global deduction, with the difference that the functional equivalence is only valid on a subset of all possible plaintexts or ciphertexts.

4. *Distinguisher*. The cryptanalyist can efficiently distinguish between two black boxes; one contains the block cipher with a randomly chosen key, while the other contains a randomly chosen permutation.

Models of Cryptanalysis Above goals of cryptanalysis may differ in the data resources that they grant the cryptanalyst. The cryptographic literature knows the following basic models (see, e.g., [VTJ14]).

1. *Ciphertext-only analysis*. The cryptanalyst only knows a set of ciphertexts, but has no information about the corresponding plaintexts.

2. *Known plaintext analysis*. The cryptanalyst knows a set of plaintexts $p_1, \ldots, p_l$ and corresponding ciphertexts $c_1, \ldots, c_l$. But the cryptanalyst has no control over the pairs of plaintexts and ciphertexts that are available.

3. *Chosen plaintext analysis*. The cryptanalyst a priori chooses a set of plaintexts $p_1, \ldots, p_l$ and obtains the corresponding ciphertexts $c_1, \ldots, c_l$.

4. *Adaptively chosen plaintext analysis*. The cryptanalyst chooses a set of plaintexts $p_1, \ldots, p_l$ interactively while receiving their corresponding ciphertexts $c_1, \ldots, c_l$. In other words, the cryptanalyst chooses p_1, obtains c_1, then chooses p_2, and obtains c_2, and so forth.

5. *(Adaptively) Chosen ciphertext analysis*. Similar to a (adaptively) chosen plaintext analysis, with the difference that the cryptanalyst may (adaptively) choose ciphertexts.

Models 1. to 4. are listed in increasing strength, in the sense that the power of the cryptanalyst increases. A successful analysis under model $i-1$ is also successful under model i. Seen from another perspective, if a block cipher resists analysis under model i, it is also resistant against analysis under model $i-1$. The (adaptively) chosen ciphertext model in 5. is considered on a par with the corresponding (adaptively) chosen plaintext model. Hence, a block cipher that resists (adaptively) chosen plaintext and ciphertext cryptanalysis provides the strongest security guarantee. In contrast, a cipher that only resists ciphertext-only attacks provides the weakest security guarantee. From a design perspective, it would be ideal to argue that a cipher is secure against any form of (adaptively) chosen plaintext and ciphertext analysis, even if in practice a cryptanalyst might never be that powerful. In practice, however, the common approach in cryptanalysis is to find security arguments against *any known means of analysis*.

Block Ciphers as Pseudo-Random Permutations Colloquially speaking, the primary security notion of a block cipher $\mathcal{E}(\mathcal{M}, \mathcal{K})$ is the following. The family of $|\mathcal{K}|$ permutations should "statistically look like" the family of all $|\mathcal{M}|!$ permutations of $\mathcal{M}$ *if* the data and computing resources to distinguish between the two of them are *bounded*. This means, for any computationally

[8] Functional equivalence of two functions E and F means $E(x) = F(x)$, for all $x \in \mathcal{M}$.

bounded procedure, a permutation parametrized by a randomly chosen key $k \in \mathcal{K}$ should appear to be a randomly chosen permutation from the set of all possible permutations. In our next considerations, we will motivate and formalize this intuitive notion of "statistical likeness". Eventually, this will lead us to the fundamental concept of pseudo-random permutations (see Definition 6 and Definition 7). At the heart of this concept lies the paradigm

> "Two objects are viewed as equal if they cannot be told apart by any efficient procedure."

One vital thing to keep in mind is, when we talk about "statistical likeness" of permutations, we are dealing with *families* of permutations rather than individual permutations. To be even more precise, we are dealing with *distributions* over families of permutations. When we talk about distributions, in this context we mean discrete probability distributions over a finite domain $\mathcal{M}$. An important example is the uniform distribution over $\mathcal{M}$ that assigns each element in $\mathcal{M}$ an equal probability of $|\mathcal{M}|^{-1}$. With above preparatory remarks at hand, we formulate a first approximation of pseudo-random permutations.

> "A family of permutations on $\mathcal{M}$ is called pseudo-random if it cannot be told apart from the family of all permutations on $\mathcal{M}$ by any efficient probabilistic algorithm."

An algorithm is called probabilistic if it can take random steps. A random step is a random choice of several predetermined steps to take next, where each step is equally likely. We call these choices the algorithm's *(internal) randomness* or *(internal) coin tosses*. Consequently, for each input the corresponding output of a probabilistic algorithm is described by a probability distribution, where the probability is taken over the algorithm's randomness. It remains to discuss what we mean by an "efficient" procedure.

When designing and analyzing block ciphers, we usually formulate security with respect to a fixed security level and determine relevant parameters of a block cipher accordingly. Hence, any *asymptotic* notion (e.g., probabilistic polynomial time algorithms or negligible functions) might not be the right tool to assess the security for concrete parameter values.[9] As an example, let $\mathcal{P}$ denote the set of all permutations on $\mathcal{M}$. Instead of requiring that no probabilistic polynomial-time algorithm can distinguish $\mathcal{E}$ from $\mathcal{P}$ with non-negligible probability[10], we demand that no algorithm running in time at most t and making at most q queries can distinguish $\mathcal{E}$ and $\mathcal{P}$ with probability better than ϵ.

In our presentation, we align ourselves with the discussion outlined in [KL21, Section 3.1.1] and [MF21, Section 2.2.2], and adopt a *concrete* approach to security that is formulated relative to a concrete security parameter λ. The parameter λ upper bounds the resources, or complexities, we grant a cryptanalyst, as well as the success probability of the cryptanalyst. Any adequate notion of complexity should consider the time, data and memory resources needed for cryptanalysis. A natural upper bound for the data and memory complexity is the length $|\mathcal{M}|$ of the full *codebook*, i.e., the set of all possible plaintext-ciphertext pairs. Whereas, a natural upper bound for the time complexity is $|\mathcal{K}|$, the time to try out all possible keys. The concrete approach to security we are adopting and using to eventually define

[9] Any block cipher $\mathcal{E}(\mathcal{M}, \mathcal{K})$ is, theoretically, broken by an exhaustive key search in time $|\mathcal{K}|$ or a dictionary analysis with $|\mathcal{M}|$ data and memory.

[10] In theoretical cryptography, this is a canonical definition of a pseudo-random permutation.

pseudo-random permutations originates with Bellare, Kilian, and Rogaway [BKR94; BKR00] and builds upon the viewpoint of Luby and Rackoff [LR88]. Luby and Rackoff use the term "pseudo-random permutation" to denote a permutation that is computationally indistinguishable from the family of all functions. In contrast, Mihir, Bellare and Rogaway define pseudo-random permutations as permutations that are computationally indistinguishable from the family of all permutations, not the family of all functions. See also [BKR98, Section 2.4] for a brief historical discussion and [BKR94, Section 1.2] for a collection of other works which address the notion of concrete (or exact) security prior to [BKR94].

We return to our clarification and explain what we mean by an "efficient" procedure. Informally speaking and taking into account above concrete approach to security, in this context we say:

> "An 'efficient' procedure is any procedure that stays within the predefined complexity limits for time, data, and memory."

The following definitions formalize our discussion from above.

Definition 4. Let $\mathcal{M}$ and $\mathcal{K}$ be finite sets and let $\mathcal{P} := \mathcal{P}(\mathcal{M})$ be the set of all permutations of $\mathcal{M}$. Let

$$\mathcal{E} := \mathcal{E}(\mathcal{M}, \mathcal{K}) := \{E_k : \mathcal{M} \longrightarrow \mathcal{M},\ k \in \mathcal{K},\ E_k \text{ a permutation}\},$$

be a block cipher. The (t, q)-*pseudo-random (distinguishing) advantage* of $\mathcal{E}$ is defined as

$$\mathrm{Adv}^{\mathrm{PRP}}_{\mathcal{E}}(t, q) := \max_{A_{t,q}} |Pr_{k \leftarrow \mathcal{K}}(A_{t,q}(E_k) = 1) - Pr_{\pi \leftarrow \mathcal{P}}(A_{t,q}(\pi) = 1)|.$$

Here, $A_{t,q}(F)$ is a (computationally bounded) probabilistic algorithm that takes as input an oracle $\mathcal{O}_F$ for a function $F : \mathcal{M} \longrightarrow \mathcal{M}$ and that outputs a bit $b \in \{0, 1\}$. $A_{t,q}$ makes at most q oracle queries and runs in time at most t.

In Definition 4, the first probability is taken over the randomness of $A_{t,q}$ and randomly chosing $k \in \mathcal{K}$, and the second probability is taken over the randomness of $A_{t,q}$ and randomly chosing $\pi \in \mathcal{P}$. The PRP advantage corresponds to the scenario of a chosen plaintext analysis. The scenario where a cryptanalyst, additionally, has access to an oracle of E_k^{-1} corresponds to a chosen plaintext and ciphertext analysis. To cover this scenario, we introduce the notion of strong PRP advantage.

Definition 5. Let $\mathcal{M}$ and $\mathcal{K}$ be finite sets and let $\mathcal{P} := \mathcal{P}(\mathcal{M})$ be the set of all permutations of $\mathcal{M}$. Let

$$\mathcal{E} := \mathcal{E}(\mathcal{M}, \mathcal{K}) := \{E_k : \mathcal{M} \longrightarrow \mathcal{M},\ k \in \mathcal{K},\ E_k \text{ a permutation}\},$$

be a block cipher. The *strong* (t, q)-*pseudo-random (distinguishing) advantage* of $\mathcal{E}$ is defined as

$$\mathrm{Adv}^{\mathrm{sPRP}}_{\mathcal{E}}(t, q) := \max_{A_{t,q}} |Pr_{k \leftarrow \mathcal{K}}(A_{t,q}(E_k, E_k^{-1}) = 1) - Pr_{\pi \leftarrow \mathcal{P}}(A_{t,q}(\pi, \pi^{-1}) = 1)|.$$

Here, $A_{t,q}$ is a (computationally bounded) probabilistic algorithm that takes as input two oracles $\mathcal{O}_F, \mathcal{O}_G$ for functions $F, G : \mathcal{M} \longrightarrow \mathcal{M}$ and that outputs a bit $b \in \{0, 1\}$. $A_{t,q}$ makes at most q oracle queries and runs in time at most t.

Definition 6 (Section 2.3, [BKRoo])**.** A block cipher $\mathcal{E}$ is called a (t, q, ϵ)-pseudo-random permutation (PRP) if the pseudo-random distinguishing advantage $\mathrm{Adv}_{\mathcal{E}}^{\mathrm{PRP}}(t, q)$ is smaller than ϵ.

The definition of a strong pseudo-random permutation is completely analogous.

Definition 7. A block cipher $\mathcal{E}$ is called a strong (t, q, ϵ)-pseudo-random permutation (sPRP) if the strong (t, q)-pseudo-random distinguishing advantage $\mathrm{Adv}_{\mathcal{E}}^{\mathrm{sPRP}}(t, q)$ is smaller than ϵ.

For which values of ϵ do we consider $\mathcal{E}$ secure, and for which not? We presume to keep this point informal, because, in fact, it is the task of dedicated cryptanalysis to find a (lower) bound on the advantage of a given distinguishing algorithm. As a rule of thumb we state, if any distinguishing algorithm $A_{t,q}$ achieves a "high" advantage ϵ within the complexity limits defined by q and t, then we consider this block cipher as broken. To get a feel for the large values of t and the small values of ϵ, that are typical of modern cryptosystems we refer to [KL21, Section 3.1.1]

The standard notion of security for block ciphers is the (strong) pseudo-random permutation assumption introduced in Definition 6 (and Definition 7, respectively). As a more colloquial notion of security, we formulate:

> "A block cipher is considered secure if no cryptanalyst is able to deduce any information about the plaintext, the ciphertext or the secret key more efficiently than any generic means of analysis."

In the following, we describe two important generic means for analyzing block ciphers: exhaustive key search and dictionary analysis. A minimal requirement for a secure block cipher, thus, would be: the key size should be large enough to prevent exhaustive search of the key space and the block size should be large enough to prevent dictionary attacks.

Exhaustive (Key) Search Let (p, c) be a known plaintext-ciphertext-pair that has been encrypted under a fixed, but unknown, key $k \in \mathcal{K}$. In short, we have the relation $E_k(p) = c$. It takes $|\mathcal{K}|$ steps to exhaustively try all possible keys in $\mathcal{K}$ and test whether a key $k' \in \mathcal{K}$ satisfies the relation $E_{k'}(c) = p$. A satisfying key k' is considered a candidate for the correct key k. If $\mathcal{E}$ is a pseudo-random permutation, we expect

$$\frac{|\mathcal{K}|}{|\mathcal{M}|}$$

satisfying keys k', with the correct key k being one of them. Hence, if $|\mathcal{K}| < |\mathcal{M}|$ we expect a single satisfying key - the correct key k. If $|\mathcal{K}| > |\mathcal{M}|$, we might need more than one plaintext-ciphertext pair to identify the correct key among all candidate keys. Let us assume we have two known plaintext-ciphertext-pairs $(p_1, c_1), (p_2, c_2)$, both encrypted under the same key k. Exhaustively trying all keys $k' \in \mathcal{K}$ and testing whether $E_{k'}(p_1) = c_1$, $E_{k'}(p_2) = c_2$ is satisfied, we expect to find

$$\frac{\mathcal{K}}{|\mathcal{M}| \cdot (|\mathcal{M}| - 1)}$$

satisfying keys k'. More generally, given l known plaintext-ciphertext-pairs $(p_1, c_1), \ldots, (p_l, c_l)$, all encrypted under the same, fixed but unknown, key $k \in \mathcal{K}$, we expect that

$$\frac{|\mathcal{K}|}{|\mathcal{M}| \cdot (|\mathcal{M}| - 1) \cdot \cdots \cdot (|\mathcal{M}| - l + 1)} \approx \frac{|\mathcal{K}|}{|\mathcal{M}|^l}$$

keys $k' \in \mathcal{K}$ satisfy $E_{k'}(p_1) = c_1, \ldots, E_{k'}(p_l) = c_l$. Hence, we expect that

$$\frac{\log_2(|\mathcal{K}|)}{\log_2(|\mathcal{M})|}$$

plaintext-ciphertext pairs are enough to single out the correct key k.

If $\mathcal{X}$ is the random variable describing the number of random tries until the correct key has been identified, the expected time complexity for an exhaustive key search is given by[11]

$$\sum_{i=1}^{|\mathcal{K}|} i \cdot P(\mathcal{X} = i) = \sum_{i=1}^{|\mathcal{K}|} \frac{i}{|\mathcal{K}|} = \frac{|\mathcal{K}| \cdot (|\mathcal{K}| + 1)}{2 \cdot |\mathcal{K}|} \approx \frac{|\mathcal{K}|}{2}.$$

This means, on average it takes $|\mathcal{K}|/2$ guesses until the correct key has been identified. The memory requirement of exhaustive search is neglible and only a few known plaintext-ciphertex pairs are needed.

Dictionary Analysis as Global Deduction Theoretically, a dictionary analysis may be assembled if we collect the ciphertexts of all $|\mathcal{M}|$ plaintexts and store them in a table. This takes $|\mathcal{M}|$ memory and data, but negligible time. Thereafter, we are able to encrypt (or decrypt) without any knowledge about the secret key. Such a dictionary analysis is also the reason, why the state size of a block cipher should not be too small.

Dictionary Analysis as Key Recovery Another form of (theoretical) dictionary analysis is the following. We encrypt a fixed plaintext p under all possible keys $k' \in \mathcal{K}$ and store the corresponding ciphertexts $E_{k'}(p)$ in a table. Afterwards, we request the ciphertext $c = E_k(p)$ of p under the unknown key $k \in \mathcal{K}$ and check which table entries satisfy

$$E_{k'}(p) = E_k(p).$$

If $\mathcal{E}$ is a pseudo-random permutation, we expect to find

$$\frac{|\mathcal{K}|}{|\mathcal{M}|}$$

matching entries. The rest of the discussion is analogous to our discussion of exhaustive key search. In particular, we expect that

$$\frac{\log_2(|\mathcal{K}|)}{\log_2(|\mathcal{M})|}$$

chosen plaintext-ciphertext pairs suffice to single out the correct key. This type of dictionary attack requires $|\mathcal{K}|$ time and memory, but only a few chosen plaintext-ciphertext pairs. It is the reason why the key size of a block cipher should not be too small.

The following listing summarizes the complexity requirements, in terms of time, memory and data, of above generic means of analysis.

- *Exhaustive (key) search.* $|\mathcal{K}|$ (or $|\mathcal{K}|/2$) time, negligible data (a few known plaintext-ciphertext pairs), negligible memory.

[11] For $n = |\mathcal{K}|$, the probability to guess the correct key on the i-th attempt after $i-1$ wrong guesses (with discarding wrongly guessed keys) is $\frac{n-1}{n} \cdot \frac{n-2}{n-1} \cdots \frac{n-i+1}{n-i+2} \cdot \frac{1}{n-i+1} = \frac{1}{n}$.

- *Dictionary analysis*. $|\mathcal{K}|$ time, $|\mathcal{K}|$ memory, negligible data (only a few plaintext-ciphertext pairs) in a key-recovery dictionary analysis. In a dictionary analysis as global deduction, we require $|\mathcal{M}|$ memory and data, and negligible time.

We conclude our discussion of generic block cipher cryptanalysis with some further references. For the topic of time-memory tradeoffs we recommend [Hel80; BPV98], for a discussion of the effective key-length of block ciphers we refer to [HL14], and for general further reading we refer to [KR11; SSL15; Ava16].

4 DIFFERENT FLAVORS OF ALGEBRAIC CRYPTANALYSIS

The fact that most primitives in symmetric cryptography operate over *finite fields* provides the starting point and the theoretical underpinning of algebraic cryptanalysis.[1] At its core, algebraic cryptanalysis exploits a fundamental property of finite fields, namely, that all functions over finite fields can be expressed as polynomial functions. We give an account of this property in Proposition 1.

Proposition 1. *Every map $f : \mathbb{F}_q \to \mathbb{F}_q$ on a finite field $\mathbb{F}_q$ can be uniquely described as a univariate polynomial over $\mathbb{F}_q$ with maximum degree $q-1$.*

Proof. For existence, consider the polynomial

$$F(X) := \sum_{a \in \mathbb{F}_q} f(a)(1 - (X-a)^{q-1}). \tag{1}$$

Or, alternatively, the polynomial

$$F(X) := \sum_{a \in \mathbb{F}_q} f(a) \prod_{\substack{b \in \mathbb{F}_q \\ b \neq a}} \frac{X-b}{a-b}. \tag{2}$$

For uniqueness, observe, if there are two polynomials $F, G \in \mathbb{F}_q[X_1, \ldots, X_n]$ of degree at most $q-1$ with

$$F(x) = f(x) = G(x), \text{ for all } x \in \mathbb{F}_q,$$

then $F - G$ has q roots. Since a non-zero polynomial of degree at most $q-1$ has at most $q-1$ zeroes it follows $F - G = 0$ and thus, $F = G$. □

An analogous result holds for multivariate functions over $\mathbb{F}_q^n$, see the following Proposition 2.

Proposition 2. *Every map $f : \mathbb{F}_q^n \to \mathbb{F}_q$ can be uniquely described as a multivariate polynomial over $\mathbb{F}_q$ in n variables with maximum degree $q-1$ in each variable.*

Proof. For existence, consider the polynomial

$$\varphi(f)(X_1, \ldots, X_n) := \sum_{(a_1, \ldots, a_n) \in \mathbb{F}_q^n} f(a_1, \ldots, a_n) \prod_{1 \leq i \leq n} (1 - (X_i - a_i)^{q-1}). \tag{3}$$

Uniqueness follows from a cardinality argument: the two finite sets

$$\mathcal{S} := \mathbb{F}_q[X_1, \ldots, X_n]/(X_1^q - X_1, \ldots, X_n^q - X_n)$$

and

$$\mathcal{R} := \{f : \mathbb{F}_q^n \to \mathbb{F}_q\}$$

[1] It might be more precise to say "can be regarded to operate over finite fields", since, e.g., the set of bit strings of length n natively only carries the structure of a $\mathbb{F}_2$-vector space but can be endowed with a field structure (coming from $\mathbb{F}_{2^n}$). For more details, see the remark after Proposition 2.

have the same cardinality q^{q^n}. Furthermore, the map $\varphi : \mathcal{R} \to \mathcal{S}$ with $f \mapsto \varphi(f)$ is injective. Indeed, if for two functions $f, g : \mathbb{F}_q^n \to \mathbb{F}_q$ it holds $\varphi(f) = \varphi(g)$, then by Eq. (3) and since $\iota : \mathbb{F}_q^n \to \mathbb{F}_q$

$$\iota(x_1, \ldots, x_n) := \prod_{1 \leq i \leq n} (1 - (x_i - a_i)^{q-1})$$

is the indicator function of $\{(a_1, \ldots, a_n)\}$ this yields

$$f(a_1, \ldots, a_n) = g(a_1, \ldots, a_n), \text{ for all } (a_1, \ldots, a_n) \in \mathbb{F}_q^n,$$

and consequently $f = g$. Since a function on a finite domain is injective if and only if it is surjective, we conclude that φ is bijective. □

As an immediate consequence of Proposition 1 we state the following corollary.

Corollary 1. *Let $(x_0, y_0), \ldots, (x_d, y_d)$ be a set of $d + 1$ pairs of elements in $\mathbb{F}_q$ such that $x_i \neq x_j$ for $i \neq j$ (which implies $d \leq q$). Then there exists a unique polynomial $F \in \mathbb{F}_q[X]$ with maximum degree d such that $F(x_i) = y_i$, for all $i = 0, 1, \ldots, d$.*

Almost all cryptographic primitives in symmetric cryptography operate over finite fields or can be interpreted to work over finite fields. For example, "binary primitives" that manipulate bit-strings of length n, are, in fact, functions working over the n-fold cartesian product $\mathbb{F}_2^n$. The set $\mathbb{F}_2^n$ naturally carries the structure of an $\mathbb{F}_2$-vector space and, furthermore, is isomorphic (as vector space) to the finite field $\mathbb{F}_{2^n}$ with 2^n elements. What is more, the field structure of $\mathbb{F}_{2^n}$ carries over to $\mathbb{F}_2^n$, or, in other words, $\mathbb{F}_2^n$ can be endowed with the field structure of $\mathbb{F}_{2^n}$. To see this, choose a vector space basis $\{a_1, \ldots, a_n\} \subseteq \mathbb{F}_{2^n}$ of $\mathbb{F}_{2^n}$ and consider the bijective map

$$\kappa = \kappa_{a_1, \ldots, a_n} : \mathbb{F}_2^n \longrightarrow \mathbb{F}_{2^n}, (x_1, \ldots, x_n) \mapsto x_1 a_1 + \ldots + x_n a_n.$$

Then the multiplication on $\mathbb{F}_2^n$ induced by κ

$$(x_1, \ldots, x_n) \cdot (y_1, \ldots, y_n) := \kappa^{-1} \left(\kappa(x_1, \ldots, x_n) \cdot \kappa(y_1, \ldots, y_n) \right) \tag{4}$$

together with coordinate-wise addition of vectors gives $\mathbb{F}_2^n$ the structure of a field. Note that the second operator $\cdot$ in Eq. (4) denotes field multiplication in $\mathbb{F}_{2^n}$.

The preceding discussion is the reason, why we use $\mathcal{M} = \mathbb{F}_q$ as native domain for a block cipher $\mathcal{E}(\mathcal{M}, \mathcal{K})$ in the rest of this section. There are different branches of algebraic cryptanalysis. Each branch exploits the fact that functions over finite fields are, indeed, polynomial functions in a different manner. We motivate the following methods of algebraic cryptanalysis:

- polynomial interpolation,
- higher-order differential distinguishers, and
- algebraic equation solving via Gröbner bases.

Other established means of cryptanalysis, considered as algebraic in the literature, include: the division property [Tod15] and its dual notion monomial prediction [HSW+20], cube distinguishers [DS09], zero-sum and integral distinguishers [DKR97; KR07; AM09], and linearization [KS99] including the XL-family of algorithms [Cou02]. Each of these methods spans its own line of research and is an important topic in its own regard. However, discussing them more comprehensively lies beyond the scope of this thesis.

4.1 Polynomial Interpolation

Polynomial interpolation (or *interpolation cryptanalysis*) as a means of algebraic cryptanalysis was introduced in 1997 by Thomas Jakobsen and Lars Knudsen [JK97] to successfully analyze block ciphers proven to be secure against linear and differential analysis. Further and more recent research works can be found in [YG00; DLM+15; EGL+20; LP19; ZZD+21; RAS21].

The conceptual basis of interpolation cryptanalysis are Proposition 1, Proposition 2, and Corollary 1. Interpolation cryptanalysis aims at constructing the polynomial representing the encryption (or decryption) function from a set of known or chosen plaintext-ciphertext-pairs. In addition, it might also aim at recovering (part of) the secret-key. In general, an instance of a block cipher $E_k : \mathbb{F}_q \longrightarrow \mathbb{F}_q$ is considered secure against interpolation cryptanalysis, if the polynomial representation[2] of E_k in $\mathbb{F}_q[X]/\langle X^q - X\rangle$ given by

$$P(X) = \sum_{i=1}^{q-2} a_i X^i \in \mathbb{F}_q[X]/\langle X^q - X\rangle,$$

has a degree close to the maximum degree $q-2$ and a number of non-zero coefficients close to $q-1$.[3]

The rest of this section is devoted to a discussion of different approaches to polynomial interpolation. In the following, let $E : \mathbb{F}_q \longrightarrow \mathbb{F}_q$ be a fixed function (e.g., an instance of a block cipher). Let

$$\hat{E}(X) = \sum_{i=0}^{q-1} a_i X^i \in \mathbb{F}_q[X]/\langle X^q - X\rangle$$

denote the (unique) polynomial representation of E over $\mathbb{F}_q$. The aim of polynomial interpolation is to recover the coefficient vector $(a_0, \ldots, a_{q-1})$ from the value vector $(y_0, \ldots, y_{q-1})$, where $y_i := E(x_i)$ and $\mathbb{F}_q = \{x_0, x_1, \ldots, x_{q-1}\}$.

A textbook approach for interpolating E is to set up the equation system

$$y_i = E(x_i) = a_0 + a_1 x_i + \cdots a_{q-1} x_i^{q-1},\ 0 \leq i \leq q-1,$$

of q linear equations over $\mathbb{F}_q$ in the q variables $a_0, a_1, \ldots, a_{q-1}$. Subsequently, we solve the equation system for $a_0, a_1, \ldots, a_{q-1}$. In other words, we solve the following matrix equation

$$\boldsymbol{y} = V \cdot \boldsymbol{a},$$

with $\boldsymbol{y} = (y_0, \ldots, y_{q-1})^t$, $\boldsymbol{a} = (a_0, \ldots, a_{q-1})^t$ and

$$V = \begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^{q-1} \\ 1 & x_1 & x_1^2 & \ldots & x_1^{q-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{q-1} & x_{q-1}^2 & \ldots & x_{q-1}^{q-1} \end{pmatrix}.$$

Inverting V recovers the coefficient vector $\boldsymbol{a}$. Setting up above equation system takes q data points[4] $(x_0, y_0), \ldots, (x_{q-1}, y_{q-1})$. The matrix V is also called a *Vandermonde matrix* and inverting it can be done in

$$\mathcal{O}\left(q^\omega\right)$$

[2] For a polynomial $P \in \mathbb{F}_q[X]$, the term *polynomial representation* of E_k refers to the property $P(x) = E_k(x)$, for all $x \in \mathbb{F}_q$.

[3] To see why a permutation polynomial of $\mathbb{F}_q$, for $q \geq 3$, has at most degree $q-2$, consider the fact that for $\alpha \in \mathbb{F}_q$, $\alpha \neq 0, 1$, the map $x \mapsto \alpha x$ is a bijection of $\mathbb{F}_q$. Hence, $\sum_{x\in\mathbb{F}_q} x = \sum_{x\in\mathbb{F}_q} \alpha x$ and therefore $(1-\alpha)\sum_{x\in\mathbb{F}_q} x = 0$. This means, the coefficient of X^{q-1} in Eq. (1) is 0.

[4] I.e., q (known) plaintext-ciphertext-pairs in case E is an instance of a block cipher.

field operations in $\mathbb{F}_q$, where $2 \leq \omega \leq 3$ is the exponent of matrix multiplication [GCL92, p.129]. A more general approach for interpolating E is to choose any vector space basis

$$\{\alpha_0(X), \ldots, \alpha_{q-1}(X)\}$$

for the q-dimensional $\mathbb{F}_q$-vector space $\mathbb{F}_q[X]/\langle X^q - X\rangle$ and solve the following system of equations

$$y_i = E(x_i) = a_0\alpha_0(x_i) + a_1\alpha_1(x_i) + \cdots a_{q-1}\alpha_{q-1}(x_i),\ 0 \leq i \leq q-1,$$

for $a_0, \ldots, a_{q-1}$, and, again, $\mathbb{F}_q = \{x_0, \ldots, x_{q-1}\}$. Clearly, choosing the canonical basis $\{1, X, \ldots, X^{q-1}\}$ coincides with the previous approach of constructing the Vandermonde matrix. A well-known basis is the Lagrange basis $\{L_0(X), L_1(X), \ldots, L_{q-1}(X)\}$, with

$$L_i(X) := \prod_{\substack{j=0 \\ j\neq i}}^{d} \frac{X - x_j}{x_i - x_j} \in \mathbb{F}_q[X]\,/\langle X^q - X\rangle.$$

Since $L_i(x) = 1$, if $x = x_i$, and $L_i(x) = 0$, for $x \neq x_i$, the Lagrange basis leads to the matrix equation

$$\boldsymbol{y} = L \cdot \boldsymbol{a},$$

with the particularly convenient $q \times q$ identity matrix

$$L = \begin{pmatrix} 1 & 0 & 0 & \ldots & 0 \\ 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 1 \end{pmatrix}.$$

Another important basis is the Newton basis $\{N_0(X), N_1(X), \ldots, N_{q-1}(X)\}$ with[5]

$$N_i(X) := \prod_{j=0}^{i-1}(X - x_j) \in \mathbb{F}_q[X]\,/\langle X^q - X\rangle.$$

The property $N_i(x_k) = 0$, for all $k < i$, leads to the matrix equation

$$\boldsymbol{y} = N \cdot \boldsymbol{a},$$

with the $q \times q$ triangular matrix

$$N = \begin{pmatrix} 1 & 0 & 0 & \ldots & 0 \\ 1 & (x_1 - x_0) & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & (x_{q-1} - x_0) & \prod_{j=0}^{1}(x_{q-1} - x_j) & \ldots & \prod_{j=0}^{q-2}(x_{q-1} - x_j) \end{pmatrix}.$$

Using either of these bases takes

$$\mathcal{O}(q^2)$$

field operations in $\mathbb{F}_q$ to recover $a_0, \ldots, a_{q-1}$[GCL92, p.129] and data points (or plaintext-ciphertext pairs).

[5] For the empty product $N_0(X)$ we set $N_0(X) = 1$.

The most efficient approaches to polynomial interpolation rely on the divide-and-conquer strategy of the Fast Fourier Transform (FFT) and, in case of interpolating a data set of N points, lead to a complexity of

$$\mathcal{O}(N \log N)$$

field operations in $\mathbb{F}_q$ [GCL92, p.132]. In a nutshell, the FFT is a fast version of the Discrete Fourier Transform (DFT) given by

$$\begin{aligned} \mathbb{F}_q^N &\to \mathbb{F}_q^N \\ (a_0, \ldots, a_{N-1}) &\mapsto (\hat{a}_0, \ldots, \hat{a}_{N-1}), \end{aligned}$$

such that

$$\hat{a}_j = \sum_{k=0}^{N-1} a_k \phi^{jk}, \quad j = 0, 1, \ldots, N-1,$$

for a primitive N-th root of unity $\phi \in \mathbb{F}_q$. Stated differently, the DFT evaluates the polynomial

$$Q(X) = a_0 + a_1 X + \cdots + a_{N-1} X^{N-1}$$

at the N points $1, \phi, \phi^2, \ldots, \phi^{N-1}$ and returns $Q(1), Q(\phi), \ldots, Q(\phi^{N-1})$. We do not discuss the many varieties of FFT-algorithms here, but refer to [RKH10] for an overview of state-of-the-art algorithms. An FFT-based approach for interpolating Q, i.e., for recovering $a_0, \ldots, a_{N-1}$, on the set of N-th roots of unity $1, \phi, \phi^2, \ldots, \phi^{N-1}$ leads to the following matrix equation

$$\boldsymbol{y} = V_{FFT} \cdot \boldsymbol{a},$$

with $\boldsymbol{y} = (E(1), E(\phi^1), \ldots, E(\phi^{N-1}))^t$, $\boldsymbol{a} = (a_0, \ldots, a_{N-1})^t$ and where V_{FFT} denotes the special Vandermonde matrix

$$V_{FFT} = \begin{pmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & \phi^1 & \phi^2 & \ldots & \phi^{N-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \phi^{N-1} & \phi^{2(N-1)} & \ldots & \phi^{(N-1)(N-1)} \end{pmatrix}.$$

The inverse matrix of V_{FFT} is given by [GCL92, Theorem 4.2, p.130]

$$V_{FFT}^{-1} = \frac{1}{n} \begin{pmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & \phi^{-1} & \phi^{-2} & \ldots & \phi^{-(N-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \phi^{-(N-1)} & \phi^{-2(N-1)} & \ldots & \phi^{-(N-1)(N-1)} \end{pmatrix}.$$

Hence, recovering $(a_0, \ldots, a_{N-1})$ is a matter of applying the FFT with ϕ^{-1} and a subsequent scaling with the factor $1/n$.

So far we have discussed *univariate* interpolation, i.e., recovering the univariate polynomial representation of $E : \mathbb{F}_q \longrightarrow \mathbb{F}_q$ in

$$\mathbb{F}_q[X] / \langle X^q - X \rangle.$$

However, if $E : \mathbb{F}_q^t \longrightarrow \mathbb{F}_q^t$ is a *multivariate* or *t-variate* function, we obtain a multivariate polynomial representation as a t-tuple of elements in

$$\mathbb{F}_q[X_1, \ldots, X_t] / \langle X_1^q - X_1, \ldots, X_t^q - X_t \rangle.$$

For the special case $q = 2$, we speak of *(vectorial) Boolean functions* and the corresponding multivariate polynomial representation of $E : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$ as a t-tuple of elements in

$$\mathbb{F}_2[X_1, \ldots, X_t]/\langle X_1^2 - X_1, \ldots, X_t^2 - X_t \rangle$$

is called *Algebraic Normal Form* (ANF) of E. We discuss the different polynomial representations of (vectorial) Boolean functions and their use in cryptanalysis in more detail in Chapter 8.

4.2 Higher-Order Differential Distinguishers

Higher-order differential distinguishers have been introduced in [Lai94] and are distinguishers of block ciphers over binary extension fields $\mathbb{F}_2^n$ with low algebraic degree. Today, higher-order differential distinguishers are well-established means of cryptanalysis. Some important and recent works in this direction are [MSK98; BCC11; DRS20; EGL+20; CGG+22; BCP23; LAW+23], with recent generalizations in [BCD+20a; GCR+21]. A higher-order differential distinguisher may even allow the cryptanalyst to recover the secret key, as demonstrated in [EGL+20].

It is well-known that almost all permutation polynomials of $\mathbb{F}_q$ have degree $q - 2$ [KP02]. For $q = 2^n$ and in light of $\mathbb{F}_2^n \cong \mathbb{F}_{2^n}$ (as vector spaces), this translates to the statement:

> "Almost all bijective (vectorial) boolean functions $\mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$ have algebraic degree exactly n-1."

Bijective (vectorial) boolean functions cannot have algebraic degree n (see also Footnote 3). Arbitrary (vectorial) boolean functions may have algebraic degree n. Based on the results in [Car10, p.295], [Bak19, Theorem 1], we assert in a more general context:

> "Almost all (vectorial) boolean functions $\mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$ have algebraic degree at least n-1."

The basic idea behind higher-order differential distinguishers is the following. If a substantial number of block cipher instances admit a polynomial representation with algebraic less than $n - 1$, then this property may be used to distinguish this block cipher from a random permutation. This, in turn, violated the pseudo-random permutation claim of the block cipher. In what follows, we discuss how higher-order differential distinguishers exploit this idea.

Let $\Delta_a f$ denote the derivative of a vectorial boolean function $f : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$ at a point $a \in \mathbb{F}_2^n$ defined by

$$(\Delta_a f)(x) := f(x + a) + f(x). \tag{5}$$

For $i > 1$ and $\mathbb{F}_2$-linearly independent[6] points $a_1, \ldots, a_i \in \mathbb{F}_2^n$, the i-th (order) derivative $\Delta_{a_1,\ldots,a_i} f$ of f is defined recursively as

$$\Delta_{a_1,\ldots,a_i} f = \Delta_{a_i}(\Delta_{a_1,\ldots,a_{i-1}} f).$$

The (first-order) derivative in Eq. (5) is also called a *differential*. The generalization to higher-order derivatives, thus, lends its name to higher-order differential distinguishers. Similar to ordinary calculus over the real numbers, the derivative $\Delta_a f$ satisfies [Lai94, p.3]

[6] The concept of i-th order derivatives can also be defined for linearly *dependent* points $a_1, \ldots, a_i$. But then $\Delta_{a_1,\ldots,a_i} f = 0$, which is of no interest.

1. $\Delta_a(f+g) = \Delta_a f + \Delta_a g$ ("homomorphic with respect to addition").
2. For all $x \in \mathbb{F}_2^n$ it holds $\Delta_a(f \cdot g)(x) = (\Delta_a f)(x) \cdot g(x) + f(x+a) \cdot (\Delta_a g)(x)$ ("almost Leibnitz").

The next two lemmata are crucial for the application of higher-order derivatives in algebraic cryptanalysis.

Lemma 1 ([Lai94], Proposition 2). *Let $f : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$ be a vectorial boolean function and $a \in \mathbb{F}_2^n$. Then*

$$\deg(\Delta_a f) \leq \deg(f) - 1.$$

Lemma 2 ([Lai94], Proposition 3). *Let $f : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$ be a vectorial boolean function, let $a_1, \ldots, a_i \in \mathbb{F}_2^n$ be $\mathbb{F}_2$-linearly independent and let*

$$S_i := S_i(a_1, \ldots, a_i) := \{\lambda_1 a_1 + \cdots + \lambda_i a_i : \lambda_j \in \mathbb{F}_2\}$$

be the set of all 2^i linear combinations of $a_1, \ldots, a_i$. Then it holds for all $x \in \mathbb{F}_2^n$

$$(\Delta_{a_1,\ldots,a_i} f)(x) = \sum_{s \in S_i} f(x+s).$$

As immediate consequences of Lemma 1 and Lemma 2, respectively, we formulate two corollaries.

Corollary 2. *Let $f : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$ be a vectorial boolean function and let $a_1, \ldots, a_i \in \mathbb{F}_2^n$ be $\mathbb{F}_2$-linearly independent. Then*

$$\deg(\Delta_{a_1,\ldots,a_i} f) \leq \deg(f) - i.$$

Corollary 3. *Let $f : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$ be a vectorial boolean function with algebraic degree δ and let $a_1, \ldots, a_{\delta+1} \in \mathbb{F}_2^n$ be a set of $\delta+1$ $\mathbb{F}_2$-linearly independent elements. Then*

$$\Delta_{a_1,\ldots,a_{\delta+1}} f = 0.$$

In particular,

$$(\Delta_{a_1,\ldots,a_{\delta+1}} f)(x) = \sum_{s \in S_{\delta+1}} f(x+s) = 0,$$

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

Stated differently, if $f : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$ is a vectorial boolean function with algebraic degree δ, then all the evaluations of f over any affine subspace of $\mathbb{F}_2^n$ of dimension $\delta+1$ sum to zero. I.e., for any $\mathbb{F}_2$-linearly independent elements $a_1, \ldots, a_{\delta+1} \in \mathbb{F}_2^n$ with

$$S_{\delta+1} := \{\lambda_1 a_1 + \cdots + \lambda_{\delta+1} a_{\delta+1} : \lambda_j \in \mathbb{F}_2\},$$

and any $\lambda \in \mathbb{F}_2^n$ we have

$$\sum_{s \in S_{\delta+1}} f(\lambda + s) = 0.$$

If many, or all, instances of a block cipher $\mathcal{E}(\mathbb{F}_2^n, \mathcal{K})$ have algebraic degree strictly less than $n-1$, a higher-order differential distinguisher A may exploit this property in the following way. For an oracle $\mathcal{O}_{E_k}$ of a randomly chosen instance $E_k \in \mathcal{E}$ and any vector subspace $S \subseteq \mathbb{F}_2^n$ of dimension $\delta+1$, the distinguisher A requests the ciphertexts $\{E_k(s) : s \in S\}$ from $\mathcal{O}_{E_k}$ and checks if

$$\sum_{s \in S} E_k(s) = 0.$$

For said block cipher instances, above sum is zero. With overwhelming probability (see [KP02]), the same property does not hold for the ciphertexts $\{\pi(s) : s \in S\}$ retrieved from an oracle $\mathcal{O}_\pi$, where π is a randomly chosen permutation from the family of all permutations of $\mathbb{F}_2^n$.

In practice, the problem of finding a higher-order differential distinguisher boils down to establishing (tight upper) bounds on the algebraic degree of a block cipher, typically independent of the secret key k. Bounding the algebraic degree is, in itself, a hard problem and an active area of research in symmetric cryptography. We discuss this topic in more detail in Chapter 8.

4.3 Gröbner Basis Analysis

Independent of cryptography, Gröbner basis theory is an area of ongoing research and continues to be actively developed. The use of Gröbner bases in cryptography is due to their application in solving systems of polynomial equations that arise in the security analysis of cryptographic primitives. The authors of [CW09] attribute the first documented use of Gröbner bases in symmetric cryptography to [SK98], as an improvement for the linear cryptanalysis of DES. The adoption of Rijndael as the AES in 2002 [DR02b], together with the simple algebraic structure of Rijndael, motivated a considerable amount of research on algebraic cryptanalysis, and, in particular, on Gröbner basis techniques. Some relevant research works from this time are [CP02; Cou02; AFI+04; CL05; CMR06; BPW06]. The increasing popularity of "algebraic" permutations and hash functions tailored for MPC-, FHE- and ZKP-protocols has led to a renewed interest in Gröbner basis cryptanalysis in the last decade. To the best of our knowledge, our research in [ACG+19] marks the first successful Gröbner basis analysis of a block cipher and hash function. Today, many MPC-, FHE-, or ZKP-friendly primitives also provide arguments against algebraic cryptanalysis, including Gröbner basis analysis. This applies notably to all AFPs discussed in Section 1.2.

We turn our attention to the general procedure of Gröbner basis cryptanalysis. Conceptually, it consists of two steps.

1. Modelling a cryptographic primitive as a system of polynomial equations with unknown parameters as variables. A parameter of interest might be the secret key of a block cipher, a solution to the CICO Problem of a permutation (see also Definition 2), or the preimage of a given hash value.

2. Solving the system of polynomial equations using Gröbner basis techniques.

We note that equation systems stemming from problems in symmetric cryptography most often have a finite number of solutions. Hence, we may assume that the equation system generates a zero-dimensional ideal. In step 2., "solving" may have one of the following meanings: (a) finding exactly one solution, (b) finding all solutions, or (c) finding that no solution exists. The situation where there is no solution might arise in a so-called hybrid form of Gröbner basis analysis [BFP09; BFP12]. The idea is to combine exhaustive search with Gröbner basis computations: we fix some of the variables and, subsequently, solve the remaining equation system. The remaining equation system might be ill-conditioned, and, hence, might not admit any solution. In practice, step 2. is usually a triad of computations.

1. Computing a *degrevlex* Gröbner basis for the zero-dimensional input system using an algorithm such as F4 [Fau99], or F5 [Fau02].
2. Convert the *degrevlex* Gröbner basis to a *lex* Gröbner basis using an algorithm such as FGLM [FGL+93], or a probabilistic [FGH+14] or sparse variant thereof [FM11a; FM17].
3. Factor the univariate polynomial in the *lex* Gröbner basis using any polynomial factoring algorithm. Determine the solutions of the corresponding variable. If needed, back-subsitute any solution into the other equations in the *lex* Gröbner basis to obtain a full solution of the system.

Complexity of Gröbner Basis Computations Runtime complexities for Gröbner basis algorithms are based on the runtime analysis of *matrix-based* algorithms, like `Lazard` (see Algorithm 4) or `MatrixF5` [BFS15]. Since the homogeneous case admits an easier and more elegant treatment, the input polynomials are often assumed to be homogeneous. Any system of polynomials may be homogenized, however, at the cost of adding an extra variable. Most often in (symmetric) cryptography, practically relevant polynomial systems generate a zero-dimensional ideal. Hence, the literature commonly uses the assumption of zero-dimensional ideals to estimate the runtime complexity of Gröbner basis algorithms. We state a first estimate on the runtime complexity of computing a d-Gröbner basis with Algorithm 4.

Theorem 1 ([BFS15], Proposition 1). *Let $F := \{f_1, \ldots, f_m\} \subseteq \mathbb{F}[X_1, \ldots, X_n]$ be a set of m homogeneous polynomials in n variables generating the (not necessarily zero-dimensional) ideal $I := \langle f_1, \ldots, f_m \rangle$. Computing a d–Gröbner basis of I with `Lazard` (see Algorithm 4) requires*

$$\mathcal{O}\left(md \cdot \binom{n+d-1}{d}^{\omega}\right) \tag{6}$$

operations in $\mathbb{F}$, where ω denotes the linear algebra exponent.

For the special case of regular (Definition 23) and semi-regular sequences[7] (Definition 25) the `F5` algorithm performs particularly well, as has been shown in [Fau02, Corollary 3], [Bar04, Théorème 3.2.9 & 3.2.10]. Theorem 2 estimates the runtime of a matrix-based version `MatrixF5` [BFS15] of `F5` under these assumptions. For a more detailed estimate, we refer to the step-by-step analysis of `MatrixF5` in [BFS15, Section 3.2] and [Bar04, Section 3.4].

Theorem 2 ([BFS+05], Proposition 6). *Let $F := \{f_1, \ldots, f_m\} \subseteq \mathbb{F}[X_1, \ldots, X_n]$ be a semi-regular sequence of m polynomials in n variables (i.e., $m \geq n$). Then there is no reduction to 0 in `MatrixF5` for degrees smaller than i_{reg}. Computing a Gröbner basis with `MatrixF5` [BFS15] of the ideal $\langle f_1, \ldots, f_m \rangle$ requires*

$$\mathcal{O}\left(\binom{n+i_{reg}}{i_{reg}}^{\omega}\right) \tag{7}$$

operations in $\mathbb{F}$, where ω denotes the linear algebra exponent.

For zero-dimensional homogeneous ideals, the index of regularity i_{reg} is a definite termination criterion for Algorithm 4 such that the resulting

[7] Semi-regular sequences are a generalization of regular sequences to overdetermined polynomial systems, i.e., to systems with m polynomials in n variables, where $m \geq n$. In particular, semi-regular sequences generate zero-dimensional ideals.

i_{reg}–Gröbner basis is a Gröbner basis (see our discussion in Section 6.2.2 and also Proposition 15). The same remark holds true for `MatrixF5` since it follows the basic strategy of the `Lazard` algorithm. This means, a key parameter for estimating the runtime of step 1. is the index of regularity i_{reg}. In general, determining the index of regularity i_{reg} is a difficult task and requires to compute a Gröbner basis. However, for special classes of polynomial systems there are explicit formulae for i_{reg}. For a regular sequence $f_1, \ldots, f_n \in \mathbb{F}[X_1, \ldots, X_n]$ generating a zero-dimensional ideal[8] the index of regularity is given by the Macaulay bound

$$i_{reg} = 1 + \sum_{i=1}^{n} (d_i - 1), \tag{8}$$

where $d_i = \deg(f_i)$. For semi-regular sequences $f_1, \ldots, f_m \in \mathbb{F}[X_1, \ldots, X_n]$, with $m \geq n$, the index of regularity is the index of the first non-positive coefficient in the Hilbert series

$$\mathrm{HS}_{R/I}(T) = \frac{\prod_{i=1}^{m}(1 - T^{d_i})}{(1-T)^n}.$$

Complexity of FGLM The runtime complexity of the FGLM algorithm [FGL+93] is

$$\mathcal{O}\left(n \cdot d_I^3\right), \tag{9}$$

operations in $\mathbb{F}$, where n is the number of variables in $\mathbb{F}[X_1, \ldots, X_n]$ and $d_I := \dim_{\mathbb{F}}(R/I)$ is the dimension of the quotient ring R/I as $\mathbb{F}$-vector space. For systems fulfilling the premises of the Shape Lemma (Theorem 13), d_I is equal to the degree of the unique univariate polynomial g in the reduced *lex* Gröbner basis [KR00, Theorem 3.7.25]. Moreover, under these premises, d_I is also equal to the number of solutions of I in the algebraic closure of $\mathbb{F}$ [KR00, Theorem 3.7.19]. To summarize, under the premises of the Shape Lemma, we have

$$d_I = |V(I)| = \deg(g).$$

There exists also a sparse variant of the FGLM algorithm [FM11b]. It has a runtime complexity of

$$\mathcal{O}\left(d_I(N_1 + n \log d_I)\right), \tag{10}$$

where N_1 is the number of nonzero entries of a multiplication matrix, which is sparse even if the input system spanning I is dense. For ideals satisfying the premises of the Shape Lemma (Theorem 13), the expected time of converting a *degrevlex* Gröbner basis to a *lex* one using a probabilistic algorithm is [FGH+14]

$$\mathcal{O}\left(\log d_I (d_I^{\omega} + n d_I \log(d_I) \log(\log d_I))\right)$$

arithmetic operations in $\mathbb{F}$, where ω is the linear algebra constant $2 \leq \omega \leq 3$. For generic sequences (see [FGH+14, Definition 5]), the complexity of converting a *degrevlex* Gröbner basis to a *lex* one is

$$\mathcal{O}\left(\sqrt{\frac{6}{n\pi}} d_I^{2+\frac{n-1}{n}}\right) \tag{11}$$

arithmetic operations in $\mathbb{F}$. Thus, the key parameter for estimating the runtime of step 2. is d_I.

[8] This amounts to the square case of $m = n$ equations.

Complexity of Factoring Polynomials Polynomial factorization is a classical problem and for this purpose we may choose one of many factoring algorithms [Ber71; CZ81; KS98; Gen07; KU11]. See also [Vas07, Section 6.7] for a summary of classical factorization algorithms. For example, a fast version [KS98] of the (probabilistic) Cantor-Zassenhaus algorithm [CZ81] for factoring a univariate polynomial of degree D over $\mathbb{F}_q$ uses an expected number of

$$\mathcal{O}\left(D^{1.815}\right) \tag{12}$$

operations in $\mathbb{F}_q$. In step 3., we factor the (unique) univariate polynomial in the (minimal) *lex* Gröbner basis. Let us call this polynomial f. Hence, the key parameter to estimate the runtime of this step is $\deg(f)$. We may reduce the cost of step 3. by performing steps 1. and 2. twice, for two different boundary constraints (e.g., for two different plaintext-ciphertext pairs). If X denotes the last *lex* variable of the input system, then we generate two univariate polynomials, say $f_1(X)$ and $f_2(X)$. Let us assume X represents the unknown parameter u. Then $f_1(u) = 0 = f_2(u)$, or in other words, u is a root of, both, f_1 and f_2. Hence

$$(X - u) \mid f_1 \text{ and } (X - u) \mid f_2.$$

As a result, $(X - u) \mid \gcd(f_1, f_2)$. We hope, the cost of computing steps 1. and 2. twice, together with the subsequent factorization of $\gcd(f_1, f_2)$, is lower than directly factoring f. Computing the GCD of two polynomials over $\mathbb{F}_q$ of degree at most D requires

$$\mathcal{O}\left(D(\log D)^2\right) \tag{13}$$

operations in $\mathbb{F}_q$. See [AHU74, Theorem 7.4] for the $\mathcal{O}(D \log D)$ complexity of FFT-based polynomial multiplication and [AHU74, Theorem 8.18] for the $\mathcal{O}(M(D) \log D)$ complexity of computing polynomial GCDs. Here, $M(D)$ denotes the number of operations required to multiply two polynomials of degree at most D.

Summary Let $I := \langle f_1, \ldots, f_m \rangle \subseteq R$ be a homogeneous zero-dimensional ideal. We provide a quick reference for the key parameters determining the runtime complexities of steps 1., 2., and 3..

1. Number of variables n and index of regularity i_{reg}. The number of polynomials m is, implicitly, considered by i_{reg}.
2. Number of variables n and dimension $d_I = \dim_{\mathbb{F}}(R/I)$ of R/I as $\mathbb{F}$-vector space.
3. Degree D of the unique univariate polynomial in a minimal *lex* Gröbner basis of I.

Part II

BACKGROUND

5.1 Notation and Basic Definitions

In this section we introduce our notation and provide a streamlined but complete account of relevant definitions for the rest of Part II. Owing to a concise presentation, we choose to present many definitions in a colloquial manner and only use a formal definition environment to highlight the most important definitions.

The set $\mathbb{N}$ denotes the set $\{0, 1, 2, \ldots\}$ of all natural numbers (we include 0). In the following, $\mathbb{F}$ denotes a field and $\overline{\mathbb{F}}$ its algebraic closure. We write $\mathbb{F}_q$ for the finite field with q elements. Central for us is the polynomial ring $\mathbb{F}[X_1, \ldots, X_n]$ with n indeterminates (or variables) $X_1, \ldots, X_n$, $n \in \mathbb{N} \setminus \{0\}$, which is why we occasionally use the shorthand notation

$$\boldsymbol{X}^a := X_1^{a_1} \cdots X_n^{a_n},$$

for $a := (a_1, \ldots, a_n) \in \mathbb{N}^n$, and in particular $\boldsymbol{X} := X_1 \cdots X_n$, to write down a product of variables in $\mathbb{F}[X_1, \ldots, X_n]$. Whenever convenient, we succinctly write

$$R := \mathbb{F}[X_1, \ldots, X_n]$$

for the polynomial ring itself. The set R^m of m-tuples of polynomials in R is a finitely generated free R-module with basis $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_m\}$. The element

$$\boldsymbol{e}_i := (0, \ldots, 0, 1, 0, \ldots, 0) \in R^m$$

is called the *i-th canonical basis vector of* R^m which has the 1 in position i. We use bold-face letters $\boldsymbol{f}$ to denote module elements in R^m. Any module element $\boldsymbol{h} = (h_1, \ldots, h_m) \in R^m$ can be uniquely written in the form

$$\boldsymbol{h} = (h_1, \ldots, h_m) = \sum_{i=1}^{m} h_i \cdot \boldsymbol{e}_i.$$

In the context of polynomial rings, for us, a *term* is an expression of the form $\boldsymbol{X}^a$, whereas a *monomial* is a product of a constant and a term $c_a \cdot \boldsymbol{X}^a$, with $c_a \in \mathbb{F}$. We denote the set of all terms in $\mathbb{F}[X_1, \ldots, X_n]$ with T, i.e.,

$$T := \{\boldsymbol{X}^a \in R : a \in \mathbb{N}^n\}.$$

For a polynomial

$$f = \sum_{a \in \mathbb{N}^n} c_a \cdot \boldsymbol{X}^a \in R,$$

we write $T(f) := \{\boldsymbol{X}^a : c_a \neq 0\}$ to denominate the *set of terms in* f. Some authors call this set also the *support of* f. For a set of polynomials $F \subseteq \mathbb{F}[X_1, \ldots, X_n]$ we call

$$\langle F \rangle := \left\{ \sum_{i=1}^{l} h_i \cdot f_i : h_i \in \mathbb{F}[X_1, \ldots, X_n], f_i \in F, l \in \mathbb{N} \right\}$$

the *ideal generated by* F. Note, $\langle F \rangle$ is indeed an ideal of R.

Definition 8. A *term order* is a relation $\leq$ on the set of terms T which satisfies the following properties:

1. $\leq$ is a total order on T.
2. $\forall t \in T : 1 \leq t$.
3. $\forall r, s, t \in T : r \leq s \Rightarrow r \cdot t \leq s \cdot t$.

It is also possible to define term orders on the set $\mathbb{N}^n$ since any term $X_1^{a_1} \cdots X_n^{a_n}$ can be identified with the exponent vector $(a_1, \ldots, a_n)$. Multiplication of terms corresponds to addition of exponent vectors. In other words, there is an order-preserving isomorphism (of abelian monoids) $\pi : (T, \cdot) \to (\mathbb{N}^n, +)$

$$\pi : X_1^{a_1} \cdots X_n^{a_n} \mapsto (a_1, \ldots, a_n),$$

which allows us to define term orders on $\mathbb{N}^n$ and carry it over to T via π^{-1}. An equivalent definition would then read as follows:

1. $\leq$ is a total order on $\mathbb{N}^n$.
2. $\forall a \in \mathbb{N}^n : 0 \leq a$.
3. $\forall r, s, t \in \mathbb{N}^n : r \leq s \Rightarrow r + t \leq s + t$.

For a total order $\leq$ on T (or $\mathbb{N}^n$), properties 2. and 3. in above definitions are equivalent to $\leq$ being a well-order on T (or $\mathbb{N}^n$). In essence, this follows from the proof of Dickson's Lemmas, see for example [CLO15, p.73, Theorem 5].

Definition 9. A term order $\leq$ is called *graded* if terms of different degrees are ordered according to their degrees, i.e., if for $s, t \in T$, with $\deg(s) \neq \deg(t)$, it holds

$$s < t \iff \deg(s) < \deg(t).$$

We summarize some of the most important (strict) term orders in Definition 10.

Definition 10. Let $\boldsymbol{X}^a = X_1^{a_1} \cdots X_n^{a_n}$ and $\boldsymbol{X}^b = X_1^{b_1} \cdots X_n^{b_n}$ be two terms in T. We define ...

1. ... the *lexicographic order* $<_{deglex}$ as

$$\boldsymbol{X}^a <_{lex} \boldsymbol{X}^b :\iff \text{rightmost nonzero entry of the vector difference } b - a \text{ is positive.}$$

2. ... the *degree lexicographic order* $<_{glex}$ as

$$\boldsymbol{X}^a <_{deglex} \boldsymbol{X}^b :\iff \deg(\boldsymbol{X}^a) < \deg(\boldsymbol{X}^b) \text{ or } \deg(\boldsymbol{X}^a) = \deg(\boldsymbol{X}^b) \text{ and } \boldsymbol{X}^a <_{lex} \boldsymbol{X}^b.$$

3. ... the *degree reverse lexicographic order* $<_{degrevlex}$ as

$$\boldsymbol{X}^a <_{degrevlex} \boldsymbol{X}^b :\iff \deg(\boldsymbol{X}^a) < \deg(\boldsymbol{X}^b) \text{ or } \deg(\boldsymbol{X}^a) = \deg(\boldsymbol{X}^b) \text{ and leftmost nonzero entry of the vector difference } b - a \text{ is negative.}$$

For each (strict) term order $<$, the associated (non-strict) term order $\leq$ is given by the reflexive closure of $<$, i.e., by the relation

$$\{(s,t) : s,t \in T,\ s < t\} \cup \{(t,t) : t \in T\}.$$

For a given term order $\leq$ on T, the *leading term* $\mathrm{LT}(f)$ of a non-zero polynomial $f = \sum_{t \in T(f)} c_t \cdot \in R$ is defined as

$$\mathrm{LT}(f) \coloneqq \max_{\leq} T(f),$$

and the *leading coefficient* $\mathrm{LC}(f)$ is given by the associated coefficient of $\mathrm{LT}(f)$. The *leading monomial* $\mathrm{LM}(f)$ of f is the product $\mathrm{LC}(f) \cdot \mathrm{LT}(f)$. We remark, for the zero polynomial 0 leading term, leading monomial and leading coefficient are not defined. For $f = \sum_{t \in T(f)} c_t \cdot t \in R$, we use the notation

$$C_t(f) \coloneqq \begin{cases} c_t & \text{if } t \in T(f), \\ 0 & \text{otherwise.} \end{cases}$$

to denote the coefficient of a given term t in f. In later sections we often work with the "decapitated" part of a non-zero polynomial $f \in R$, i.e., the part that remains when the leading monomial is subtracted. This is why we call the polynomial

$$\mathrm{Tail}(f) \coloneqq f - \mathrm{LC}(f) \cdot \mathrm{LT}(f) = f - \mathrm{LM}(f)$$

the *tail of* f. For $F \subseteq R$, the set

$$T(F) \coloneqq \{T(f) : f \in F\}$$

is the *set of terms in* F and

$$\mathrm{LT}(F) \coloneqq \{\mathrm{LT}(F) : f \in F, f \neq 0\}$$

is the *set of leading terms in* F. We say a term u *divides* a term t, if t is a term multiple of u. In other words

$$u \mid t :\Longleftrightarrow \exists v \in T : u \cdot v = t.$$

If $u \neq 1$ and $u \neq t$ we call u a *proper divisor* of t and say u divides t properly or t is a *proper multiple* of u. Otherwise we call u a *trivial divisor* of t.

Definition 11. Let $G \subseteq R \setminus \{0\}$, $g \in R \setminus \{0\}$ and $f, h \in R$. We say ...

1. ... *f reduces to h modulo g* if there exists a term $t \in T(f)$ such that $\mathrm{LT}(g) \mid t$ and $h = f - C_t(f) \cdot \mathrm{LC}(g)^{-1} \cdot ug$, where $u \coloneqq t/\mathrm{LT}(g)$. In symbols, we write
$$f \longrightarrow_g h.$$

2. ... *f reduces to h modulo G in one step*, if there exists an element $g \in G$ such that $f \longrightarrow_g h$. In symbols, we write $f \longrightarrow_G h$.

3. ... *f reduces to h modulo G in a finite number of steps* if
$$f \longrightarrow_G h_1 \longrightarrow_G h_2 \longrightarrow_G \cdots \longrightarrow_G h_k = h,$$
for some $k \in \mathbb{N}$ and $h_1, \ldots, h_k \in R$. In symbols, we write $f \xrightarrow{*}_G h$.

4. ... *f is reducible by g* or *g reduces f* if there exists a term $t \in T(f)$ such that $\mathrm{LT}(g) \mid t$.

5. ... f *is reducible modulo* G or f *is* G*-reducible* if there exists an element $g \in G$ and term $t \in T(f)$ such that $\mathrm{LT}(g) \mid t$.

If g reduces f and $u := t/\mathrm{LT}(g)$, the element ug is called a *reductor of* f. For the sake of a simple notation, we call any scalar multiple $c \cdot ug$, $c \in \mathbb{F}$, a reductor as well. If $t = \mathrm{LT}(f)$, we call the reduction step a *top*-reduction, otherwise a *tail-reduction* and the corresponding reductors are called *top-reductor* and *tail-reductor*, respectively. If a polynomial is not reducible (or tail-reducible) modulo G, it is called *irreducible* (or *tail-irreducible*) *modulo* G.

Definition 12. Let $f, f' \in \mathbb{F}[X_1, \ldots, X_n]$ and let $G \subseteq \mathbb{F}[X_1, \ldots, X_n] \setminus \{0\}$ be a set of non-zero polynomials. We call f' a *normal form of* f *modulo* G or a G*-normal form of* f if $f \xrightarrow{*}_G f'$ and f' is irreducible modulo G. We write

$$f \bmod G := \{f' \in P : f' \text{ a normal form of } f \text{ modulo } G\}$$

for the set of all normal forms of f modulo G.

Colloquially, we often speak of "reducing f modulo G". In a more formal way, this means we are computing a normal $f' \in f \bmod G$ through a process outlined in Algorithm 1. We note, every polynomial has at least one normal form modulo G, see Theorem 5. The zero polynomial 0 is irreducible modulo any set G (since $T(0) = \emptyset$) and thus the unique normal form of 0 is 0 itself.

5.2 A Pinch of Commutative Algebra

Lemma 3 ([CLO15], p.70, Lemma 2)**.** *Let $A \subseteq T$ be a set of terms. For a term $t \in T$ it holds $t \in \langle A \rangle$ if and only if $a \mid t$, for some $a \in A$.*

Lemma 4 (Dickson Lemma, [CLO15], p.52, Theorem 5)**.** *Let $I := \langle A \rangle$ be a monomial ideal for some $A \subseteq T$. Then I is finitely generated, i.e., there exist $a_1, \ldots, a_k \in A$ such that $I = \langle a_1, \ldots, a_k \rangle$.*

Theorem 3 (Hilbert Basis Theorem, [CLO15], p.77, Theorem 4)**.** *If a ring S is noetherian, then $S[X_1, \ldots, X_n]$ is noetherian. In particular, $\mathbb{F}[X_1, \ldots, X_n]$ is noetherian. In other words, every ideal $I \subseteq \mathbb{F}[X_1, \ldots, X_n]$ is finitely generated.*

Theorem 4 (Macaulay Basis Theorem, [KRoo], p.62, Theorem 1.5.7)**.** *Let $R = \mathbb{F}[X_1, \ldots, X_n]$ and let $I \subseteq R$ be an ideal. Then the set*

$$B := \{t \in T : t \notin \langle \mathrm{LT}(I) \rangle\}.$$

is an $\mathbb{F}$-vector space basis for the quotient ring R/I.

Theorem 5 (Division with Remainder, [CLO15], p.64, Theorem 3)**.** *Let $\leq$ be a term order on T, and let $G := \{g_1, \ldots, g_m\} \subseteq R \setminus \{0\}$ be a set of non-zero polynomials. Then every $f \in R$ can be written as*

$$f = q_1 g_1 + \cdots q_m g_m + r,$$

where $q_i, r \in R$, and either $r = 0$ or no term in r is divisible by any element of $\mathrm{LT}(g_1), \ldots, \mathrm{LT}(g_m)$. In particular, f has at least one normal form modulo G, i.e., $f \bmod G \neq \emptyset$

Algorithm 1 depicts a basic procedure for computing normal forms in $\mathbb{F}[X_1, \ldots, X_n]$ and is also known as *division with remainder* or *multivariate polynomial division* in $\mathbb{F}[X_1, \ldots, X_n]$. Often, the textbook division-with-remainder-algorithm presumes a fixed order when processing polynomials in F, but for

Algorithm 1: Division with remainder in $\mathbb{F}[X_1, \ldots, X_n]$

Input: Set of non-zero polynomials $F := \{f_1, \ldots, f_m\} \in R$, a non-zero polynomial $p \in R$ and a term order $\leq$

Result: A normal form $p' \in p \bmod F$ and $q_1, \ldots, q_m \in R$ with $p = q_1 f_1 + \cdots q_m f_m + p'$

```
p' ← p
q_1 ← 0, ..., q_m ← 0
while ∃t ∈ T(p') ∃1 ≤ i ≤ m : LT(f_i) | t do
    Select such t
    Select such f_i
    u ← LM(p') / LM(f_i)
    p' ← p' − u · f_i
    q_i ← q_i + u
return p', q_1, ..., q_m
```

proving termination and correctness this order is irrelevant. A particularity of Algorithm 1 is: in general, the resulting normal form is not unique and depends on the choices made if there exists more than one reductor for a given term. However, if the set F is a Gröbner basis of $\langle F \rangle$ (see Definition 13), the resulting normal form is indeed unique. In fact, a unique normal form is a characterizing property of Gröbner bases. Theorem 6 summarizes this fact.

Theorem 6 ([KRoo], p.111, Theorem 2.4.1)**.** *Let $G := \{f_1, \ldots, f_k\} \subseteq R$ and $I := \langle f_1, \ldots, f_k \rangle$. The following properties are equivalent:*

1. $\langle \mathrm{LT}(I) \rangle = \langle \mathrm{LT}(f_1), \ldots, \mathrm{LT}(f_k) \rangle$.
2. *Every $f \in \mathbb{F}[X_1, \ldots, X_n]$ has a unique normal form modulo G. In other words, $f \bmod G$ only contains a single element.*

5.2.1 Basic Gröbner Basis Theory

Definition 13. Let $\leq$ be a term order on T and $I \subseteq R$ an ideal. A set $G := \{f_1, \ldots, f_m\} \subseteq R$ is called a *Gröbner basis of I with respect to $\leq$* if

$$\langle \mathrm{LT}(f_1), \ldots, \mathrm{LT}(f_m) \rangle = \langle \mathrm{LT}(I) \rangle.$$

Every Gröbner basis G assumes a certain term order and is to be understood with respect to the ideal $I := \langle G \rangle$. Sometimes it is more convenient to be sloppy and omit this term order, or the ideal $\langle G \rangle$, when we speak about Gröbner bases. Thus, whenever we make a statement about a Gröbner basis G without mentioning a concrete term order or ideal, the statement holds true for any term order and is meant with respect to the ideal $\langle G \rangle$.

In Definition 13, the inclusion $\langle \mathrm{LT}(f_1), \ldots, \mathrm{LT}(f_m) \rangle \subseteq \langle \mathrm{LT}(I) \rangle$ is always true. The non-trivial part is the reverse inclusion

$$\langle \mathrm{LT}(I) \rangle \subseteq \langle \mathrm{LT}(f_1), \ldots, \mathrm{LT}(f_m) \rangle,$$

which is equivalent to

$$\mathrm{LT}(I) \subseteq \langle \mathrm{LT}(f_1), \ldots, \mathrm{LT}(f_m) \rangle.$$

Using Lemma 3, we see that the last inclusion is, in turn, equivalent to

$$\forall t \in \mathrm{LT}(I)\, \exists 1 \leq i \leq m : \mathrm{LT}(f_i) \mid t. \tag{14}$$

This means, we could define Gröbner bases also using the requirement in (14). In another way of speaking, the requirement in (14) says that every element in the ideal I is top-reducible modulo $G = \{f_1, \ldots, f_m\}$.

Definition 14. A Gröbner basis $G \subseteq R$ is called *reduced* if for every element $g \in G$ it holds $\mathrm{LC}(g) = 1$ and g is irreducible modulo $G \setminus \{g\}$.

Definition 15. Let $f, g \in R \setminus \{0\}$ be two non-zero polynomials. We call the polynomial

$$S(f,g) := \frac{\mathrm{lcm}(\mathrm{LT}(f), \mathrm{LT}(g))}{\mathrm{LM}(f)} \cdot f - \frac{\mathrm{lcm}(\mathrm{LT}(f), \mathrm{LT}(g))}{\mathrm{LM}(g)} \cdot g$$

the *S-polynomial of f and g*.

Theorem 7 ([CLO15], p.78, Corollary 6)**.** *For every term order $\leq$ and every ideal $I \subseteq R$, there exists a Gröbner basis of I with respect to $\leq$. Furthermore, if G is a Gröbner basis of I, then $I = \langle G \rangle$.*

It is natural to ask whether an ideal has more than one Gröbner basis (for a given term order). In general, the answer to this question is yes. However, the next theorem says that a reduced Gröbner basis of a given ideal is unique (for a fixed term order).

Theorem 8 ([CLO15], p.93, Theorem 5)**.** *Let $G, G' \subseteq R$ be reduced Gröbner bases with $\langle G \rangle = \langle G' \rangle$, then $G = G'$.*

The following proposition is just another view on the fact that non-reduced Gröbner bases might contain redundant generators.

Proposition 3 ([CLO15], p.92, Lemma 3)**.** *Let G be a Gröbner basis of an ideal $I \subseteq R$ and $f, g \in G$, $f \neq g$. If $\mathrm{LT}(f) \mid \mathrm{LT}(g)$, then $G \setminus \{g\}$ is a Gröbner basis as well.*

Theorem 9 (Buchberger Criterion, [CLO15], p.86, Theorem 6)**.** *Let $I \subseteq R$ be an ideal. A subset $G := \{g_1, \ldots, g_m\} \subseteq I$ is a Gröbner basis of I if and only if for all S-polynomials $S(g_i, g_j)$, $1 \leq i < j \leq m$, it holds $S(g_i, g_j) \xrightarrow{*}_G 0$.*

The criteron in Theorem 9 suggests a first procedure for computing Gröbner bases: starting with the set $G = \{f_1, ..., f_m\}$, the procedure calculates all possible S-polynomials of elements in G, reduces them modulo G, and adds any non-zero remainders to G. This is repeated by recalculating the new S-polynomials, reducing them modulo G, and, again, adding any non-zero remainder to G. Essentially, above procedure describes the `Buchberger` algorithm (see Algorithm 2) for which a proof of termination was first provided by Bruno Buchberger in [Buc65] (see [Buc06] for a translation to English). We present a more detailed discussion of the `Buchberger` algorithm in Section 6.1.

An important notion in the context of Gröbner basis computations are syzygies. Simply puts, syzygies describe relations of polynomials $f_1, \ldots, f_m$ that yield the zero-polynomial, i.e., relations of the form

$$h_1 f_1 + \cdots h_m f_m = 0,$$

for certain $h_1, \ldots, h_m \in R$. Syzygies play an important role when it comes to detecting reductions to zero during a Gröbner basis computation (see the discussion after Theorem 10). We describe criteria that use information coming from syzygies for detecting reductions to zero in more detail in

Section 6.5. In what follows, we introduce syzygies in a formal way. For an ordered set of polynomials $F := (f_1, \dots, f_m) \in R^m$ we call the map

$$\nu_F : R^m \to R, (h_1, \dots, h_m) \mapsto \sum_{i=1}^{m} h_i \cdot f_i$$

the *canonical R-module homomorphism with respect to F* or the *(canonical) evaluation homomorphism on R^m with respect to F*. If the set F is known from the context, we allow ourselves to omit the with-respect-to-F part and simply write ν instead of ν_F.

Definition 16. For an m-tuple $F := (f_1, \dots, f_m) \in R^m$, the kernel of the evaluation homomorphism ν_F is called the *syzygy module of F* and is denoted by $\mathrm{Syz}(f_1, \dots, f_m)$ or $\mathrm{Syz}(F)$. In symbols, we write

$$\mathrm{Syz}(F) := \mathrm{Syz}(f_1, \dots, f_m) := \ker(\nu_F) = \left\{ (h_1, \dots, h_m) \in R^m : \sum_{i=1}^{m} h_i f_i = 0 \right\}.$$

An element $(h_1, \dots, h_m)$ of $\mathrm{Syz}(f_1, \dots, f_m)$ is called a *syzygy of* $(f_1, \dots, f_m)$.

For any $i, j \in \{1, \dots, m\}$, the syzygy $h_j \cdot \boldsymbol{e}_i - h_i \cdot \boldsymbol{e}_j$ is called a *trivial syzygy* or *Koszul syzygy* or *principal syzygy*. A syzygy $(h_1, \dots, h_m)$ of $F = (f_1, \dots, f_m)$ is called *homogeneous* if for all $i, j \in \{1, \dots, m\}$ with $h_i, h_j \neq 0$ it holds

$$\mathrm{LT}(f_i h_i) = \mathrm{LT}(f_j h_j).$$

The module $\mathrm{Syz}(F)$ is, indeed, a R-module. In particular, it is a submodule of R^m and, as such, it is a finitely generated R-module. A generating set $\boldsymbol{S} \subseteq \mathrm{Syz}(F)$ is called *homogeneous* if every $\boldsymbol{s} \in \boldsymbol{S}$ is homogeneous. The elements in $\boldsymbol{S}$ need not necessarily be homogeneous with respect to the same leading term.

Proposition 4 ([CLO15], p.111, Proposition 5)**.** *Let $R = \mathbb{F}[X_1, \dots, X_n]$ and let $(f_1, \dots, f_m)$ be an m-tuple of polynomials in R. The module*

$$Syz(\mathrm{LT}(f_1), \dots, \mathrm{LT}(f_m))$$

is generated (as R-module) by the set

$$\{\boldsymbol{s}_{i,j} : 1 \leq i < j \leq m\},$$

where

$$\boldsymbol{s}_{i,j} := \frac{\mathrm{lcm}(\mathrm{LT}(f_i), \mathrm{LT}(f_j))}{\mathrm{LM}(f_i)} \cdot \boldsymbol{e}_i - \frac{\mathrm{lcm}(\mathrm{LT}(f_i), \mathrm{LT}(f_j))}{\mathrm{LM}(f_j)} \cdot \boldsymbol{e}_j.$$

In other words, for every syzygy $(h_1, \dots, h_m) \in Syz(\mathrm{LT}(f_1), \dots, \mathrm{LT}(f_m))$ there exist $h_{i,j} \in R$ such that it can be written in the form

$$(h_1, \dots, h_m) = \sum_{1 \leq i < j \leq m} h_{i,j} \cdot \boldsymbol{s}_{i,j}.$$

Moreover, the elements $h_{i,j}$ are unique.

The relevance of Proposition 4 becomes evident when we look at the particular construction of S-polynomials: in fact, the word "S-polynomial" is an abbreviation of "Syzygy-polynomial". Comparing the definition of S-polynomials in Definition 15 and the definition of the $\boldsymbol{s}_{i,j}$ in Proposition 4

shows a very similar construction. This is, of course, no coincidence and for $F := (f_1, \ldots, f_m)$ we have the relation

$$\nu_F(\boldsymbol{s}_{i,j}) = S(f_i, f_j).$$

In light of this relation, we call $\nu_F(\boldsymbol{s}_{i,j})$ the *S-polynomial corresponding to* $\boldsymbol{s}_{i,j}$. Put another way, the set of all S-polynomials of F induces a particular homogeneous generating set of $\mathrm{Syz}(\mathrm{LT}(f_1), \ldots, \mathrm{LT}(f_m))$, namely the set of all $\boldsymbol{s}_{i,j}$. The interesting point here is, we can characterize Gröbner bases via any homogeneous generating set of $\mathrm{Syz}(\mathrm{LT}(f_1), \ldots, \mathrm{LT}(f_m))$.

Theorem 10 ([CLO15], p.111, Theorem 6)**.** *A basis $F := \{f_1, \ldots, f_m\} \subseteq R$ for an ideal I is a Gröbner basis if and only if for every element $\boldsymbol{s} = (s_1, \ldots, s_m)$ in a homogeneous generating set of $\mathrm{Syz}(\mathrm{LT}(f_1), \ldots, \mathrm{LT}(f_m))$ it holds $\nu_F(\boldsymbol{s}) = \sum_{i=1}^{m} s_i f_i \xrightarrow{*}_G 0$.*

The order of the basis elements $f_1, \ldots, f_m$ is not important in Theorem 10. Hence, any order of $f_1, \ldots, f_m$ can be chosen in the evaluation homomorphism ν_F if, of course, this is also reflected by the corresponding order of elements in $(s_1, \ldots, s_m)$.

An important observation in the context of Gröbner basis computations is the following. During the execution of the `Buchberger` algorithm it might happen that the reduction of an S-polynomial modulo the current basis G results in the zero polynomial. This is referred to as a *reduction to zero*. Any Gröbner basis algorithm leveraging Theorem 9 (or Theorem 10, respectively) suffers from a particular inefficiency if no remedy is put in place: it might happen that many S-polynomials reduce to zero, meaning, they do not contribute to the final Gröbner basis. Thus, computing a reduction to zero is in some sense "useless".[1] Any reduction to zero increases the computational overhead, hence, it is desirable to avoid them as much as possible. In our next considerations, we discuss possible remedies for these "useless" reductions to zero. Definitely, it were desirable to have concrete criteria for detecting them in advance, i.e., without having to carry out the actual reduction. Luckily, such criteria exist. One of these criteria examines the leading terms of f and g to predict when the S-polynomial $S(f, g)$ reduces to zero. This criterion is also known as *Product Criterion* or *Buchberger's First Criterion*.

Proposition 5 ([CLO15], p.106, Proposition 4)**.** *Let $G \subseteq \mathbb{F}[X_1, \ldots, X_n]$ be a finite set of polynomials. If for all $f, g \in G$ it holds $\gcd(\mathrm{LT}(f), \mathrm{LT}(g)) = 1$, then $S(f, g) \xrightarrow{*}_G 0$.*

In practice, Proposition 5, yields an efficient test if a set is already a Gröbner basis. We emphasize this fact with the following corollary.

Corollary 4. *A set $G \subseteq \mathbb{F}[X_1, \ldots, X_n]$ is a Gröbner basis if for all $f, g \in G$, $f \neq g$ it holds $\gcd(\mathrm{LT}(f), \mathrm{LT}(g)) = 1$.*

There is a further way to decrease the number of redundant reductions of S-polynomials which is theoretically deeper than Buchberger's First Criterion. The generating set $\boldsymbol{S} := \{\boldsymbol{s}_{i,j} : 1 \leq i < j \leq m\}$ for $\mathrm{Syz}(\mathrm{LT}(f_1), \ldots, \mathrm{LT}(f_m))$ is not necessarily linearly independent (as R-module), hence, if a subset $\boldsymbol{U} \subseteq \boldsymbol{S}$ is generating $\mathrm{Syz}(\mathrm{LT}(f_1), \ldots, \mathrm{LT}(f_m))$ as well, we can work with $\boldsymbol{U}$ instead of $\boldsymbol{S}$ during a Gröbner basis computation, see Theorem 10. The following approach in Proposition 6 helps to detect whether a syzygy $\boldsymbol{s} \in \boldsymbol{S}$

[1] A reduction to zero still carries some information but, still, the corresponding S-polynomial does not contribute to the final Gröbner basis.

is redundant, or, more formally, whether $\boldsymbol{s}$ is a R-linear combination of other syzygies. This has the consequence that the corresponding S-polynomial of every such redundant syzygy does not have to be considered during the computation of a Gröbner basis. Proposition 6 formalizes this fact and is also called *Chain Criterion* or *Buchberger's Second Criterion*. We prefer the syzygy-formulation of the Chain Criterion in [CLO15, p.113, Proposition 8] but remark that also other formulations exist, e.g., the one via t-representations in [BW93, p. 223, Proposition 5.70].

Proposition 6 ([CLO15], p.113, Proposition 8)**.** *Let $\{f_1, \ldots, f_m\}$ be a set of polynomials in $\mathbb{F}[X_1, \ldots, X_n]$ and let $\boldsymbol{S} := \{\boldsymbol{s}_{i,j} : 1 \leq i < j \leq m\}$ be the generators of $Syz(\mathrm{LT}(f_1), \ldots, \mathrm{LT}(f_m))$ from Proposition 4. If there exist $1 \leq u < v < w \leq m$, such that*

1. $\mathrm{LT}(f_w) \mid \mathrm{lcm}(\mathrm{LT}(f_u), \mathrm{LT}(f_v))$,
2. $\boldsymbol{s}_{u,w}, \boldsymbol{s}_{v,w} \in \boldsymbol{S}$,

then $\boldsymbol{S} \setminus \{\boldsymbol{s}_{u,v}\}$ is also a generating set of $Syz(\mathrm{LT}(f_1), \ldots, \mathrm{LT}(f_m))$.

Proposition 7 summarizes the different characterizing properties of Gröbner basis discussed in this section.

Proposition 7. *Let $I \subseteq R$ be an ideal and let $G := \{g_1, \ldots, g_m\}$ be a set of polynomials in R. The following properties are equivalent:*

1. *G is a Gröbner basis of I, i.e., $\langle \mathrm{LT}(g_1), \ldots, \mathrm{LT}(g_m) \rangle = \langle \mathrm{LT}(I) \rangle$.*
2. *Every $f \in \mathbb{F}[X_1, \ldots, X_n]$ has a unique normal form modulo G.*
3. *All S-polynomials $S(g_i, g_j)$, $1 \leq i < j \leq m$, reduce to zero modulo G.*
4. *For every syzygy $\boldsymbol{s} = (s_1, \ldots, s_m)$ in a homogeneous generating set of $Syz(\mathrm{LT}(g_1), \ldots, \mathrm{LT}(g_m))$ the corresponding polynomial $\nu_G(\boldsymbol{s}) = \sum_{i=1}^{m} s_i g_i$ reduces to zero modulo G.*

5.2.2 Algebraic Varieties

Definition 17. Let $F \subseteq \mathbb{F}[X_1, \ldots, X_n]$ be a set of polynomials. The set of common zeroes (or the zero locus) in the algebraic closure $\overline{\mathbb{F}}$ of polynomials in F

$$V(F) := \{(a_1, \ldots, a_n) \in \overline{\mathbb{F}}^n : f(a_1, \ldots, a_n) = 0, \text{ for all } f \in F\}$$

is called the *variety of F*. If only zeroes in $\mathbb{F}$ are of interest, then we call

$$V_{\mathbb{F}}(F) := \{(a_1, \ldots, a_n) \in \mathbb{F}^n : f(a_1, \ldots, a_n) = 0, \text{ for all } f \in F\}$$

the *affine variety of F*. Moreover, if $F = \{f_1, \ldots, f_k\}$ is a finite set, we also write $V(f_1, \ldots, f_k)$ instead of $V(F)$ and $V_{\mathbb{F}}(f_1, \ldots, f_k)$ instead of $V_{\mathbb{F}}(F)$, respectively.

The next proposition justifies why, in the process of solving systems of algebraic equations, it is possible to work with the ideal generated by the polynomials defining the equations rather than the polynomials themselves. This "switch" from the generators of the ideal to the ideal itself lays the basis for solving systems of algebraic equations through Gröbner bases: instead of the original equation system, we may work with any equation system that generates the same ideal.

Proposition 8 ([CLO15], p.31, Proposition 4). *Let $F \subseteq \mathbb{F}[X_1, \ldots, X_n]$ be a set of polynomials. Then:*

1. $V(F) = V(\langle F \rangle)$.
2. *For $\mathbb{F} = \mathbb{F}_q$ and $F' := F \cup \{X_i^q - X_i : 1 \leq i \leq n\}$ it holds $V(F') = V_{\mathbb{F}_q}(F)$.*

Most often in (symmetric) cryptography we are dealing with equation systems having a finite number of solutions. We summarize important properties of such equation systems in Proposition 9.

Proposition 9 (Finiteness Criterion, [KRoo], p.243, Proposition 3.7.1). *Let $\leq$ be a term order on T and $I \subseteq R = \mathbb{F}[X_1, \ldots, X_n]$ be an ideal with zero locus $V(I) \subseteq \overline{\mathbb{F}}^n$. The following conditions are equivalent:*

1. *$V(I)$ is finite.*
2. *The $\mathbb{F}$-vector space R/I is finite-dimensional.*
3. *The set $T \backslash \mathrm{LT}(I)$ is finite.*
4. *For every $i = 1, \ldots, n$, we have $I \cap \mathbb{F}[X_i] \neq \emptyset$.*
5. *For every $i = 1, \ldots, n$, there exists a number $a_i \in \mathbb{N}$ such that $X_i^{a_i} \in \mathrm{LT}(I)$.*

Definition 18. Let $I \subseteq R$ be an ideal. We call I *zero-dimensional* if $|V(I)| < \infty$.

Proposition 10 ([KRoo], p.245, Proposition 3.7.5). *Let $I \subseteq R = \mathbb{F}[X_1, \ldots, X_n]$ be a zero-dimensional ideal with generators $f_1, \ldots, f_k \in R$. Then $|V(I)| \leq \dim_{\mathbb{F}}(R/I)$. In other words, the system of equations*

$$f_1(x_1, \ldots, x_n) = \cdots = f_k(x_1, \ldots, x_n) = 0$$

has at most $\dim_{\mathbb{F}}(R/I)$ solutions in $\overline{\mathbb{F}}^n$.

5.2.3 Gröbner Bases and Variable Elimination

The Elimination Theorem in Theorem 11 is central for solving systems of polynomial equations using Gröbner bases and, together with the Extension Theorem in Theorem 12, provides a theoretical underpinning for the solving process itself. In essence, the Elimination Theorem allows us to derive a *partial* solution[2] of an equation system which, subsequently, can be extended to a complete solution using the Extension Theorem.

Theorem 11 (Eliminiation Theorem, [CLO15], pp.122-123, Theorem 2). *Let $I \subseteq \mathbb{F}[X_1, \ldots, X_n]$ be an ideal and let $G \subseteq I$ be a Gröbner basis of I with respect to the lex term order, where $X_1 > X_2 > \cdots > X_n$. Then for any $k \in \{0, 1, \ldots, n-1\}$ the set*

$$G_k := G \cap \mathbb{F}[X_{k+1}, \ldots, X_n]$$

is a Gröbner basis of the k-th elimination ideal $I_k := I \cap \mathbb{F}[X_{k+1}, \ldots, X_n]$.

Theorem 12 (Extension Theorem, [CLO05], p.27). *Let $\mathbb{F}$ be algebraically closed, $I \subseteq \mathbb{F}[X_1, \ldots, X_n]$ be an ideal and let I_k be the $k-th$ elimination ideal. Then a partial solution $(a_{k+1}, \ldots, a_n) \in V(I_k)$ extends to a solution $(a_k, a_{k+1}, \ldots, a_n) \in V(I_{k-1})$ if the leading coefficient polynomials of the elements of a lex Gröbner basis of I_{k-1} do not all vanish at $(a_{k+1}, \ldots, a_n)$.*

[2] A partial solution only involves some coordinates of a complete solution.

Directly from Theorem 11 we derive the following proposition about the structure of the reduced *lex* Gröbner basis of a zero-dimensional ideal.

Proposition 11 ([CG20], Theorem 2.6). *Let $I \subseteq \mathbb{F}[X_1, \ldots, X_n]$ be a zero-dimensional ideal. Then the reduced lex Gröbner basis of I with $X_1 \geq_{lex} \cdots \geq_{lex} X_n$ has the following triangular form*

$$\begin{aligned}
&g_{1,1}(X_1, \ldots, X_n), \ldots, g_{1,t_1}(X_1, \ldots, X_n),\\
&g_{2,1}(X_2, \ldots, X_n), \ldots, g_{2,t_2}(X_2, \ldots, X_n),\\
&\ldots\\
&g_{n-1,1}(X_{n-1}, X_n), \ldots, g_{n-1,t_{n-1}}(X_{n-1}, X_n),\\
&g_{n,1}(X_n),
\end{aligned}$$

where $\deg_{X_n}(g_{i,j}) < \deg(g_{n,1})$ *for* $i \neq n$.

For zero-dimensional radical ideals in normal position there is an even stronger result regarding the form of the reduced *lex* Gröbner basis.

Definition 19. Let $I \subseteq \mathbb{F}[X_1, \ldots, X_n]$ be a zero-dimensional ideal and let $i \in \{1, \ldots, n\}$. We say that I is in *normal X_i–position* if any two distinct zeroes $(a_1, \ldots, a_n), (b_1, \ldots, b_n) \in V(I)$ satisfy $a_i \neq b_i$.

Theorem 13 (Shape Lemma, [KRoo], p.257, Theorem 3.7.25). *Let $\mathbb{F}$ be a perfect field and let $I \subseteq \mathbb{F}[X_1, \ldots, X_n]$ be a zero-dimensional and radical ideal that is in normal X_n-position. Then the reduced lex Gröbner basis of I with $X_1 \geq_{lex} \cdots \geq_{lex} X_n$ is of the form*

$$\begin{aligned}
&X_1 - g_1(X_n),\\
&X_2 - g_2(X_n),\\
&\ldots\\
&g_n(X_n),
\end{aligned}$$

where $\deg(g_1), \ldots, \deg(g_{n-1}) < \deg(g_n)$.

Note, Theorem 13 assumes that the ideal I is in normal position with respect to the *last* variable in *lex* order. The requirement of $\mathbb{F}$ being a perfect field is definitely fulfilled if $\mathbb{F}$ is a finite field or the characteristic of $\mathbb{F}$ is zero. Thus, for our particular focus on applications in symmetric cryptography, this requirement is not a limitation since ideals coming from this domain can be assumed to be defined over finite fields. For a large enough field $\mathbb{F}$, a certain linear change of variables allows us to bring any zero-dimensional ideal $I \subseteq \mathbb{F}[X_1, \ldots, X_n]$ into normal position. We remark, the field $\mathbb{F}$ need not be algebraically closed.

Proposition 12 ([KRoo], p.255, Proposition 3.7.22). *Let $R = \mathbb{F}[X_1, \ldots, X_n]$, $I \subseteq R$ a zero-dimensional ideal and let $D := \dim_{\mathbb{F}}(R/I)$. If $\mathbb{F}$ contains more than $\binom{D}{2}$ elements, there exists a tuple $(c_1, \ldots, c_{n-1}) \in \mathbb{F}^{n-1}$ such that*

$$c_1 a_1 + \cdots c_{n-1} a_{n-1} + a_n \neq c_1 b_1 + \cdots c_{n-1} b_{n-1} + b_n,$$

for all pairs of distinct zeroes $(a_1, \ldots, a_n), (b_1, \ldots, b_n) \in V(I)$. Consequently, the linear change of coordinates given by

$$\varphi : \begin{cases} X_i \mapsto X_i, i \in \{1, \ldots, n-1\}, \\ X_n \mapsto X_n - c_1 X_1 - \cdots - c_{n-1} X_{n-1}, \end{cases}$$

transforms I into an ideal in normal X_n position.

The linear change of coordinates φ (or in pure algebra lingo: the ring homomorphism) described in Proposition 12 transforms the ideal I into the ideal

$$J \coloneqq \langle f(X_1, \ldots, X_{n-1}, X_n - c_1 X_1 - \cdots - c_{n-1} X_{n-1}) : f \in I \rangle$$

and has the inverse transformation

$$\varphi^{-1} : \begin{cases} X_i \mapsto X_i, i \in \{1, \ldots, n-1\}, \\ X_n \mapsto X_n + c_1 X_1 + \cdots c_{n-1} X_{n-1}. \end{cases}$$

Now it is easy to see that $(a_1, \ldots, a_n) \in V(I)$ implies

$$(a_1, \ldots, a_{n-1}, a_n + c_1 a_1 + \cdots c_{n-1} a_{n-1}) \in V(J)$$

and, furthermore, that $(b_1, \ldots, b_n) \in V(J)$ implies

$$(b_1, \ldots, b_{n-1}, b_n - c_1 b_1 - \cdots - c_{n-1} b_{n-1}) \in V(I).$$

In total, we conclude

$$V(J) = \{(a_1, \ldots, a_{n-1}, a_n + c_1 a_1 + \cdots c_{n-1} a_{n-1}) : (a_1, \ldots, a_n) \in V(I)\},$$

or in other words, the transformation φ brings I indeed into normal X_n-position.

The Shape Lemma, stated in Theorem 13, deserves a few more remarks. Most often, ideals stemming from applications in symmetric cryptography are zero-dimensional. Hence, the requirement of being zero-dimensional is not a strong one from this point of view. That Theorem 13 requires I to be in normal X_n-position is, per se, not a limitation either. As Proposition 12 tells us, any zero-dimensional ideal can be brought into normal position via a linear transformation if the base field is large enough. This linear change of coordinates is discussed further in [MH20]. The authors of [MH20] present different approaches for bringing an ideal into normal position and analyze their computational complexities. A similar remark applies to the assumption of I being radical. In symmetric cryptography we usually work over a finite field $\mathbb{F}_q$ and, thus, any ideal I may be made radical by adding the *field polynomials*[3]

$$X_1^q - X_1, \ldots, X_n^q - X_n$$

to the ideal I. The resulting ideal

$$I_{rad} \coloneqq I + \langle X_1^q - X_1, \ldots, X_n^q - X_n \rangle = \langle I \cup \{X_1^q - X_1, \ldots, X_n^q - X_n\} \rangle$$

is radical. The downside of adding the field polynomials is, however, that it might be much harder to compute a Gröbner basis for I_{rad}. This especially applies to the case of large finite fields with cardinalities of, e.g., 2^{128}. Finite fields of this size are commonly used in symmetric cryptography and adding field polynomials of such high degrees might render any Gröbner basis computation prohibitively expensive. Only in case of small finite fields, or in Gröbner basis computations where the intermediate degree exceeds the field size, we might expect an advantage of adding the field polynomials to I.

[3] Often also called *field equations*, due to the relation $x^q = x$, for all $x \in \mathbb{F}_q$.

5.2.4 Homogeneous Ideals and Hilbert Series

The polynomial ring R is naturally a *graded* ring (in fact, a graded algebra). Meaning, R can be written as a direct sum of additive subgroups

$$R = \bigoplus_{d \geq 0} R_d,$$

such that $R_m \cdot R_n \subseteq R_{m+n}$. The gradation is given by

$$R_d := \{f \in R : \text{for all } t \in T(f),\ \deg(t) = d\} \cup \{0\}.$$

For $d \in \mathbb{N}$, we define

$$R_{\leq d} := \bigoplus_{l=0}^{d} R_l.$$

Stated differently, $R_{\leq d}$ is the set of all polynomials in R of degree at most d (together with the zero polynomial). Let $I \subseteq R$ be an ideal. For $d \in \mathbb{N}$, we denote the vector space of all homogeneous polynomials in I of degree d with

$$I_d := I \cap R_d = \{f \in I \setminus \{0\} : \text{for all } t \in T(f),\ \deg(f) = d\} \cup \{0\},$$

and the vector space of all polynomials in I of degree at most d by

$$I_{\leq d} := I \cap R_{\leq d} = \{f \in I \setminus \{0\} : \deg(f) \leq d\} \cup \{0\}.$$

Similarly, we denote the set of all terms in T of degree $d \in \mathbb{N}$ by

$$T_d := \{t \in T : \deg(t) = d\},$$

and the set of all terms in R of degree at most d with

$$T_{\leq d} := \{t \in T : \deg(t) \leq d\}.$$

Definition 20. An ideal $I \subseteq R$ is said to be *homogeneous* if there exist homogeneous polynomials $f_1, \ldots, f_m \in R$ (of not necessarily the same degree) such that

$$I = \langle f_1, \ldots, f_m \rangle.$$

Lemma 5 ([Frö98], p.100, Lemma 2)**.** *An ideal $I \subseteq R$ is homogeneous if and only if for every $f \in I$ it holds: every homogeneous component of f belongs to I. In other words,*

$$I = \bigoplus_{d \geq 0} I_d.$$

Corollary 5. *Let $I \subseteq R$ be a homogeneous ideal. Then*

$$I_{\leq d} = \bigoplus_{l=0}^{d} I_l.$$

Lemma 6. *Let $I \subseteq R$ be a homogeneous ideal and let $d \in \mathbb{N}$ such that $I_d = R_d$. Then*

$$\mathrm{LT}(I) \subseteq \langle \mathrm{LT}(I_{\leq d}) \rangle.$$

Proof. We have $T_d := \{t \in T : \deg(t) = d\} \subseteq R_d = I_d$. Hence,

$$\mathrm{LT}(I_l) \subseteq \langle \mathrm{LT}(I_d) \rangle, \text{ for all } l \geq d.$$

Indeed, let $f \in I_l$ and $t := \mathrm{LT}(f)$. Since f is homogeneous of degree $l \geq d$, we can write $t = u \cdot v$, for $u, v \in T$ and $v \in T_d$. This gives us

$$v \in T_d \subseteq R_d = I_d.$$

Hence, $v = \mathrm{LT}(v) \in \mathrm{LT}(I_d)$, and therefore $t \in \langle \mathrm{LT}(I_d) \rangle$. Since I is homogeneous, it holds for all $l \geq 0$

$$I_{\leq l} = \bigoplus_{k \leq l} I_k$$

This means, every $g \in I_{\leq l}$ can be written as

$$g = \sum_{k=1}^{l} g_k, \text{ for } g_k \in I_k.$$

Thus, $\mathrm{LT}(g) = \mathrm{LT}(g_k)$, for some $1 \leq k \leq l$. Therefore, $\mathrm{LT}(I_{\leq l}) \subseteq \bigcup_{k \leq l} \mathrm{LT}(I_k)$ and consequently

$$\mathrm{LT}(I_{\leq l}) \subseteq \langle \mathrm{LT}(I_{\leq d}) \rangle, \text{ for all } l \geq 0. \tag{15}$$

Now let $f \in I$. Then $f \in I_{\leq l}$, for some $l \in \mathbb{N}$. With the inclusion in (15), the statement follows. □

Proposition 13 ([Frö98], p.100, Lemma 4)**.** *Let $I \subseteq R$ be a homogeneous ideal in the graded ring*

$$R = \bigoplus_{d \geq 0} R_d$$

with gradation $R_d := \{f \in R : f \text{ homogeneous of degree } d\}$. Then the quotient ring R/I is a graded ring

$$R/I = \bigoplus_{d \geq 0} (R_d + I) = \bigoplus_{d \geq 0} R_d / I_d \tag{16}$$

with gradation $R_d/I_d := \{r + I_d : r \in R_d\}$.

Definition 21. Let $I \subseteq R$ be a homogeneous ideal. The *Hilbert function of R/I* is the function $\mathrm{HF}_{R/I} : \mathbb{N} \to \mathbb{N}$ with

$$\mathrm{HF}_{R/I}(d) := \dim_{\mathbb{F}}(R_d/I_d) = \dim_{\mathbb{F}}(R_d) - \dim_{\mathbb{F}}(I_d).$$

The *Hilbert series of R/I* is the generating function of $\mathrm{HF}_{R/I}$, i.e., the formal power series

$$\mathrm{HS}_{R/I}(T) := \sum_{d \geq 0} \mathrm{HF}_{R/I}(d) \cdot T^d \in \mathbb{Z}[[T]].$$

Theorem 14 ([Frö98], pp.130-131, Theorem 7 & Corollary 8)**.** *Let I be a homogeneous ideal in $\mathbb{F}[X_1, \ldots, X_n]$. Then*

$$HS_{R/I}(T) = \frac{q(T)}{(1-T)^l},$$

for some polynomial $q \in \mathbb{Z}[T]$ with $q(0) = 1$, $q(1) \neq 0$ and some $l \in \mathbb{N}$, $l \leq n$. Moreover,

- *if $l > 0$, then $HF_{R/I}(d) = p(d)$ for a polynomial $p \in \mathbb{Q}[T]$ of degree $l - 1$, and,*
- *if $l = 0$, then $HF_{R/I}(d) = 0$,*

for large enough d. In particular, l is the Krull dimension of R/I, see also [Frö98, Definition 10].

The condition $q(1) \neq 0$ in Theorem 14 guarantees that q does not contain the factor $(1-T)$.

Definition 22. Let $I \subseteq R$ be a homogeneous ideal. By Theorem 14, there exists a unique polynomial $p \in \mathbb{Q}[T]$ and a number d', such that for all $d \geq d'$ it holds

$$\mathrm{HF}_{R/I}(d) = p(d).$$

We call p the Hilbert polynomial of R/I and denote it by $\mathrm{HP}_{R/I}(T)$. The number

$$i_{reg}(I) := \min\{i \in \mathbb{N} : \mathrm{HF}_{R/I}(d) = \mathrm{HP}_{R/I}(d) \text{ for all } d \geq i\}$$

is called the *index of regularity of $HF_{R/I}$* or, simply, the *index of regularity of I*. If I is clear from the context, we also just write i_{reg}.

For zero-dimensional homogeneous ideals, the quotient ring R/I has a finite vector space dimension, see Proposition 9. In this case, the Hilbert series is, in fact, a finite sum, by Eq. (16). Proposition 14 gives an explicit account of this fact.

Proposition 14. *Let $I \subseteq \mathbb{F}[X_1, \ldots, X_n]$ be a zero-dimensional homogeneous ideal. Then the Hilbert series $HS_{R/I}(T)$ is a polynomial.*

Regular sequences play an important role when it comes to estimating the complexity of computing a Gröbner basis. For regular sequences there exists an explicit formula for the index of regularity, and thus, for the maximum degree reached during a Gröbner basis computation using Algorithm 4, see Section 6.2.2 and Section 4.3 for more details. Theorem 15 states some important properties of regular sequences.

Definition 23. A set of m homogeneous polynomials $f_1, \ldots, f_m$ in $R = \mathbb{F}[X_1, \ldots, X_n]$, for $m \leq n$, is called a *regular sequence* if the following conditions are satisfied.

1. $R/\langle f_1, \ldots, f_m\rangle \neq 0$, i.e., $\langle f_1, \ldots, f_m\rangle \neq R$.
2. For all $i \in \{1, \ldots, m\}$ and all $g \in R$ it holds, if $g \cdot f_i = 0$ in $R/\langle f_1, \ldots, f_{i-1}\rangle$, then $g = 0$ in $R/\langle f_1, \ldots, f_{i-1}\rangle$.

Definition 24. A set of m (not necessarily homogeneous) polynomials $f_1, \ldots, f_m$ in $R = \mathbb{F}[X_1, \ldots, X_n]$ is called a *regular sequence* if the homogeneous parts of highest degree $f_1^{top}, \ldots, f_m^{top}$ form a regular sequence.

Theorem 15 ([KR05], p.203, Corollary 5.2.17)**.** *Let $m, n \in \mathbb{N}$, $m \leq n$ and let $f_1, \ldots, f_m \in \mathbb{F}[X_1, \ldots, X_n]$ be homogeneous polynomials of degrees $d_1, \ldots, d_m$ which generate the ideal $I := \langle f_1, \ldots, f_m\rangle$. Then the following conditions are equivalent.*

1. *The sequence $f_1, \ldots, f_m$ is a regular sequence in $\mathbb{F}[X_1, \ldots, X_n]$.*
2. *For $I = \langle f_1, \ldots, f_m\rangle$ it holds*

$$HS_{R/I}(T) = \frac{\prod_{i=1}^{m}(1 - T^{d_i})}{(1-T)^n}.$$

3. *For every permutation* $\pi(1), \ldots, \pi(m)$ *of the sequence* $1, \ldots, m$*, the sequence* $f_{\pi(1)}, \ldots, f_{\pi(m)}$ *is a regular sequence in* $\mathbb{F}[X_1, \ldots, X_n]$.

In addition, the following properties hold.

4. *The Krull dimension of* $\mathbb{F}[X_1, \ldots, X_n]/\langle f_1, \ldots, f_m \rangle$ *is* $n - m$ *[Frö98, p.132, Lemma 12].*

5. *For* $m = n$*, the sequence* $f_1, \ldots, f_m$ *is regular if and only if the Hilbert series is a polynomial [BFS15, Proposition 4].*

Proposition 15 ([Spa12], p.47, Corollary 1.66)**.** *Let* $I := \langle f_1, \ldots, f_m \rangle \subseteq R$ *be a zero-dimensional homogeneous ideal and let* $d_i := \deg(f_i)$*. Then*

$$i_{reg}(I) = 1 + \deg(HS_{R/I}).$$

If $m = n$*, then*

$$i_{reg}(I) = 1 + \sum_{i=1}^{m} (d_i - 1).$$

Moreover, for any monomial ordering, i_{reg} *bounds the degree of all polynomials in a minimal homogeneous Gröbner basis of* I*.*

The authors of [Bar04; BFS04] generalize the notion of regular sequences to the overdefined case, i.e., to the case $m \geq n$.

Definition 25 ([BFS+05], Definition 5)**.** Let $F := \{f_1, \ldots, f_m\}$ be a set of m homogeneous polynomials in $R = \mathbb{F}[X_1, \ldots, X_n]$ generating a zero-dimensional ideal $I := \langle f_1, \ldots, f_m \rangle$. Then F is called a *semi-regular sequence* if the following conditions are satisfied.

1. $R/\langle f_1, \ldots, f_m \rangle \neq 0$, i.e., $\langle f_1, \ldots, f_m \rangle \neq R$.
2. For all $i \in \{1, \ldots, m\}$ and all $g \in R$ it holds, if $g \cdot f_i = 0$ in $R/\langle f_1, \ldots, f_{i-1} \rangle$ and $\deg(g \cdot f_i) < i_{reg}(I)$, then $g = 0$ in $R/\langle f_1, \ldots, f_{i-1} \rangle$.

Definition 26 ([BFS+05], Definition 5)**.** A set of m (not necessarily homogeneous) polynomials $f_1, \ldots, f_m$ in $R = \mathbb{F}[X_1, \ldots, X_n]$ is called a *semi-regular sequence* if the homogeneous parts of highest degree $f_1^{top}, \ldots, f_m^{top}$ form a semi-regular sequence.

Proposition 16 ([Bar04], Proposition 3.2.5)**.** *Let* $F := \{f_1, \ldots, f_m\}$ *be a set of* m *homogeneous polynomials in* $R = \mathbb{F}[X_1, \ldots, X_n]$*,* $m \geq n$*, with degrees* $d_1, \ldots, d_m$ *which generate the ideal* $I := \langle f_1, \ldots, f_m \rangle$*.*

1. *The sequence* F *is a semi-regular sequence in* R *if and only if the Hilbert series of* I *is*
$$HS_{R/I}(T) = \frac{\prod_{i=1}^{m}(1 - T^{d_i})}{(1 - T)^n}.$$
2. *If* $f_1, \ldots, f_m$ *is a semi-regular sequence, then for every permutation* π *of* $1, \ldots, m$*, the sequence* $f_{\pi(1)}, \ldots, f_{\pi(m)}$ *is also a semi-regular sequence.*
3. *If* F *is a semi-regular sequence, the index of regularity* $i_{reg}(I)$ *is the index of the first non-positive coefficient in* $HS_{R/I}(T)$*.*

For an overview and a more comprehensive treatment of different notions of "degree of regularity" we refer to [BND+20; CG20].

6.1 Buchberger Algorithm

The pioneering `Buchberger` algorithm is the first algorithm for computing Gröbner bases and was devised by Bruno Buchberger in 1965 [Buc65]. To motivate the `Buchberger` algorithm, let us discuss the reason why a set of generators $\{f_1, \ldots, f_m\} \subseteq \mathbb{F}[X_1, \ldots, X_n]$ might not be a Gröbner basis of the ideal $I := \langle f_1, \ldots, f_m \rangle$. Stated differently, we discuss why the condition

$$\langle \mathrm{LT}(f_1), \ldots, \mathrm{LT}(f_m) \rangle = \langle \mathrm{LT}(I) \rangle$$

might not be satisfied. In general, we are only guaranteed that

$$\langle \mathrm{LT}(f_1), \ldots, \mathrm{LT}(f_m) \rangle \subseteq \mathrm{LT}(I),$$

but the reverse inclusion is not necessarily true. There might exist a combination $q := h_1 f_1 + \cdots + h_m f_m \in I$, for $h_i \in \mathbb{F}[X_1, \ldots, X_n]$, such that

$$\mathrm{LT}(q) \notin \langle \mathrm{LT}(f_1), \ldots, \mathrm{LT}(f_m) \rangle.$$

This means, $\langle \mathrm{LT}(I) \rangle$ might be strictly larger than $\langle \mathrm{LT}(f_1), \ldots, \mathrm{LT}(f_m) \rangle$.

A Counterexample We give an example of a basis for an ideal that is not a Gröbner basis. Let $f_1 := XY - 2Y$ and $f_2 := X^2 - 2X - Y$ be two polynomials in $\mathbb{F}[X, Y]$ and let $I := \langle f_1, f_2 \rangle$ denote the ideal generated by f_1, f_2. Furthermore, let $\leq$ denote the graded reverse lexicographic order. Then

$$X \cdot f_1 - Y \cdot f_2 = X \cdot (XY - 2Y) - Y \cdot \left(X^2 - 2X - Y\right) = Y^2. \tag{17}$$

It holds, $Y^2 \in I$ and thus $\mathrm{LT}\left(Y^2\right) = Y^2 \in \mathrm{LT}(I)$. But

$$Y^2 \notin \langle \mathrm{LT}(f_1), \mathrm{LT}(f_2) \rangle,$$

since neither $\mathrm{LT}(f_1) = XY$ nor $\mathrm{LT}(f_2) = X^2$ divides Y^2, see Lemma 3. Hence

$$\langle \mathrm{LT}(I) \rangle \nsubseteq \langle \mathrm{LT}(f_1), \mathrm{LT}(f_2) \rangle,$$

and, thus, $\{f_1, f_2\}$ is not a Gröbner basis of I. A minimum requirement for finding a Gröbner basis of I is, thus, to extend $\{f_1, f_2\}$ by $Y^2 = X \cdot f_1 - Y \cdot f_2$. The intuition behind the kind of cancellation in Eq. (17) is the following. Whenever a polynomial combination $q = h_1 f_1 + h_2 f_2$ results in a cancellation of the leading terms $\mathrm{LT}(h_1 f_1)$, $\mathrm{LT}(h_2 f_2)$, the leading term of q might not be an element of $\langle \mathrm{LT}(f_1), \mathrm{LT}(f_2) \rangle$.

Gröbner Bases and Syzygies Putting above observations in a more general context, for $F := \{f_1, \ldots, f_m\} \subseteq R$, we are interested in those polynomial combinations $q = h_1 f_1 + \cdots + h_m f_m$ that cancel all the leading terms

$\mathrm{LT}(h_1 f_1), \ldots, \mathrm{LT}(h_m f_m)$.[1] In more technical terms, we are interested in the syzygy module[2]

$$\mathcal{S} := \mathrm{Syz}(\mathrm{LT}(F)) := \mathrm{Syz}(\mathrm{LT}(f_1), \ldots, \mathrm{LT}(f_m)).$$

The syzygy module $\mathcal{S}$ contains all possible m-tuples $(h_1, \ldots, h_m) \in R^m$ that evaluate to the zero polynomial under the evaluation homomorphism ν_H for $H := \{\mathrm{LT}(f_1), \ldots, \mathrm{LT}(f_m)\}$. This means, any $(h_1, \ldots, h_m) \in \mathcal{S}$ satisfies

$$\nu_H(h_1, \ldots, h_m) = h_1 \cdot \mathrm{LT}(f_1) + \cdots + h_n \cdot \mathrm{LT}(f_m) = 0.$$

In other words, $\mathcal{S}$ accounts for all possible cancellations between the $\mathrm{LT}(f_i)$. The crucial point here is, $\mathcal{S}$ is a *finitely* generated R-module, see Proposition 4. Thus, any syzygy, i.e., any cancellation between the $\mathrm{LT}(f_i)$, can be described by a (finite) polynomial combination of generators of $\mathcal{S}$. The last observation already hints at a basic strategy for computing a Gröbner basis of the ideal generated by F. For any generator $b = (b_1, \ldots, b_m)$ of $\mathcal{S}$ with

$$\mathrm{LT}(\nu_F(b)) = \mathrm{LT}(b_1 f_1 + \cdots b_m f_m) \notin \langle \mathrm{LT}(F) \rangle,$$

we add the element $\nu_F(b)$ to F. Subsequently, we recalculate a generating set for the new syzygy module $\mathrm{Syz}(\mathrm{LT}(F))$ and repeat this process. It is to Bruno Buchberger's merit that he resolved the technicalities of this process and established a proof of termination in [Buc65].

Buchberger's Algorithm In [Buc65], Buchberger pioneered the first algorithmic approach for computing Gröbner bases. This algorithm is now known as the `Buchberger` algorithm. Buchberger chose the following generating set

$$\{ \boldsymbol{s}_{i,j} : 1 \leq i < j \leq m \},$$

for $\mathrm{Syz}(\mathrm{LT}(f_1), \ldots, \mathrm{LT}(f_m))$, see also Proposition 4, where

$$\boldsymbol{s}_{i,j} := \frac{\mathrm{lcm}(\mathrm{LT}(f_i), \mathrm{LT}(f_j))}{\mathrm{LM}(f_i)} \cdot \boldsymbol{e}_i - \frac{\mathrm{lcm}(\mathrm{LT}(f_i), \mathrm{LT}(f_j))}{\mathrm{LM}(f_j)} \cdot \boldsymbol{e}_j.$$

In Algorithm 2, we present the `Buchberger` algorithm, albeit without any optimizations or improvements. For an improved version of Algorithm 2 that implements the *Buchberger criteria* for detecting unnecessary reductions to zero (see Proposition 5 and Proposition 6) we refer the reader to the well-known Gebauer-Möller installation [GM88].

6.2 Gröbner Bases via Macaulay Matrices

There is an intimate connection between linear algebra and Gröbner basis theory, enabling us to compute Gröbner bases using matrix algebra. This connection goes back to [GM86] and is fundamental for some of the most important Gröbner basis algorithms, such as `F4` [Fau99], see also Section 6.3, or `MatrixF5` [BFS15]. A first observation linking linear algebra and Gröbner basis theory is the following. Every ideal $I = \langle f_1, \ldots, f_m \rangle \subseteq R$ is an infinite $\mathbb{F}$-vector space and the following (infinite) set

$$B := \{ t \cdot f_i : t \in T, 1 \leq i \leq m \}.$$

[1] If $\mathrm{LT}(q) = \mathrm{LT}(h_i \cdot f_i) = \mathrm{LT}(h_i) \cdot \mathrm{LT}(f_i)$, for some i, it is evident that $\mathrm{LT}(q) \in \langle \mathrm{LT}(F) \rangle$, hence this case is clear. The case $\mathrm{LT}(q) \neq \mathrm{LT}(h_i f_i)$, for all $i \in \{1, \ldots, m\}$, is the interesting one.

[2] For $A := \{h_1, \ldots, h_m\} \subseteq R$, the syzygy module $\mathrm{Syz}(A) = \{(q_1, \ldots, q_m) \in R^m : \sum_{i=1}^m q_i h_i = 0\}$ implicitly assumes a fixed order for the elements in A. The notation $\mathrm{Syz}(A)$ might hide this order, however, we believe this causes no further confusion.

Algorithm 2: `Buchberger`

Input: A set $F := \{f_1, \dots, f_m\}$ of polynomials in R, a term order $\leq$
Result: A Gröbner basis G of the ideal generated by $f_1, \dots, f_m$

$G \leftarrow \{f_1, \dots, f_m\}$
$B \leftarrow \{\{i, j\} : 1 \leq i < j \leq s\}$
while $B \neq \emptyset$ **do**
  Select $b \leftarrow \{i, j\} \in B$ // Some selection strategy
  $B \leftarrow B \setminus \{b\}$
  $r \leftarrow \texttt{Reduce}\big(S(f_i, f_j), G, \leq\big)$
  if $r \neq 0$ **then**
    $t \leftarrow |G|$
    $f_{t+1} \leftarrow r$
    $G \leftarrow G \cup \{f_{t+1}\}$
    $B \leftarrow B \cup \{\{i, t+1\} : 1 \leq i \leq t\}$
return G

Algorithm 3: `Reduce`

Input: A polynomial p, a finite set F of non-zero polynomials, a term order $\leq$
Result: A normal form $p' \in p \bmod F$ with respect to $\leq$

$p' \leftarrow p$
while $\exists f \in F : \mathrm{LT}(f) \mid \mathrm{LT}(p')$ **do**
  Select such f // Some selection strategy
  $p' \leftarrow p' - \mathrm{LC}(p') \cdot \frac{\mathrm{LM}(p')}{\mathrm{LM}(f)} \cdot f$
return p'

generates I as vector space. Before we further explore this connection, let us introduce two new concepts.

d-Gröbner Bases A *d-Gröbner basis* of I (with respect to some term order $\leq$) is a set $G \subseteq I$ such that

$$\mathrm{LT}(I_{\leq d}) \subseteq \langle \mathrm{LT}(G) \rangle.$$

The importance of d-Gröbner bases will become clear soon, especially, once we have arrived at Eq. (18). A generating set B for a vector space in R is called *staggered* (or *triangular*) if no two distinct elements in B have the same leading term. We call a generating set for I as $\mathbb{F}$-vector space also a *linear basis* of I.

Macaulay Matrices For a finite and nonempty set of polynomials $F := \{f_1, \dots, f_m\}$ in R, $\Lambda := \{1, 2, \dots, m\}$ and $\Gamma := \Gamma(m) := T(F) = \bigcup_{f \in F} T(f)$, we call the matrix

$$\begin{aligned} \mathrm{Mac}(F) : \Lambda \times \Gamma &\to \mathbb{F}, \\ (i, t) &\mapsto C_t(f_i), \end{aligned}$$

a *Macaulay matrix* or *coefficient matrix* of F. If $M = (m_{i,t})_{i\in\Lambda,\, t\in\Gamma}$ is a Macaulay matrix, we call

$$\mathrm{pol}(M) := \left\{ \sum_{t\in\Gamma} m_{i,t} \cdot t : i \in \Lambda \right\} \setminus \{0\}$$

the *set of polynomials associated with* M or the *polynomial representation of* M or, simply, the *polynomials of* M. We define *the contour of* M to be

$$\mathrm{cont}(M) := \{f \in \mathrm{pol}(M) : \text{ for all } f \neq g \in \mathrm{Pol}(M),\ \mathrm{LT}(g) \nmid \mathrm{LT}(f)\}.$$

Let $M = (M_i)_{i\in\Lambda}$ be a description of a Macaulay matrix in terms of its rows. Then M is said to be in *row echelon form* (or *triangular*) if no two distinct rows in M have the same leading term. Furthermore, M is said to be in *reduced row echelon form* if M is in echelon form and for all $i \in \Lambda$ with $M_i \neq 0$ and all $s \in \mathrm{LT}(M) \setminus \{\mathrm{LT}(M_i)\}$ it holds

$$\mathrm{LC}(M_i) = 1 \text{ and } m_{i,s} = 0.$$

Gröbner Bases and Linear Algebra With above definitions at hand, we are ready to state the following proposition which lies at the heart of the connection between linear algebra and Gröbner basis theory.

Proposition 17 ([GM86]). *Let $I \subseteq \mathbb{F}[X_1, \ldots, X_n]$ be an ideal. If B is a staggered generating set for I as vector space, the set*

$$G := \{f \in B : \textit{for all } g \neq f \in B,\ \mathrm{LT}(g) \nmid \mathrm{LT}(f)\}$$

is a (minimal) Gröbner basis of I.

We can leverage Proposition 17 as follows. Let $F := \{f_1, \ldots, f_m\}$ generate the ideal I and let $M := \mathrm{Mac}(B)$ denote the (infinite) Macaulay matrix of all "shifts" of F

$$B = \{t \cdot f_i : t \in T,\ 1 \leq i \leq m\}.$$

If we brought M into reduced row echelon form, then this row echelon form yielded the reduced Gröbner basis of I, with the understanding that all rows whose leading terms are divisible by a leading term of another row are discarded. From an algorithmic viewpoint, this observation does not provide much help as B is, after all, infinite. Hence, it is only a constructive method but not an algorithm. However, another important observation leads us to the core principle of computing Gröbner bases through matrix algebra. For any ideal $I \subseteq R$, the ascending sequence of ideals

$$\langle \mathrm{LT}(I_{\leq 0}) \rangle \subseteq \langle \mathrm{LT}(I_{\leq 1}) \rangle \subseteq \langle \mathrm{LT}(I_{\leq 2}) \rangle \subseteq \ldots$$

eventually stabilizes (by the ascending chain condition, see [CLO15, p.80, Theorem 7]), i.e., there exists a certain $D \in \mathbb{N}$ such that

$$\langle \mathrm{LT}(I_{\leq 0}) \rangle \subseteq \langle \mathrm{LT}(I_{\leq 1}) \rangle \subseteq \ldots \subseteq \langle \mathrm{LT}(I_{\leq D}) \rangle = \langle \mathrm{LT}(I_{\leq D+1}) \rangle = \ldots .$$

Without loss of generality, we assume D to be minimal with this property. It follows

$$\mathrm{LT}(I) = \bigcup_{d\geq 0} \mathrm{LT}(I_{\leq d}) \subseteq \bigcup_{d\geq 0} \langle \mathrm{LT}(I_{\leq d}) \rangle = \bigcup_{d=0}^{D} \langle \mathrm{LT}(I_{\leq d}) \rangle = \langle \mathrm{LT}(I_{\leq D}) \rangle,$$

Hence

$$\langle \mathrm{LT}(I) \rangle = \langle \mathrm{LT}(I_{\leq D}) \rangle. \tag{18}$$

By the argument in the proof of Theorem 3 in [CLO15, p.77, Theorem 4], we also deduce

$$I = \langle I_{\leq D} \rangle.$$

The crucial point is: for large enough $D \in \mathbb{N}$, a D–Gröbner basis G of an ideal I is, in fact, a Gröbner basis of I since

$$\langle \mathrm{LT}(I) \rangle = \langle \mathrm{LT}(I_{\leq D}) \rangle \subseteq \langle \mathrm{LT}(G) \rangle.$$

In light of this observation, we may replace I in Proposition 17 with (the finite-dimensional vector space) $I_{\leq d}$ and, thereby, arrive at the following proposition.

Proposition 18. *Let $I \subseteq \mathbb{F}[X_1, \ldots, X_n]$ be an ideal and let $d \in \mathbb{N}$. If B is a staggered generating set for the vector space $I_{\leq d}$, the set*

$$G := \{f \in B : \text{for all } f \neq g \in B, \mathrm{LT}(g) \nmid \mathrm{LT}(f)\}$$

is a (minimal) d-Gröbner basis of I.

The premises in Proposition 18 deserve a comment. It is evident that a staggered generating set for any vector space U in R with

$$U \supseteq I_{\leq d}$$

yields a d–Gröbner basis as well. We will use this observation in Section 6.2.1.

Matrix-Based Approaches for Computing Gröbner Bases Above considerations only give an existential proof of $D \in \mathbb{N}$ such that $\mathrm{LT}(I) = \langle \mathrm{LT}(I_{\leq D}) \rangle$, but do not tell us how to find D. In our next considerations, we present two approaches for devising a matrix-based Gröbner basis algorithm using Proposition 18. Each approach tackles the problem of finding D in a different manner. Both approaches have in common that they find a staggered generating set for a vector space $U \supseteq I_{\leq D}$.

The first approach was initially proposed by Bruno Buchberger [Buc18] and, later, further investigated by Manuela Wiesinger-Widi [WW11]. Buchberger does not directly use the vector space $I_{\leq D}$ but, instead, works with the vector space generated by all large enough "D-shifts"

$$T_{\leq D} \cdot F = \{t \cdot f : t \in T_{\leq D}, f \in F\}.$$

In particular, Wiesinger-Widi [WW11] proves how large D must be such that $T_{\leq D} \cdot F$ yields a Gröbner basis of $I = \langle F \rangle$, see Proposition 19. Stated differently, Buchberger's approach finds a staggered generating set for a vector space $U := \mathrm{span}_{\mathbb{F}}(T_{\leq D} \cdot F)$ satisfying $U \supseteq I_{\leq D}$. We discuss Buchberger's approach in more detail in Section 6.2.1. The second approach was developed by Daniel Lazard [Laz79; Laz83] and primarily targets homogeneous ideals. The reason for targeting homogeneous ideals is due to the property that for a homogeneous ideal I it holds

$$I = \bigoplus_{d \geq 0} I_d.$$

As a consequence, for any $d \in \mathbb{N}$ we obtain

$$I_{\leq d} = \bigoplus_{j=1}^{d} I_j. \tag{19}$$

The equality in Eq. (19) suggests an efficient method for obtaining a generating set for $I_{\leq D}$: we simply join the generating sets for $I_0, \ldots, I_D$. This fact serves as the core principle behind Lazard's approach, see Section 6.2.2 for a more elaborate discussion.

In order to provide a quick reference, we summarize the relations between the concepts in this introductory part in Table 1.

Generating set B for $U \supseteq I_{\leq D}$	**Macaulay matrix of B**	**Gröbner basis of I**
staggered (triangular)	row echelon form (triangular)	minimal
	reduced row echelon form	reduced

Table 1: Relation between properties of linear bases, Macaulay matrices and Gröbner bases for an ideal $I = \langle f_1, \ldots, f_m \rangle \subseteq \mathbb{F}[X_1, \ldots, X_n]$, and for D large enough such that $\langle \mathrm{LT}(I_{\leq D}) \rangle = \langle \mathrm{LT}(I) \rangle$.

6.2.1 Buchberger's Approach

Let F be a finite set of generators for an ideal $I \subseteq \mathbb{F}[X_1, \ldots, X_n]$. Bruno Buchberger [Buc18] describes a particular connection between linear algebra and Gröbner basis theory. By transforming the Macaulay matrix of

$$T_{\leq D} \cdot F = \{t \cdot f : t \in T_{\leq D}, f \in F\},$$

into reduced row echelon form, we can extract the reduced Gröbner basis of I from this reduced row echelon form *if* D is large enough. Let us assume we already knew which D to choose. Then, Buchberger's approach follows these steps:

1. For a finite set of polynomials $F \subseteq \mathbb{F}[X_1, \ldots, X_n]$ and all term multiples

   $$T_{\leq D} \cdot F = \{t \cdot f : t \in T_{\leq D}, f \in F\},$$

   construct the Macaulay matrix of $T_{\leq D} \cdot F$. The resulting matrix $M = (m_{i,t})$ has rows indexed by $\Lambda \subseteq \mathbb{N}$ and columns indexed by $\Gamma = \bigcup_{h \in T_{\leq D} \cdot F} T(h) \subseteq T$.

2. Apply Gaussian row operations to M until M is in reduced row echelon form. Here, only the following row operations are allowed: $(1st)$, exchanging two rows, $(2nd)$, multiplying a row with a constant in $\mathbb{F}$, and, $(3rd)$, adding a constant multiple of one row to the constant multiple of another one. The $(3rd)$ operation is only allowed if the resulting row only contains zeroes or has a smaller leading term than the input rows. For more details on these row reductions see [WW11, Sec. 2.2].

3. Extract $\mathrm{cont}(M)$. As a result, $\mathrm{cont}(M)$ is the reduced Gröbner basis of $\langle F \rangle$.

The main question to ask at this point is: how large does D have to be? Proposition 19 provides an answer to this question.

Proposition 19 ([WW11], Theorem 2.3.6). *Let* $F \subseteq \mathbb{F}[X_1, \ldots, X_n]$ *and let* $d := \max\{\deg(f) : f \in F\}$. *If*

$$D := 2\left(\frac{d^2}{2} + d\right)^{2^{n-1}} + \sum_{j=0}^{n-1} (md)^{2^j}$$

and M *is the reduced row echelon form of the Macaulay matrix of* $T_{\leq D} \cdot F$, *then* $\mathrm{cont}(M)$ *is the reduced Gröbner basis of* $\langle F \rangle$.

Buchberger's approch is remarkably simple although, unfortunately, not practical. One of the problems seems to be the bound in Proposition 19 since it causes excessively large matrices. As Buchberger notes in [Buc18, §5], to compute the reduced Gröbner basis for the ideal generated by

$$\{XY^2 - X, X^2Y - X\}$$

we needed to set up a Macaulay matrix of polynomials up to degree 155, whereas a Gröbner basis computation based on the S-polynomial method, as, e.g., in the `Buchberger` algorithm, does not exceed degree 4. Buchberger [Buc18, §5] further states, in general, the S-polynomial method produces polynomials of substantially lower degree and only very few of the rows in the Macaulay matrix set up by Buchberger's matrix approach. Despite its inefficiency, Buchberger's matrix approach may be considered as a theoretical frame for other Gröbner basis algorithms leveraging linear algebra on Macaulay matrices, including Algorithm 4, and state-of-the-art algorithms such as `F4` [Fau99], `F5` [Fau02], and `MatrixF5` [BFS15].

6.2.2 Lazard's Approach for Homogeneous Ideals

In the introductory part of Section 6.2 we have discussed how the problem of finding a Gröbner basis of an ideal I reduces to the problem of finding a d–Gröbner basis of I *if* d is sufficiently large. According to Proposition 18, finding a d–Gröbner basis of I, in turn, reduces to obtaining a staggered generating set for $I_{\leq d}$. In the case of homogeneous ideals, we know an efficient way to obtain a generating set for $I_{\leq d}$. We use the relation

$$I_{\leq d} = \bigoplus_{j=0}^{d} I_j,$$

find a staggered generating set for each I_j and, eventually, join the staggered generatings sets for $I_0, \ldots, I_d$ to get a staggered generating set for $I_{\leq d}$. If $I := \langle f_1, \ldots, f_m \rangle$, for homogeneous $f_1, \ldots, f_m \in R$, then a generating set for each I_j is given by

$$B_j := \{t \cdot f_i : t \in T,\ i \in \{1, \ldots, m\},\ \deg(t \cdot f_i) = j\ \}.$$

With a generating set B_j for I_j at hand, the process of making B_j staggered boils down to basic matrix algebra, i.e., Gaussian row reduction. These steps are already a basic outline of the `Lazard` algorithm depicted in Algorithm 4. The ideas behind Algorithm 4 are based on the observations in [Laz79; Laz83] and [Giu84; Giu85].

It is clear from the discussion above that Algorithm 4 correctly computes a d–Gröbner basis.[3] Hence, at this point we may ask: at which d should we

[3] The author of [Spa12, p.43, Theorem 1.61] also provides an explicit proof.

Algorithm 4: `Lazard`

Input: A set $F := \{f_1, \dots, f_m\}$ of homogeneous polynomials in R, $d \in \mathbb{N}$, a term order $\leq$
Result: A d–Gröbner basis G of the ideal generated by F

$G \leftarrow \emptyset$
for $j = 1, \dots, d$ **do**
  $B_j \leftarrow \{t \cdot f_i : t \in T,\ i = 1, \dots, m,\ \deg(t \cdot f_i) = j\}$
  $M_j \leftarrow \mathrm{Mac}(F_j)$
  $M_j' \leftarrow$ reduced row echelon form of M_j
  $G_j \leftarrow \mathrm{cont}(M_j')$
  $G \leftarrow G \cup G_j$
return G

terminate the algorithm if our goal is to obtain a Gröbner basis? In other words, how large does d have to be such that a d–Gröbner basis is, in fact, a Gröbner basis? While it is difficult to provide a general answer for arbitrary homogeneous ideals, in the case of *zero-dimensional* homogeneous ideals we have a definitive bound on d. We develop the idead behind this bound in the following considerations.

Bound on d for Zero-Dimensional Ideals Let $I \subseteq R$ be a homogeneous zero-dimensional ideal. The polynomial ring R is a graded $\mathbb{F}$-algebra

$$R = \bigoplus_{d \geq 0} R_d,$$

with gradation $R_d = \{f \in R : f \text{ homogeneous of degree } d\}$. Since I is homogeneous, we obtain for $I_d = I \cap R_d$ that

$$R/I = \bigoplus_{d \geq 0} R_d / I_d,$$

according to Proposition 13. Here, R_d/I_d is the quotient vector space given by

$$R_d/I_d = \{r + I_d : r \in R_d\}.$$

By Proposition 9, we know that I is zero-dimensional if and only if $\dim_{\mathbb{F}}(R/I)$ is finite, i.e., if and only if

$$\dim_{\mathbb{F}}(R/I) \overset{I \text{ homog.}}{=} \sum_{d \geq 0} \dim_{\mathbb{F}}(R_d/I_d) < \infty.$$

This is equivalent to the existence of a number $d_0 \in \mathbb{N}$, such that

$$\dim_{\mathbb{F}}(R_d/I_d) = 0,\ \forall d \geq d_0,$$

which, in turn, is equivalent to

$$I_d = R_d,\ \forall d \geq d_0.$$

If we take the minimal d_0 with above property, we have $d_0 = i_{reg}$ by definition of i_{reg} in Definition 22 and since I is zero-dimensional. Indeed, in this case the Hilbert polynomial is the zero polynomial by Proposition 14 and i_{reg} is the smallest integer such that

$$\dim_{\mathbb{F}}(R_d/I_d) = \mathrm{HF}_{R/I}(d) = \mathrm{HP}_{R/I}(d) = 0, \text{ for all } d \geq i_{reg}.$$

The crucial point is: using Lemma 6, we conclude that

$$\mathrm{LT}(I) \subseteq \mathrm{LT}(I_{\leq d})\,.$$

This means, for zero-dimensional homogeneous ideals we have a definite termination criterion for Algorithm 4 such that the resulting d–Gröbner basis is, in fact, a Gröbner basis. We terminate Algorithm 4 once degree i_{reg} has been processed. In general, determining i_{reg} is a hard problem in its own regard. However, for special classes of polynomial systems, e.g., regular and semi-regular sequences, there exist explicit formulae for i_{reg}. We briefly touch upon this topic in Section 4.3.

6.3 F4 Algorithm

One of the most efficient Gröbner basis algorithms to date is the `F4` algorithm devised by Jean-Charles Faugère [Fau99]. Conceptually, the `F4` algorithm is based on the S-polynomial method (see Theorem 9) but Faugère blends it with techniques from linear algebra to achieve fast polynomial reduction. The main idea is as follows: `F4` reduces several S-polynomials at once by translating polynomial reduction of S-polynomials into row reduction of Macaulay matrices. This idea yields an efficient reduction routine and when integrated with the Buchberger criteria for detecting reductions to zero (see Proposition 5 and Proposition 6) the resulting algorithm achieves a high level of efficiency.

Our intention is to illustrate the main idea of `F4`. Hence, we only describe a version of `F4` that does not implement the Buchberger criteria. For an improved version of `F4` that implements these criteria we refer to the original article [Fau99, Sec. 2.4]. The basic outline of `F4` is depicted in Algorithm 5. It follows the same "Select, Reduce, Update" schema that also the `Buchberger` algorithm employs. The main difference to the `Buchberger` algorithm is two-fold: $(1st)$, `F4` selects a set of S-polynomials (see Line 4 in Algorithm 5) instead of individual ones. $(2nd)$, the reduction routine of `F4` (see Line 6 in Algorithm 5) reduces several S-polynomials at once instead of sequentially. We depict the reduction routine of `F4` in more detail in Algorithm 6. The process of updating the set of S-polynomials in `F4` is the same as in the `Buchberger` algorithm.

The idea behind the efficient reduction routine in Algorithm 6 is the fact that for $f \in R$ and $G := \{g_1, \ldots, g_k\} \subseteq R$ any normal $f' \in f \bmod G$ can be written as

$$f' = f - c_{i_1} \cdot t_{i_1} g_{i_1} - \cdots - c_{i_l} \cdot t_{i_n} g_{i_l},$$

for certain $c_{i_k} \in \mathbb{F}$, $t_{i_k} \in T$, and $g_{i_k} \in G$. In other words, f' is a linear combination of suitable reductors $t_{i_1} g_{i_1}, \ldots, t_{i_l} g_{i_l}$ and f. Here, $t_{i_k} g_{i_k}$ is a reductor of either a G-reducible term in $T(f)$ or a G-reducible term that has been introduced by another $t_{i_j} g_{i_j}$ in the course of the reduction process. The salient point of Algorithm 6 is the following. In a symbolic preprocessing phase (see Line 3 to Line 8), Algorithm 6 collects all G-reductors for several S-polynomials at once and, subsequenty, brings the Macaulay matrix of

$$\{S\text{-polynomials}\} \cup \{\text{collected reductors}\}$$

Algorithm 5: F4

Input: A set $F := \{f_1, \ldots, f_m\}$ of polynomials in R, a term order $\leq$
Result: A Gröbner basis G for the ideal generated by $f_1, \ldots, f_m$

```
1  G ← {f_1, ..., f_m}
2  B ← {{i, j} : 1 ≤ i < j ≤ s;}
3  while B ≠ ∅ do
4      Select non-empty subset B' ⊆ B        // Some selection strategy
5      B ← B \ {B'}
6      R ← ReduceF4(B', G, ≤)
7      for r ∈ R do
8          t ← |G|
9          f_{t+1} ← r
10         G ← G ∪ {f_{t+1}}
11         B ← B ∪ {{i, t+1} : 1 ≤ i ≤ t}
12 return G
```

into reduced row echelon form. More precisely, instead of using proper S-polynomials, Algorithm 6 works with the left and right halves of S-polynomials: for $g_i, g_j \in G$, $g_i \neq g_j$, these are the polynomials

$$\left\{ \frac{\operatorname{lcm}(\operatorname{LT}(g_i), \operatorname{LT}(g_j))}{\operatorname{LM}(g_i)} \cdot g_i \right\} \cup \left\{ \frac{\operatorname{lcm}(\operatorname{LT}(g_i), \operatorname{LT}(g_j))}{\operatorname{LM}(g_j)} \cdot g_j \right\}.$$

Then, Algorithm 6 brings the Macaulay matrix $M := \operatorname{Mac}(L)$ of

$$L = \{\text{left and right halves of } S\text{-polynomials}\} \cup \{\text{collected reductors}\}$$

into reduced row echelon form M'. Now, all non-zero rows in M' whose leading terms are not in $\operatorname{LT}(L)$ are exactly the non-zero normal forms of the selected S-polynomials.

In spirit, this technique is very similar to what we have discussed in Proposition 18. Seen from the perspective of Proposition 18, we are interested in a staggered generating set for the vector space spanned by L. Hence, instead of completely row reducing M it would suffice to bring M into triangular form, see also [CLO15, p. 575-576, Exercise 2, part (e)].

6.4 M4GB Algorithm

In 2017 Rusydi Makarim and Marc Stevens published the M4GB algorithm for computing Gröbner bases [MS17]. Like the F4 algorithm, M4GB is based on the S-polynomial method (see Theorem 9) and, thus, follows the canonical "Select, Reduce, Update" schema laid forth by the Buchberger algorithm. The main conceptual advantage of M4GB over the Buchberger algorithm is due to the following two properties:

(*a*) M4GB performs reductions only with tail-reduced reductors, and,

(*b*) it maintains a list of already used (tail-reduced) reductors for future use.

The benefit of these two properties are faster polynomial reductions because (*b*) allows to reuse an already constructed (tail-reduced) reductor instead of

Algorithm 6: `ReduceF4`

Input: A set B' of pairs, current basis G and a term order $\leq$
Result: A set R of new basis elements

$L \leftarrow \left\{ \frac{\operatorname{lcm}(\operatorname{LT}(f_i), \operatorname{LT}(f_j))}{\operatorname{LM}(f_i)} \cdot f_i : \{i, j\} \in B' \right\}$
$done \leftarrow \operatorname{LT}(L)$
while $done \neq T(L)$ **do** // Symbolic preprocessing
  Select largest term $t \in (T(L) \setminus done)$ with respect to $\leq$
  $done \leftarrow done \cup \{t\}$
  if $\exists g \in G : LT(g) \mid t$ **then**
    Select such g // Some selection strategy
    $L \leftarrow L \cup \left\{ \frac{t}{\operatorname{LM}(g)} \cdot g \right\}$
$M \leftarrow \operatorname{Mac}(L)$
$M' \leftarrow$ reduced row echelon form of M
$R' \leftarrow \operatorname{pol}(M')$
$R \leftarrow \{r \in R' : \operatorname{LT}(r) \notin \operatorname{LT}(L)\}$
return R

re-constructing it again, while (a) ensures that during a reduction step no new reducible terms are introduced.

Our discussion of `M4GB` aims at highlighting the main innovations. Therefore, in our presentation, we presume to deviate from the original version of `M4GB` as formulated in [MS17, Sec. 4] on the following points. These deviations do not affect the termination nor correctness of the algorithm. However, they allow for a clearer presentation of the core ideas. For any efficient implementation of `M4GB` we refer to the considerations in [MS17, Sec. 4.1].

No Buchberger Criteria The original version of `M4GB` in [MS17] uses the Gebauer-Möller implementation [GM88] of Buchberger's product and chain criterion (Proposition 5 and Proposition 6, respectively) to update the set of S-polynomials and discard unnecessary S-polynomials. In contrast, our description of `M4GB` uses an update routine that does not implement these criteria and that is very similar to the canonical update routine in the `Buchberger` algorithm.

Lazy Update of Reductors `M4GB` carries out its computations on two sets L and M. The set L contains the leading terms of elements in the intermediate basis. The set M stores any reductor that has appeared during a run of `M4GB`. In other words, the set M contains polynomials whose leading terms are in L, or multiples of these polynomials. In the original version of `M4GB`, the pair (L, M) is supposed to satisfy the following *`M4GB`-invariant*:

- $L \subseteq \operatorname{LT}(M)$.
- For all $f, g \in M$ it holds $\operatorname{LT}(f) = \operatorname{LT}(g) \Rightarrow f = g$. I.e., every leading term in M is unique.
- Every $f \in M$ is top-reducible by L.
- Every $f \in M$ is tail-irreducible by L.

Algorithm 7: `ReduceRecursive`

Input: A polynomial p, a finite set F of non-zero polynomials, a term order $\leq$
Result: A normal form $p' \in p \bmod F$ with respect to $\leq$

```
p' ← 0
for t ∈ T(p) do
    if ∃f ∈ F : LT(f) | t then
        Select f ∈ D                                  // Some selection strategy
        h ← ReduceRecursive ( t / LM(f) · Tail(f), F, ≤ )
        p' ← p' − C_t(p) · h
    else
        p' ← p' + C_t(p) · t
return p'
```

Whenever the set L is extended, `M4GB` updates the whole set M such that (L, M) keeps satisfying the `M4GB`-invariant. These computations might be time-consuming, which is why we describe a variant of `M4GB` that is sketched in the performance section of [MS17, Sec. 4.1] and that updates elements in M only on-demand. This means, only when an element $m \in M$ is reused, the algorithm checks if it needs to be tail-reduced with respect to the current basis $\{f \in M : \mathrm{LT}(f) \in L\}$.[4] Nevertheless, leading terms in M are still unique. Above variant outputs the same result as the original `M4GB` algorithm, albeit it is considered more performant due to time savings in the update process of M. The authors of `M4GB` call this variant a *lazy* variant, whereas we simply refer to this variant as `M4GB`.

We motivate the idea of using tail-reduced reductors as follows. Let $T_G(f)$ denote the set of G-reducible terms of some polynomial $f \in R$ and let G be the intermediate basis during a run of `M4GB`. Let us assume `M4GB` reduces a term t in a polynomial p by an appropriate reductor m and m is *not* tail-irreducible with respect to G. Then, the G-reducible terms in $p - m$ are given by

$$T_G(p - m) = (T_G(p) \cup T_G(m)) \setminus \{t\}$$

However, if m is tail-irreducible it holds $T_G(m) \subseteq \{\mathrm{LT}(m)\} = \{t\}$, and hence

$$T_G(p - m) = T_G(p) \setminus \{t\}.$$

The advantage is evident: by exclusively using tail-reduced reductors, only terms in $T(p)$ need to be reduced modulo G and no new reducible terms are introduced in any reduction step. Based on this observation, we formulate a recursive routine for computing normal forms in Algorithm 7. This recursive reduction routine serves as a conceptual framework for polynomial reductions in `M4GB`.

Let us discuss why Algorithm 7 terminates. First of all, we argue that the `ReduceRecursive` routine has a finite recursion depth. Whenever it calls itself while processing a term t, all terms being processed in this recursive call are strictly smaller than t. The last claim is evident as `ReduceRecursive` invokes itself on $\mathrm{Tail}(q)$ of a polynomial q with $\mathrm{LT}(q) = t$. Since there is no infinite strictly decreasing sequence of terms ($\leq$ is a well-ordering), the recursion depth must be finite. Furthermore, for any polynomial input to

[4] Colloquially, we also say m is tail-reduced with respect to L.

`ReduceRecursive`, only finitely many terms are being processed. Hence, Algorithm 7 eventually terminates.

In Algorithm 8 we present the main routine of `M4GB`. It follows the basic "Select, Reduce, Update" schema of the `Buchberger` algorithm. The set L contains the leading terms of elements in the intermediate basis and M contains monic polynomials whose leading terms are in L or any other monic reductor that has appeared during a run of `M4GB`. Leading terms in M are unique, which is why we use the notation

$$M[t] \coloneqq \text{the unique polynomial } f \in M \text{ with } \mathrm{LT}(f) = t$$

to reference elements in M. In contrast to the `Buchberger` algorithm in Algorithm 2, the set of pairs B contains pairs of leading terms instead of pairs of indices. Lines 3 to 7 perform the necessary computations to guarantee that all elements in M have unique leading terms. The update routine in `M4GB` is very similar to the update routine of Algorithm 2, with the only difference that in `M4GB` the (top-irreducible) intermediate basis G is referenced by the pair (L, M).

Algorithm 9 describes the reduction of polynomials in `M4GB`. As discussed at the beginning of this section, Algorithm 9 is based on the recursive reduction routine outlined in Algorithm 7. Since the variant of `M4GB` that we describe updates[5] elements in M on demand, we require a method to decide if an element $m \in M$ needs to be tail-reduced or not. Although not explicitly stated in [MS17], the authors implicitly use the concept of *generations* for this purpose. The generation of a reductor $m \in M$ is the cardinality $|L|$ of the set L at the time when m was added to M. Keeping track of the generation has the following aim. Whenever m is reused during the execution of `M4GB` and the generation of m is equal to the current generation, we know that m is tail-irreducible with respect to the current L and it can be reused without any further considerations. If the generation of m is strictly smaller than the current generation, m needs to be updated, i.e., tail-reduced modulo $\{f \in M : \mathrm{LT}(f) \in L\}$.

6.5 Signature-Based Algorithms

Even with a version of the `Buchberger` algorithm that implements the Buchberger criteria (Proposition 5 and Proposition 6, respectively), many of the S-polynomials might still reduce to zero. Hence, it is natural to ask if there is a way to detect (some of) those reductions to zero which are not covered by the Buchberger criteria. In the following, a change of perspective helps to establish even stronger criteria for detecting redundant reductions to zero. Let f be a polynomial in the ideal $I \coloneqq \langle f_1, \ldots, f_m \rangle$ generated by the polynomials $f_1, \ldots, f_m \in R$. Then f can be written as $f = \sum_{i=1}^{m} p_i f_i$, for some polynomials $p_1, \ldots, p_m \in R$ (which are not necessarily unique). This means, on the one hand we may consider f as polynomial. On the other hand, we may view f as module element $(p_1, \ldots, p_m) \in R^m$. We, thus, have the following dualism for $f \in I$

$$f = \sum_{i=1}^{m} p_i f_i \in I \longleftrightarrow f = (p_1, \ldots, p_m) \in R^m.$$

The idea behind adopting the module perspective is: it allows us to to keep track of how the S-polynomials generated during a Gröbner basis compu-

[5] Updating elements in M is conceptually different from updating the set of pairs B. Updating an element $m \in M$ means, tail-reducing it modulo the current set $\{f \in M : \mathrm{LT}(f) \in L\}$.

Algorithm 8: `M4GB`

Input: A set $F \coloneqq \{f_1, \ldots, f_m\}$ of polynomials in R, a term order $\leq$
Result: A Gröbner basis G of the ideal generated by $f_1, \ldots, f_m$

$L \leftarrow \{\mathrm{LT}(f_1)\}$
$M \leftarrow \{\mathrm{LC}(f_1)^{-1} \cdot f_1\}$
$B \leftarrow \emptyset$
for $i = 2$ **to** m **do**
  $(M, r) \leftarrow$ `ReduceM4GB`$(f_i, L, M, \leq)$
  if $r \neq 0$ **then**
    $(L, M, B) \leftarrow$ `UpdateM4GB`(r, L, M, B)
while $B \neq \emptyset$ **do**
  Select $b \leftarrow \{u, v\} \in B$ // Some selection strategy
  $B \leftarrow B \setminus \{b\}$
  $s \leftarrow S(M[u], M[v])$
  $(M, r) \leftarrow$ `ReduceM4GB`$(s, L, M, \leq)$
  if $r \neq 0$ **then**
    $B \leftarrow B \cup \{\{u, \mathrm{LT}(r)\} : u \in L\}$
    $L \leftarrow L \cup \{\mathrm{LT}(r)\}$
    $M \leftarrow M \cup \{\mathrm{LC}(r)^{-1} \cdot r\}$
return $G \coloneqq \{f \in M : \mathrm{LT}(f) \in L\}$

tation depend on the original input polynomials. This is motivated by the fact that any S-polynomial s produced during a Gröbner basis computation (with, e.g., Algorithm 2) on input $f_1, \ldots, f_m$ has the form

$$s = \sum_{i=1}^{m} s_i f_i \in I \longleftrightarrow s = (s_1, \ldots, s_m) \in R^m.$$

With information derived from the vector $(s_1, \ldots, s_m)$ it is possible to predict if an S-polynomial reduces to zero with respect to the current basis. But before we discuss how to use the information coming from $(s_1, \ldots, s_m)$, we need a few new definitions.

Definition 27. Let $\leq$ be a term order on T. The set

$$\boldsymbol{T}_m := \{t\boldsymbol{e_i} : t \in T, i \in \{1, ..., m\}\}$$

is called the set of *module terms* in R^m. Furthermore, a *compatible term order extension of* $\leq$ *to* $\boldsymbol{T}_m$ is a total order $\preceq$ on $\boldsymbol{T}_m$ that satisfies

$$\forall s, t \in T \; \forall i \in \{1, ..., m\} : s \leq t \Rightarrow s\boldsymbol{e_i} \preceq t\boldsymbol{e_i}.$$

Definition 28. Let $\preceq$ be a compatible extension of a term order $\leq$ on T to $\boldsymbol{T}_m$. The *module leading term* of a non-zero module element $\boldsymbol{g} = \sum_{i=1}^{m} g_i \boldsymbol{e_i} \in R^m$ is

$$\mathrm{MLT}(\boldsymbol{g}) \coloneqq \max_{\preceq}\{\mathrm{LT}(g_i)\, \boldsymbol{e}_i : i \in \{1, ..., m\}\}.$$

The *module leading monomial* of $\boldsymbol{g}$ is

$$\mathrm{MLM}(\boldsymbol{g}) \coloneqq \mathrm{LC}(g_i) \cdot \mathrm{LT}(g_i)\, \boldsymbol{e}_i,$$

where $\mathrm{LT}(g_i)\, \boldsymbol{e}_i = \mathrm{MLT}(\boldsymbol{g})$. For two module terms $\boldsymbol{s}, \boldsymbol{t} \in \boldsymbol{T}_m$, we say $\boldsymbol{s}$ *divides* $\boldsymbol{t}$, in symbols $\boldsymbol{s} \mid \boldsymbol{t}$, if there exists a term $u \in T$ such that $u \cdot \boldsymbol{s} = \boldsymbol{t}$.

Algorithm 9: `ReduceM4GB`

Input: A polynomial $p \in P$, a set of terms L, a set of monic reductors M, a term order $\leq$
Result: A possibly extended set M, a normal form p' modulo $\{f \in M : \mathrm{LT}(f) \in L\}$

$p' \leftarrow 0$
for $t \in T(p)$ **do**
  if $\exists m \in M : \mathrm{LT}(m) = t$ **then**
    $m \leftarrow M[t]$
    $h \leftarrow \mathrm{Tail}(m)$
    if $\mathrm{gen}(m) < |L|$ **then**
      $M \leftarrow M \setminus \{m\}$
      $(M, h) \leftarrow$ `ReduceM4GB` $(h, L, M, \leq)$
      $m \leftarrow t + h$
      $\mathrm{gen}(m) \leftarrow |L|$
      $M \leftarrow M \cup \{m\}$
    $p' \leftarrow p' - C_t(p) \cdot h$
  else if $\exists u \in L : u \mid t$ **then**
    Select such u // Some selection strategy
    $m \leftarrow M[u]$
    $v \leftarrow t/u$
    $(M, h) \leftarrow$ `ReduceM4GB` $(v \cdot \mathrm{Tail}(m), L, M, \leq)$
    $m \leftarrow t + h$
    $\mathrm{gen}(m) \leftarrow |L|$
    $M \leftarrow M \cup \{m\}$
    $p' \leftarrow p' - C_t(p) \cdot h$
  else
    $p' \leftarrow p' + C_t(p) \cdot t$
return (M, p')

Some important term order extensions are summarized in the following definition.

Definition 29. Let $\leq$ be a term order on T, let $s, t \in T$ be two terms, and let $i, j \in \{1, ..., m\}$. For $F := \{f_1, \ldots, f_m\} \subseteq R$ we denote by $\nu = \nu_F : R^m \to R$

$$\boldsymbol{h} = (h_1, \ldots, h_m) \mapsto \sum_{i=1}^{m} h_i f_i$$

the canonical evaluation homomorphism on R^m with respect to F. We define the following (strict) extensions $\prec$ of $\leq$ to $\boldsymbol{T}_m$. In each case, the associated (non-strict) order relation $\preceq$ is given by the reflexive closure of $\prec$

$$\{(\boldsymbol{s}, \boldsymbol{t}) : \boldsymbol{s}, \boldsymbol{t} \in \boldsymbol{T}_m,\ \boldsymbol{s} \prec \boldsymbol{t}\} \cup \{(\boldsymbol{t}, \boldsymbol{t}) : \boldsymbol{t} \in \boldsymbol{T}_m\}.$$

1. The *position over term order extension* $\preceq_{pot}$ is given by

$$s\boldsymbol{e_i} \prec_{pot} t\boldsymbol{e_j} :\Longleftrightarrow \begin{cases} i > j \text{ or} \\ i = j,\ s < t. \end{cases}$$

2. The *term over position order extension* $\preceq_{top}$ is given by

$$s\boldsymbol{e_i} \prec_{top} t\boldsymbol{e_j} :\Longleftrightarrow \begin{cases} s < t \text{ or} \\ s = t,\ i > j. \end{cases}$$

3. The *weighted order extension* $\preceq_w$ is given by

$$te_i \prec_w ue_j :\Longleftrightarrow \begin{cases} \mathrm{LT}(\nu(te_i)) < LT(\nu(ue_j)) \text{ or} \\ \mathrm{LT}(\nu(te_i)) = LT(\nu(ue_j)),\ i > j. \end{cases}$$

Definition 30. Let $f, g, h \in R^m$, $g \neq 0$, and let $G \subseteq R^m \setminus \{0\}$ be a set of non-zero module elements. Let $F \coloneqq \{f_1, \ldots, f_m\} \subseteq R^m$ and let $\nu = \nu_F$ be the canonical evaluation homomorphism with respect to F. Furthermore, let $\leq$ be a term order on T and let $\preceq$ be a compatible order extension on T^m. We say

1. ... f is *Sig-reducible by* g if there exists a term $t \in T(\nu(f))$ such that
   (a) $\mathrm{LT}(\nu(g)) \mid t$,
   (b) $\mathrm{sig}(f) \succeq \mathrm{sig}(ug)$, for $u = t/\mathrm{LT}(\nu(g))$.

   The element ug is called a *Sig-reductor of* f. If $t = \mathrm{LT}(f)$, we say f is *Sig-top-reducible by* g, otherwise *Sig-tail-reducible*. The reduction itself is called *Sig-top-reduction* and *Sig-tail-reduction*, respectively.

2. ... f is *Sig-reducible modulo* G if there exists some $g \in G$ such that f is *Sig*-reducible by g.

3. ... f *Sig-reduces to* h *modulo* g *in one step* if f is *Sig*-reducible by g, with reducible term $t \in T(\nu(f))$, and if

$$h = f - C_t(\nu(f)) \cdot \mathrm{LC}(\nu(g))^{-1} \cdot ug,$$

   for $u = t/\mathrm{LT}(\nu(g))$. In symbols, we write

$$f \longrightarrow_g h.$$

4. ... f *Sig-reduces to* h *modulo* G *in one step* if there exists an element $g \in G$ such that $f \longrightarrow_g h$. In symbols, we write $f \longrightarrow_G h$.

5. ... f *Sig-reduces to* h *modulo* G *(in a finite number of steps)* if

$$f \longrightarrow_G h_1 \longrightarrow_G h_2 \longrightarrow_G \cdots \longrightarrow_G h_k = h,$$

   for some $k \in \mathbb{N}$ and $h_1, \ldots, h_k \in R^m$. In symbols, we write $f \xrightarrow{*}_G h$.

6. ... ug is a *regular Sig*-reductor if ug is a *Sig*-reductor of f and $\mathrm{sig}(f) > \mathrm{sig}(ug)$. We call the corresponding reduction a *regular Sig*-reduction and denote this by

$$f \longrightarrow_{G,reg} f - c \cdot ug,$$

   for a certain $c \in \mathbb{F}$. We denote a finite number of regular *Sig*-reductions of f to h by

$$f \xrightarrow{*}_{G,reg} h.$$

7. ... ug is a *singular Sig-reductor* if ug is a *Sig*-reductor and if $\mathrm{sig}(f) = \mathrm{sig}(ug)$. We call the corresponding reduction a *singular* reduction.

8. ... f is *(regular, singular) Sig-irreducible modulo* G, if f is not (regular, singular) *Sig*-reducible modulo G.

9. ... f is *Sig-top-irreducible modulo* G if no $g \in G$ *Sig*-reduces $\mathrm{LT}(\nu(f))$.

10. ... f is *Sig-tail-irreducible modulo* G if no $g \in G$ *Sig*-reduces any term in $\mathrm{Tail}(\nu(f))$.

11. ... $f' \in R^m$ is a *Sig-normal form of* f *modulo* G if $f \xrightarrow{*}_G f'$ and f' is *Sig*-irreducible modulo G. We write

$$f \bmod G := \{f' \in R^m : f' \text{ is a } Sig\text{-normal form of } f \text{ modulo } G\}$$

for the set of all *Sig*-normal forms of f modulo G.

12. ... f' is a *regular Sig-normal form of* f *modulo* G if $f \xrightarrow{*}_{G,reg} f'$ and f' is regular *Sig*-irreducible modulo G. We write

$$f \bmod_{\mathrm{reg}} G := \{f' \in R^m : f' \text{ is a regular } Sig\text{-normal form of } f \text{ modulo } G\}$$

for the set of all regular *Sig*-normal forms of f modulo G.

13. ... f *Sig-reduces to zero (in a finite number of steps)* if there exists a syzygy $h \in \mathrm{Syz}(f_1, \ldots, f_m)$ such that $f \xrightarrow{*}_G h$. In symbols, we write

$$f \xrightarrow{*}_G 0.$$

This notation is justified by $\nu(h) = 0$. Whenever f *Sig*-reduces to the actual zero element $\mathbf{0}$ in R^m, we write $f \xrightarrow{*}_G \mathbf{0}$.

The following two definitions introduce the fundamental concept of *signatures* in the context of Gröbner basis computations.

Definition 31. Let $p \in R^m \setminus \{\mathbf{0}\}$ and let $\preceq$ be a compatible extension of a term order $\leq$ to T_m. The *signature of* p is defined as

$$\mathrm{sig}(p) := \mathrm{MLT}(p).$$

Definition 32. Let $F = \{f_1, ..., f_m\} \subseteq R$ and $I = \langle F \rangle$. For a module term $s \in T_m$, a set $G \subseteq R^m$ is called a *signature Gröbner basis (in short: Sig-Gröbner basis) of* I *up to signature* s if

$$\forall p \in R^m : \mathrm{sig}(p) \prec s \Longrightarrow p \xrightarrow{*}_G 0.$$

If G is a *Sig*-Gröbner basis for all possible signatures $s \in T_m$, we call G a *Sig-Gröbner basis of* I. Whenever the ideal I is clear from the context, we just say G is a *Sig*-Gröbner basis (up to s).

It is easy to see that *Sig*-Gröbner bases are, indeed, Gröbner bases. For any element $f \in R^m$ it holds

$$f \xrightarrow{*}_G 0 \Longrightarrow \nu(f) \xrightarrow{*}_{\nu(G)} 0,$$

where ν denotes the canonical evaluation homomorphism (with respect to a certain finite set $F \subseteq R$). We summarize this fact in the following proposition.

Proposition 20. *Let $F \subseteq R$ be a finite set of polynomials and let $I := \langle F \rangle$. If $G \subseteq R^m$ is a Sig-Gröbner basis of I, then $\nu_F(G)$ is a Gröbner basis of I.*

The conceptual basis of the signature-based Gröbner basis algorithms presented in Algorithm 10 and Algorithm 12, respectively, is developed by the next two results. Proposition 21 justifies why it suffices to only consider

regular *Sig*-reductions(and regular *S*-vectors, respectively; see Definition 33) when computing *Sig*-Gröbner bases. In short, regular *Sig*-reductions preserve the signature during the reduction process and whenever an element $\boldsymbol{f} \in R^m$ would be singular *Sig*-reducible modulo a *Sig*-Gröbner basis up to $\mathrm{sig}(\boldsymbol{f})$, Proposition 21 tells us that $\boldsymbol{f}$, in fact, *Sig*-reduces to zero. Moreover, Theorem 16 states an analogous formulation of the *S*-polynomial criterion (see Theorem 9) in the signature setting.

Proposition 21. *Let $\boldsymbol{G} \subseteq R^m$ be a Sig-Gröbner basis up to signature $\boldsymbol{s} \in R^m$ and let $\boldsymbol{f} \in R^m$ with $\mathrm{sig}(\boldsymbol{f}) = \boldsymbol{s}$. If $\boldsymbol{f}$ is singular Sig-reducible modulo $\boldsymbol{G}$, then $\boldsymbol{f} \xrightarrow{*}_{\boldsymbol{G}} 0$.*

We recall that for $f_1, \ldots, f_m \in R$ the set

$$\left\{ \boldsymbol{s}_{i,j} : 1 \leq i < j \leq m \right\},$$

where

$$\boldsymbol{s}_{i,j} \coloneqq S(\boldsymbol{e}_i, \boldsymbol{e}_j) \coloneqq \frac{\mathrm{lcm}(\mathrm{LT}(f_i), \mathrm{LT}(f_j))}{\mathrm{LM}(f_i)} \cdot \boldsymbol{e}_i - \frac{\mathrm{lcm}(\mathrm{LT}(f_i), \mathrm{LT}(f_j))}{\mathrm{LM}(f_j)} \cdot \boldsymbol{e}_j,$$

generates the R-module $\mathrm{Syz}(\mathrm{LT}(f_1), \ldots, \mathrm{LT}(f_m))$. In reference to the concept of *S*-polynomials (see Definition 15) we introduce the following definition.

Definition 33. Let $\boldsymbol{f}, \boldsymbol{g} \in R^m$ and let $\nu = \nu_F$ be the canonical evaluation homomorphism with respect to a certain finite set $F \subseteq R$. We call

$$S(\boldsymbol{f}, \boldsymbol{g}) \coloneqq \frac{\mathrm{lcm}(\mathrm{LT}(\nu(\boldsymbol{f})), \mathrm{LT}(\nu(\boldsymbol{g})))}{\mathrm{LM}(\nu(\boldsymbol{f}))} \cdot \boldsymbol{f} - \frac{\mathrm{lcm}(\mathrm{LT}(\nu(\boldsymbol{f})), \mathrm{LT}(\nu(\boldsymbol{g})))}{\mathrm{LM}(\nu(\boldsymbol{g}))} \cdot \boldsymbol{g},$$

the *S-vector of $\boldsymbol{f}$ and $\boldsymbol{g}$*. An *S*-vector $S(\boldsymbol{f}, \boldsymbol{g}) = c_u \cdot u\boldsymbol{f} - c_v \cdot v\boldsymbol{g}$, for $c_u, c_v \in \mathbb{F}$ and $u, v \in T$, is said to be *regular* if $\mathrm{sig}(u\boldsymbol{f}) \neq \mathrm{sig}(v\boldsymbol{g})$, and *singular* otherwise.

Theorem 16 ([ER13], Theorem 3)**.** *Let $\boldsymbol{s} \in \boldsymbol{T}_m$ be a module term and let $\boldsymbol{G} \subseteq R^m$ be a finite set of module elements. The set $\boldsymbol{G}$ is a signature Gröbner basis up to signature $\boldsymbol{s}$ if and only if for any element $\boldsymbol{p}$ in*

$$\{S(\boldsymbol{f}, \boldsymbol{g}) : \boldsymbol{f}, \boldsymbol{g} \in \boldsymbol{G},\ S(\boldsymbol{f}, \boldsymbol{g}) \text{ regular}\} \cup \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_m\}$$

with $\mathrm{sig}(\boldsymbol{p}) \prec \boldsymbol{s}$ it holds $\boldsymbol{p} \xrightarrow{*}_{\boldsymbol{G}, reg} 0$.

We are now in the position, to state a first signature-based Gröbner basis algorithm which builds upon the *S*-vector criterion discussed in Theorem 16. Algorithm 10 is the equivalent of Algorithm 2 in the signature setting. We remark, the vectors $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_m$ which are added to $\boldsymbol{P}$ at the beginning (in Line 2) ensure that Algorithm 10 computes a Gröbner basis of $I \coloneqq \langle f_1, \ldots, f_m \rangle$, and not of some ideal that is strictly contained in I. Up to this point it is not clear why Algorithm 10 provides any advantage over Algorithm 2. Indeed, it is only the following two results that justify the overhead of using *Sig*-reductions. In short, Proposition 22 and Proposition 23 provide criteria for detecting unnecessary reductions to zero and thus promise a gain in computational efficiency. We provide an improved version of Algorithm 10 that incorporates these criteria in Algorithm 12.

Proposition 22. *Let $\boldsymbol{f}, \boldsymbol{g} \in R^m$ and let $\boldsymbol{G} \subseteq R^m$ be a Sig-Gröbner basis up to signature $\boldsymbol{s}$. If $\boldsymbol{f}$ and $\boldsymbol{g}$ are regularly Sig-irreducible modulo G, then*

$$\mathrm{sig}(\boldsymbol{f}) = \mathrm{sig}(\boldsymbol{g}) = \boldsymbol{s} \Longrightarrow \nu(\boldsymbol{f}) = c \cdot \nu(\boldsymbol{g}),$$

for some $c \in \mathbb{F}$.

Algorithm 10: `BasicSB`

Input: Non-zero input polynomials $F = \{f_1, ..., f_m\}$, a term order $\leq$ on T, a compatible order extension $\preceq$ on $\boldsymbol{T}_m$
Output: A Gröbner basis of the ideal generated by F

$\boldsymbol{G} \leftarrow \emptyset$
$\boldsymbol{P} \leftarrow \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_m\}$
while $\boldsymbol{P} \neq \emptyset$
  Select some $\boldsymbol{f} \in \boldsymbol{P}$ with $\preceq$-minimal signature $\boldsymbol{s} = \mathrm{sig}(\boldsymbol{f})$
  $\boldsymbol{P} \leftarrow \boldsymbol{P} \setminus \{\boldsymbol{f}\}$
  $\boldsymbol{f'} \leftarrow \mathtt{ReduceSB}(\boldsymbol{f}, \boldsymbol{G}, \leq, \preceq)$
  if $\nu(\boldsymbol{f'}) \neq 0$ **then**
    $\boldsymbol{P} \leftarrow \boldsymbol{P} \cup \{S(\boldsymbol{f'}, \boldsymbol{g}) : \boldsymbol{g} \in G,\ S(\boldsymbol{f'}, \boldsymbol{g}) \text{ regular}\}$
    $\boldsymbol{G} \leftarrow \boldsymbol{G} \cup \{\boldsymbol{f'}\}$
return $\varphi(\boldsymbol{G})$

Algorithm 11: `ReduceSB`

Input: A module element $\boldsymbol{f} \in R^m$, a finite set $\boldsymbol{G}$ of non-zero module elements, a term order $\leq$ on T and a compatible order extension $\preceq$ to $\boldsymbol{T}_m$
Result: A regular *Sig*-normal form $f' \in \boldsymbol{f} \bmod_{\mathrm{reg}} \boldsymbol{G}$ with respect to $\leq$ and $\preceq$

$\boldsymbol{f'} \leftarrow \boldsymbol{f}$
while $\exists \boldsymbol{g} \in \boldsymbol{G} : \boldsymbol{f'}$ *is regular Sig-reducible by* $\boldsymbol{g}$ **do**
  Select such $\boldsymbol{g}$ // Some selection strategy
  $u \leftarrow t/\mathrm{LT}(\nu(\boldsymbol{g}))$, for some $t \in T(\nu(\boldsymbol{f}))$ with $\mathrm{LT}(\nu(\boldsymbol{g})) \mid t$
  $\boldsymbol{f'} \leftarrow \boldsymbol{f'} - C_t(\nu(\boldsymbol{f'})) \cdot \mathrm{LC}(\nu(\boldsymbol{g}))^{-1} \cdot u\boldsymbol{g}$
return p'

Proposition 23. *Let $F := \{f_1, \ldots, f_m\} \subseteq R$ and let $\boldsymbol{G} \subseteq R^m$ be a Sig-Gröbner basis up to $\boldsymbol{s}$. If there exists a syzygy $\boldsymbol{h} \in Syz(f_1, \ldots, f_m)$ such that $\mathrm{sig}(\boldsymbol{h}) \mid \boldsymbol{s}$, then $\boldsymbol{f} \xrightarrow{*}_G 0$, for all $\boldsymbol{f} \in R^m$ with $\mathrm{sig}(\boldsymbol{f}) = \boldsymbol{s}$.*

In what follows, we explain how Proposition 22 and Proposition 23, respectively, help to detect unnecessary reductions to zero. Let us assume $\boldsymbol{G} \subseteq R^m$ is a *Sig*-Gröbner basis up to signature $\boldsymbol{s} \in \boldsymbol{T}_m$. Proposition 22 shows that for any two regularly *Sig*-irreducible module elements $\boldsymbol{f}, \boldsymbol{g} \in R^m$ it holds

$$\mathrm{sig}(\boldsymbol{f}) = \mathrm{sig}(\boldsymbol{g}) = \boldsymbol{s} \implies \nu(\boldsymbol{f}) = c \cdot \nu(\boldsymbol{g}) \tag{20}$$

for some $c \in \mathbb{F}$. Moreover, according to Proposition 23 we know if there exists a syzygy $\boldsymbol{h} \in \mathrm{Syz}(f_1, \ldots, f_m)$ with $\mathrm{sig}(\boldsymbol{h}) \mid \boldsymbol{s}$, then

$$\forall \boldsymbol{f} \in R^m \text{ with } \mathrm{sig}(\boldsymbol{f}) = \boldsymbol{s} : \boldsymbol{f} \xrightarrow{*}_G 0. \tag{21}$$

The latter observation in (21) is called the *syzygy criterion*, while the former observation leads to the so-called concept of *signature rewriting*. In signature rewriting, instead of an S-vector with signature $\boldsymbol{s}$, we may regularly *Sig*-reduce any module element with the same signature $\boldsymbol{s}$ to check whether $\boldsymbol{s}$ contributes any new element to the final Gröbner basis. Gröbner basis algorithms based on this approach are called *rewrite* algorithms [ER13] and we motivate this approach further in Section 6.5.1. We can leverage the observations in (20) and (21), respectively, in the following way:

Algorithm 12: SB

Input: Non-zero input polynomials $F = \{f_1, ..., f_m\}$, a term order $\leq$ on T, a compatible order extension $\preceq$ on $\boldsymbol{T}_m$,
Output: A (minimal) Gröbner basis of the ideal generated by F

$\boldsymbol{H} \leftarrow \emptyset$
$\boldsymbol{G} \leftarrow \emptyset$
$\boldsymbol{P} \leftarrow \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_m\}$
while $\boldsymbol{P} \neq \emptyset$
  Select some $\boldsymbol{f} \in \boldsymbol{P}$ with $\preceq$-minimal signature $\boldsymbol{s} = \mathrm{sig}(\boldsymbol{f})$
  $\boldsymbol{P} \leftarrow \boldsymbol{P} \setminus \{\boldsymbol{p} \in \boldsymbol{P} : \mathrm{sig}(\boldsymbol{p}) = \boldsymbol{s}\}$
  if $\boldsymbol{s}$ *is not divisible by some* $\boldsymbol{h} \in \boldsymbol{H}$ **then**
    $\boldsymbol{f}' \leftarrow \mathtt{ReduceSB}(\boldsymbol{f}, \boldsymbol{G}, \leq, \preceq)$
    if $\nu(\boldsymbol{f}') = 0$ **then**
      $\boldsymbol{H} \leftarrow \boldsymbol{H} \cup \{\boldsymbol{s}\}$
    else
      $\boldsymbol{P} \leftarrow \boldsymbol{P} \cup \{S(\boldsymbol{f}', \boldsymbol{g}) : \boldsymbol{g} \in G,\ S(\boldsymbol{f}', \boldsymbol{g}) \text{ regular}\}$
      $\boldsymbol{G} \leftarrow \boldsymbol{G} \cup \{\boldsymbol{f}'\}$
return $\varphi(\boldsymbol{G})$

- for a given signature $\boldsymbol{s}$ we need to regularly *Sig*-reduce only one S-vector with signature $\boldsymbol{s}$;
- if we know that the signature of $\boldsymbol{f} \in R^m$ is a multiple of the signature of a syzygy, we may skip the reduction of $\boldsymbol{f}$ entirely. This is why a signature-based Gröbner basis algorithm keeps track of syzygy signatures.

The result of strengthening Algorithm 10 with the criteria presented in Proposition 22 and Proposition 23, respectively, leads to Algorithm 12. Like `BasicSB`, the `SB` algorithm computes a (*Sig*-)Gröbner basis incrementally, in the sense that it proceeds by increasing signature. Because of Proposition 22, `SB` only *Sig*-reduces one module element for any given signature. Hence, all elements in $\boldsymbol{G}$ have distinct signatures.

6.5.1 Signature Rewriting

The consequence of Proposition 22 is that for computing a *Sig*-Gröbner basis through Algorithm 12, only one S-vector needs to be regular *Sig*-reduced per signature $\boldsymbol{s}$. The statement of Proposition 22 is even stronger: instead of an S-vector with signature $\boldsymbol{s}$, it is possible to regular *Sig*-reduce any module element with signature $\boldsymbol{s}$. For example, any element in

$$\mathcal{C}_{\boldsymbol{s}} := \{u\boldsymbol{f} : \boldsymbol{f} \in \boldsymbol{G},\ u \in T,\ \mathrm{sig}(u\boldsymbol{f}) = \boldsymbol{s}\}$$

is a valid candidate, even if for $u\boldsymbol{f} \in \mathcal{C}_{\boldsymbol{s}}$ the element $\boldsymbol{f}$ is not involved in any S-vector. Now, the following question arises: which element in $\mathcal{C}_{\boldsymbol{s}}$ is a good choice? The authors of [EF17, Sec. 7.3] state the following heuristic. It is desirable to choose an element that is "easier" to *Sig*-reduce than the other elements in $\mathcal{C}_{\boldsymbol{s}}$. There are two canonical selections.

1. We choose an element $u\boldsymbol{f} \in \mathcal{C}_{\boldsymbol{s}}$ such that $\boldsymbol{f}$ has been added to $\boldsymbol{G}$ most recently. Here, we hope that $\boldsymbol{f}$ is furthest *Sig*-reduced modulo $\boldsymbol{G}$ and thus $u\boldsymbol{f}$ might be easier to handle in the subsequent computations.

2. We choose an element $uf \in \mathcal{C}_s$ with minimal leading term, i.e., such that $\mathrm{LT}(uf) \leq \mathrm{LT}(vg)$, for all $vg \in \mathcal{C}_s$. Here, we expect the fewest *Sig*-reduction steps that need to be carried out.

To address the matter of choosing a "good" element in $\mathcal{C}_s$, we introduce the following definitions. In particular, the *canonical rewriter*, for a given *rewrite order*, will be such a "good" element to choose.

Definition 34. Let G be the intermediate basis constructed during a run of Algorithm 12. A *rewrite order* is a total order $\preceq_r$ on G. For $s \in T_m$, an element $f \in R^m$ is called *the canonical rewriter of (signature)* s *with respect to* G *and* $\preceq_r$ if $G = \emptyset$ or if

$$f = \max_{\preceq_r} \{ f \in G : \text{there exists } u \in T \text{ such that } \mathrm{sig}(uf) = s \}.$$

We often just say "the canonical rewriter of s", because the set G and the rewrite order $\preceq_r$ will be clear from the context. Concerning the notation, we use the subscript r to distinguish a generic rewrite order $\preceq_r$ from a generic module term order $\preceq$.

Definition 35. Let G be the set constructed during a run of Algorithm 12, let $\leq$ be a term order on T and let $\preceq$ be a compatible order extension to T_m.

1. At the time when some $f \in R^m$ is added to G, we define $\mathrm{num}(f) := |G|$. Then the relation

$$f \prec_{rat} g :\Longleftrightarrow num(f) < num(g).$$

   is called the *number order* on G.

2. The relation

$$f <_{rat} g :\Longleftrightarrow \begin{cases} Sig(f)LT(g) < Sig(g)LT(f) \text{ or} \\ Sig(f)LT(g) = Sig(g)LT(f),\ Sig(f) < Sig(g). \end{cases}$$

   is called the *ratio order* on G.

We write $\preceq_{num}$ and $\preceq_{rat}$ for the corresponding (non-strict) order relation given by the reflexive closure of $\prec_{num}$ and $\prec_{rat}$.

For any given signature s, the idea is to reduce the canonical rewriter of s, i.e., the $\preceq_r$-maximal element in $\mathcal{C}_s$. Let us examine the intuition behind this choice. For $uf, vg \in \mathcal{C}_s$ with $f \preceq_{rat} g$, the ratio order $\preceq_{rat}$ inherits its name from examining the (symbolic) ratios

$$\frac{\mathrm{sig}(uf)}{\mathrm{LT}(uf)} \preceq_{rat} \frac{\mathrm{sig}(vg)}{\mathrm{LT}(vg)}.$$

Colloquially speaking, $\preceq_{rat}$ chooses the element in $\mathcal{C}_s$ with "largest ratio". The intuition behind examining these (symbolic) ratios is two-fold. Here, a large ratio $\mathrm{sig}(vg)\,/\mathrm{LT}(vg)$ either indicates a large signature $\mathrm{sig}(vg)$ or a small leading term $\mathrm{LT}(vg)$, or both. In case of a large signature $\mathrm{sig}(vg)$, we reason that g must have been added to G fairly recently and, thus, we expect g to be almost *Sig*-reduced modulo the current basis G. In case of a small leading term $\mathrm{LT}(vg)$, we expect that only a few *Sig*-reductions are necessary until vg is regular *Sig*-irreducible modulo the current G. For the number order, we have a similar intuition. Since Algorithm 12 adds elements to G in strictly increasing signatures, it is evident that

$$\mathrm{num}(f) < \mathrm{num}(g) \Longleftrightarrow \mathrm{sig}(f) \prec \mathrm{sig}(g)\,,$$

and, thus, we expect that a higher number indicates fewer necessary *Sig*-reductions modulo the current basis $\boldsymbol{G}$. For the fact that both orders, $\preceq_{num}$ and $\preceq_{rat}$, are rewrite orders, and for a more general discussion of rewrite orders we refer to [EF17, Sec. 7.3]. Adding the feature of signature rewriting to Algorithm 12 is easy: we additionally parametrize Algorithm 12 by a rewrite order $\preceq_r$ and substitute Line 5 with the following instruction.

> Select some $\boldsymbol{p} \in \boldsymbol{P}$ with $\preceq$-minimal signature $\boldsymbol{s} = \mathrm{sig}(\boldsymbol{p})$ and choose $\boldsymbol{f} \leftarrow v\boldsymbol{g} \in \mathcal{C}_{\boldsymbol{s}}$ such that $\boldsymbol{g}$ is the canonical rewriter of $\boldsymbol{s}$ with respect to $\boldsymbol{G}$ and $\preceq_r$.

Concluding Remarks The field of signature-based Gröbner basis algorithms is large and in the literature there are many variations or generalisations of the seminal `F5` algorithm [Fau02]. Our introductory treatment of signature-based Gröbner basis algorithms only aims at presenting the core ideas and enough background to motivate the basic principles behind signature-based criteria for detecting redundant reductions to zero. In particular, there are several optimisations and improvements of Algorithm 12 which we have not discussed. These include the sig-poly-pair optimization, a more elaborate use of Koszul syzygies or fast comparisons of signatures. We refer to [RS12, Sec. 3] and [ER13, Sec. 5.1] for a discussion of these improvements. Inherently, our introductory treatment cannot cover the many intricacies of signature-based Gröbner basis algorithms, which is why for a more detailed treatment we refer to the comprehensive survey article [EF17] and the excellent textbook [CLO15, Chapter 10, §4]. Specifically, the concept of *rewrite bases*, first published in [ER13] and further discussed in [EF17], provides a unifying framework for many existing signature-based Gröbner basis algorithms.

6.6 FGLM Algorithm

The `FGLM` algorithm plays a vital role in the context of solving systems of polynomial equations via Gröbner bases. Since we use the elimination property of *lex* Gröbner bases for polynomial system solving, computing such a *lex* Gröbner basis is one of the key problems in the solving process (see Theorem 11 and Section 4.3 for a more elaborate discussion). In practice, it is considered more efficient to compute a Gröbner basis with respect to a fast term order (e.g., *degrevlex*) and then change to a *lex* Gröbner basis through a basis conversion algorithm. For a zero-dimensional ideal I, the `FGLM` algorithm [FGL+93] converts a Gröbner basis G_1 of I with respect to a given term order $\leq_1$ to the reduced Gröbner basis G_2 of I with respect to a new term order $\leq_2$. The main reason behind the particular advantage of converting G_1 to G_2 using the `FGLM` algorithm compared to directly computing G_2 is the following. Given G_1, `FGLM` reduces the problem of computing G_2 to finding linear relations in $R/\langle G_1\rangle = R/I$. In essence, computing G_2 becomes a problem of detecting linear (in)dependence. In this section, we motivate the conceptual idea behind the `FGLM` algorithm, albeit without giving a formally complete proof of termination and correctness. We refer to the original article [FGL+93] (or the textbook [CLO05, Ch. 2, §3] in case $\leq_2 = \leq_{lex}$) for a full proof of termination and correctness.

The `FGLM` algorithm, fundamentally, relies on two facts. For a zero-dimensional ideal I ...

1. ... the quotient ring

$$R/I = \{[f]_I : f \in R\},$$

is a finite-dimensional $\mathbb{F}$-vector space (see Proposition 9). Here, $[f]_I = \{g \in R : f - g \in I\} = \{f + i : i \in I\} = f + I$.

2. ... and for a Gröbner basis G of I, with respect to a certain term order $\leq$, it holds

$$R/I \cong R \bmod G$$

as $\mathbb{F}$-vector spaces, where $R \bmod G := \{f \bmod G : f \in R\}$. In other words, any Gröbner basis of I provides a canonical representation of R/I.

In the following, we will prove 1. and 2. and, in particular, show that the set

$$\{t \in T : t \notin \langle \mathrm{LT}(G) \rangle\}. \tag{22}$$

is a vector space basis of R/I. Let $G := \{f_1, \ldots, f_k\}$ denote a Gröbner basis of I with respect to a given term order $\leq$. Then, any element $f \in R$ can be uniquely represented as

$$f = q_1 f_1 + \cdots + q_k f_k + r,$$

with either $r = 0$ or no term in r is divisible by any element in $\mathrm{LT}(G)$. Hence, the unique normal form $r = f \bmod G$ can be regarded as a canonical representative of the residue class $[f]_I$, i.e.,

$$[f]_I = [q_1 f_1 + \cdots + q_k f_k + r]_I = [r]_I = [f \bmod G]_I.$$

Consequently, the set

$$B := \{t \in T : t \text{ is not divisible by any element in } \mathrm{LT}(G)\}$$

generates R/I as $\mathbb{F}$-vector space.[6] Due to Lemma 3 it holds

$$t \in \langle \mathrm{LT}(G) \rangle \iff \exists s \in \mathrm{LT}(G) : s \mid t.$$

Put differently, a term $t \in T$ lies in B if and only if t lies in the complement of $\langle \mathrm{LT}(G) \rangle$. Therefore

$$B = \{t \in T : t \notin \langle \mathrm{LT}(G) \rangle\}.$$

We will show that B is finite and the elements in B are $\mathbb{F}$-linearly independent in R/I. By Proposition 9, the set B is finite: for every $i = 1, \ldots, n$ there exists a number $a_i \in \mathbb{N}$ such that $X_i^{a_i} \in \mathrm{LT}(G)$. Without loss of generality, let $a_1, \ldots, a_n$ be minimal with this property. Therefore

$$B \subseteq \{X_1^{e_1} \cdots X_n^{e_n} : e_i < a_i \text{ for } i = 1, \ldots, n\}.$$

As a next step, we show that the elements in B are $\mathbb{F}$-linearly independent in R/I. Indeed, assume

$$\sum_{t \in B} c_t \cdot [t]_I = [0]_I,$$

and not all coefficients $c_t \in \mathbb{F}$ are zero. It follows, $p := \sum_{t \in B} c_t \cdot t \neq 0$ and $p \in I$. Hence, $\mathrm{LT}(p) \in \langle \mathrm{LT}(I) \rangle = \langle \mathrm{LT}(G) \rangle$, which is a contradiction to

[6] Technically speaking, the canonical embedding of B in R/I given by $\{[t]_I : t \in B\}$ generates R/I.

$\mathrm{LT}(p) \in B$. This concludes the proof that B is a vector space basis of R/I. Moreover, the mapping

$$\begin{aligned} \varphi : R/I &\to R \bmod G, \\ [f]_I &\mapsto f \bmod G \end{aligned}$$

is well-defined and bijective. For well-definedness and injectivity, let $[f]_I, [g]_I$ bet two residuce classed in R/I. Then

$$\begin{aligned} [f]_I = [g]_I &\iff f - g \in I \\ &\iff f - g = 0 \mod G \\ &\iff f \bmod G = g \bmod G \\ &\iff \varphi([f]_I) = \varphi([g]_I). \end{aligned}$$

We note, the second equivalence is, in general, only valid if G is a Gröbner basis (more precisely, the direction "$\Longrightarrow$"). Surjectivity is clear, since for $f \bmod G \in R \bmod G$ it holds $\varphi([f]_I) = f \bmod G$. It remains to show that φ is a homomorphism of vector spaces. Let $\lambda \in \mathbb{F}$ and $f, g \in R$. Then

$$\begin{aligned} \varphi(\lambda[f]_I + [g]_I) &= \varphi([\lambda f + g]_I) = (\lambda f + g) \bmod G \\ &= \lambda(f \bmod G) + g \bmod G = \lambda\varphi([f]_I) + \varphi([g]_I). \end{aligned}$$

Letting B_1 and B_2 denote the basis of R/I with respect to $\leq_1$ and $\leq_2$, respectively, we conclude

$$R/I = \mathrm{span}_{\mathbb{F}}(B_1) = \mathrm{span}_{\mathbb{F}}(B_2)\,,$$

and

$$\dim_{\mathbb{F}}(R/I) = |B_1| = |B_2|.$$

In general, the set B differs for different term orders. However, since B is a basis for the quotient ring R/I, the number of elements in B does not change under different term orders and, thus, is an invariant of the ideal I.

Conceptual Principle The conceptual principle behind `FGLM` is explained as follows: the algorithm steps through terms in increasing $\leq_2$-order and checks if the current term is linearly dependent on the previous terms modulo G_1. If such a linear relation is found (it will be found since B_1 is finite), this linear relation yields an element of the eventual new Gröbner basis G_2. In more detail, let t denote the current term and $t_1, \ldots, t_m$ all the previous terms. Then, for $t_1 < t_2 < \cdots < t_m < t$, we have

$$\begin{aligned} &t \text{ linearly dependent on } t_1, \ldots, t_m \text{ modulo } G_1 \\ &\iff \exists c_1, \ldots, c_m \in \mathbb{F}, \text{ not all zero} : \sum_{i=1}^{m} c_i \cdot t_i = t \mod G_1 \\ &\iff 0 \neq t - \sum_{i=1}^{m} c_i \cdot t_i \in I = \langle G_1 \rangle. \end{aligned}$$

As a consequence, all term multiples of the current term t can be skipped in subsequent steps. If no linear relation modulo G_1 is found, the current term is linearly independent of all previous terms modulo G_1 and, thus, constitutes a basis element of B_2. Whenever a term t is added to B_2, all the products $t \cdot X_i$, for $i = 1, \ldots, n$, are tested for linear dependence modulo G_1 in subsequent steps. When `FGLM` terminates, G_2 is a $\leq_2$-Gröbner basis of I and B_2 is a vector space basis for R/I.

To see why linear independence modulo G_1 implies linear independence modulo G_2, let us assume that Algorithm 13 terminates and correctly computes a $\leq_2$-Gröbner basis G_2. Then, since both G_1 and G_2 are Gröbner bases of I, we have

$$R \bmod G_1 \cong R/I \cong R \bmod G_2$$

and, hence, linear (in)dependence modulo G_1 is equivalent to linear (in)dependence modulo G_2.

To summarize, the particular advantage of `FGLM` compared to directly computing a Gröbner basis (using any generic algorithm, like, e.g., `Buchberger` or `F4`) is the following. If we already have a Gröbner basis G_1 of I with respect to a term order $\leq_1$, computing a Gröbner basis G_2 of I with respect to another term order $\leq_2$ boils down to testing for linear (in)dependence modulo G_1. This concludes our discussion and we refer to Algorithm 13 for our presentation of the `FGLM` algorithm.

Algorithm 13: `FGLM` [FGL+93]

Input: A Gröbner basis G_1 of a zero-dimensional ideal I for a term order $\leq_1$, another term order $\leq_2$

Result: A reduced Gröbner basis G_2 of I with respect to $\leq_2$

$G_2 \leftarrow \emptyset$
$\mathcal{S} \leftarrow \{1\}$
$N \leftarrow \{X_1, \ldots, X_n\}$
while $N \neq \emptyset$ **do**
  $t \leftarrow \min_{\leq_2} N$
  $N \leftarrow N \backslash \{t\}$
  if *t is a term multiple of some element in* $\mathrm{LT}(G_2)$ **then**
    continue
  if *there exists a linear relation* $t = \sum_{s \in \mathcal{S}} c_s \cdot s \mod G_1$ **then**
    $p \leftarrow t - \sum_{s \in \mathcal{S}} c_s \cdot s$
    $G_2 \leftarrow G_2 \cup \{p\}$
  else
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{t\}$
    $N \leftarrow N \cup \{t \cdot X_i : i = 1, \ldots, n\}$
return G_2

Part III

SCIENTIFIC PUBLICATIONS

7

ALGEBRAIC CRYPTANALYSIS OF STARK-FRIENDLY DESIGNS: APPLICATION TO MARVELLOUS AND MIMC

Based on the peer-reviewed conference publication

[ACG+19] Martin R. Albrecht, Carlos Cid, Lorenzo Grassi, Dmitry Khovratovich, Reinhard Lüftenegger, Christian Rechberger, and Markus Schofnegger. "Algebraic Cryptanalysis of STARK-Friendly Designs: Application to MARVELLous and MiMC." in: *Advances in Cryptology - ASIACRYPT 2019*. Ed. by Steven D. Galbraith and Shiho Moriai. Vol. 11923. Lecture Notes in Computer Science. Springer, 2019, pp. 371–397. DOI: 10.1007/978-3-030-34618-8_13

Abstract The block cipher Jarvis and the hash function Friday, both members of the MARVELLous family of cryptographic primitives, are among the first proposed solutions to the problem of designing symmetric-key algorithms suitable for transparent, post-quantum secure zero-knowledge proof systems such as ZK-STARKs. In this paper we describe an algebraic cryptanalysis of Jarvis and Friday and show that the proposed number of rounds is not sufficient to provide adequate security. In Jarvis, the round function is obtained by combining a finite field inversion, a full-degree affine permutation polynomial and a key addition. Yet we show that even though the high degree of the affine polynomial may prevent some algebraic attacks (as claimed by the designers), the particular algebraic properties of the round function make both Jarvis and Friday vulnerable to Gröbner basis attacks. We also consider MiMC, a block cipher similar in structure to Jarvis. However, this cipher proves to be resistant against our proposed attack strategy. Still, our successful cryptanalysis of Jarvis and Friday does illustrate that block cipher designs for "algebraic platforms" such as STARKs, FHE or MPC may be particularly vulnerable to algebraic attacks.

Keywords Gröbner Basis, MARVELLous, Jarvis, Friday, MiMC, ZK-STARK, Algebraic Cryptanalysis, Arithmetic Circuits

7.1 Introduction

Background. Whenever a computation on sensitive data is outsourced to an untrusted machine, one has to ensure that the result is correct. Examples are database updates, user authentications, and elections. The underlying problem, formally called *computational integrity*, has been theoretically solved since the 1990s with the emergence of the PCP theorem. But the performance of actual implementations was too poor to handle any computation of practical interest. Only recently a few proof systems have appeared where the proving time is quasi-linear in the computation length (which is typically represented as an arithmetic circuit), e.g. ZK-SNARKs [PHG+13], Bulletproofs [BBB+18], and ZK-STARKs [BBH+18b]. While they all share the overall structure, these proof systems differ in details such as the need of a trusted setup, proof size, verifier scalability, and post-quantum resistance.

The cryptographic protocols that make use of such systems for zero-knowledge proofs often face the problem that whenever a hash function is involved, the associated circuit is typically long and complex, and thus the hash computation becomes a bottleneck in the proof. An example is the Zerocash cryptocurrency protocol [BCG+14b]: in order to spend a coin anonymously, one has to present a zero-knowledge proof that the coin is in the set of all valid coins, represented by a Merkle tree with coins as leaves. When a traditional hash function such as SHA-256 is used in the Merkle tree, the proof generation takes almost a minute for 28-level trees such as in Zcash [BHH+19], which represents a real obstacle to the widespread use of privacy-oriented cryptocurrencies.

The demand for symmetric-key primitives addressing the needs of specific proof systems has been high, but only a few candidates have been proposed so far: a hash function based on Pedersen commitments [BHH+19], MPC-oriented LowMC [ARS+15b], and big-field MiMC [AGR+16; AGP+19]. Even worse, different ZK proof systems use distinct computation representations. Concretely, ZK-SNARKs prefer pairing-friendly curves over prime scalar fields, Bulletproofs uses a fast curve over a scalar field, whereas ZK-STARKs are most comfortable operating over binary fields. Hence, the issue of different representations further limits the design space of ZK-friendly primitives.

STARKs. ZK-STARKs [BBH+18b] is a novel proof system which, in contrast to SNARKs, does not need a trusted setup phase and whose security relies only on the existence of collision-resistant hash functions. The computation is represented as an execution trace, with polynomial relations among the trace elements. Concretely, the trace registers must be elements of some large binary field, and the polynomials should have low degree. The proof generation time is approximately[1] $O(S \log S)$, where

$$S \approx (\text{Maximum polynomial degree} \times \text{Trace length}) .$$

The STARK paper came with a proposal to use Rijndael-based hash functions, but as these have been shown to be insecure [KBN09], custom designs are clearly needed.

Jarvis and Friday. Ashur and Dhooghe recently addressed this need with the proposal of the block cipher JARVIS and the hash function FRIDAY [AD18]. The primitives were immediately endorsed by the ZK-STARK authors[2] as possible solutions to reduce the STARK generation cost in many applications. The new hash function was claimed to offer up to a 20-fold advantage over Pedersen hashes and an advantage by a factor of 2.5 over MiMC-based hash functions, regarding the STARK proof generation time [BS18].

Albeit similar in spirit to MiMC, JARVIS comes with novel design elements in order to considerably reduce the number of rounds, while still aiming to provide adequate security. In the original proposal several types of algebraic attacks were initially ruled out, and security arguments from RIJNDAEL/AES were used to inform the choice of the number of rounds, leading to a statement that attacks were expected to cover up to three rounds only. An extra security margin was added, leading to a recommendation of 10 rounds

[1] We omit optimisations related to the trace layout.

[2] The ciphers were announced among high anticipation of the audience at the prime Ethereum conference DevCon4, held in November 2018 [BS18].

for the variant with an expected security of 128 bits. Variants with higher claims of security were also specified.

Algebraic Attacks. This class of attacks aims to utilise the algebraic properties of a construction. One example is the Gröbner basis attack, which proceeds by modelling the underlying primitive as a multivariate system of equations which is then solved using off-the-shelf Gröbner basis algorithms [Buc65; CLO15; Fau99; Fau02]. After some initial success against certain stream cipher constructions [Cou02; Cou03], algebraic attacks were also considered against block ciphers [MR02; CB07], albeit with limited success. Even approaches combining algebraic and statistical techniques [AC09] were later shown not to outperform known cryptanalytic techniques [WSM+11]. As a result algebraic attacks are typically not considered a major concern for new block ciphers. We note however that Gröbner basis methods have proven fruitful for attacking a number of public-key schemes [FGO+13; AG11; ACF+15; FPP14; FGP+15].

Contribution. In this paper we show that, while the overall design approach of Jarvis and Friday seems sound, the choice for the number of rounds is not sufficient to offer adequate security. We do this by mounting algebraic attacks on the full-round versions of the primitives with the help of Gröbner bases. Our results show that designers of symmetric-key constructions targeting "algebraic platforms" – such as STARKs, FHE and MPC – must pay particular attention to the algebraic structure of their ciphers, and that algebraic attacks should receive renewed attention from the cryptographic community.

Organisation. The remainder of this work is organised as follows. In Section 7.2 we briefly describe the block cipher Jarvis and the hash function Friday. Following, we discuss various algebraic attacks in Section 7.3, including higher-order differential attacks, interpolation attacks, and in particular attacks using Gröbner bases. In the following sections, we describe our attacks, including key-recovery attacks on Jarvis in Section 7.4 and preimage attacks on Friday in Section 7.5. In Section 7.6, we describe our experimental results from running the attacks and discuss our findings. Finally, in Section 7.7 we analyse the S-box layer of Jarvis and compare it to the AES.

7.2 MARVELlous

MARVELlous [AD18] is a family of cryptographic primitives specifically designed for STARK applications. It includes the block cipher Jarvis as well as Friday, a hash function based on this block cipher. We briefly describe the two primitives in this section.

As usual, we identify functions on $\mathbb{F}_{2^n}$ with elements in the quotient ring

$$\mathcal{R} := \mathbb{F}_{2^n}[X] / \langle X^{2^n} - X \rangle.$$

Whenever it is clear from the context, we refer to the corresponding polynomial representation in the above quotient ring when we speak of a function on $\mathbb{F}_{2^n}$ and use the notation $F(X)$, or just F, for the coset $F(X) + \langle X^{2^n} - X \rangle \in \mathcal{R}$.

7.2.1 Jarvis

JARVIS is a family of block ciphers operating on a state and a key of n bits, thus working entirely over the finite field $\mathbb{F}_{2^n}$. The construction is based on ideas used by the AES, most prominently the *wide-trail design* strategy, which guarantees security against differential and linear (statistical) attacks. However, where AES uses multiple small S-boxes in every round, JARVIS applies a single nonlinear transformation to the whole state, essentially using one large n-bit S-box. The S-box of JARVIS is defined as the generalised inverse function $S : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ with

$$S(x) := \begin{cases} x^{-1} & x \neq 0 \\ 0 & x = 0, \end{cases}$$

which corresponds to the element

$$S(X) := X^{2^n-2} \in \mathcal{R}.$$

We note that this specific S-box makes the construction efficient in the STARK setting, because verifying it uses only one quadratic constraint (note that the equality $\frac{1}{x} = y$ is equivalent to the equality $x \cdot y = 1$, and the constraint for the full S-box can be written as $x^2 \cdot y + x = 0$). We refer to [BBH+18b; AD18] for more details.

The linear layer of JARVIS is composed by evaluating a high-degree affine polynomial

$$A(X) := L(X) + \hat{c} \in \mathcal{R},$$

where $\hat{c} \in \mathbb{F}_{2^n}$ is a constant and

$$L(X) := \sum_{i=0}^{n-1} l_{2^i} \cdot X^{2^i} \in \mathcal{R}$$

is a linearised permutation polynomial. Note that the set of all linearised permutation polynomials in $\mathcal{R}$ forms a group under composition modulo $X^{2^n} - X$, also known as the *Betti-Mathieu* group [LN96].

In JARVIS, the polynomial A is built from two affine monic permutation polynomials B, C of degree 4, that is

$$B(X) := L_B(X) + b_0 := X^4 + b_2X^2 + b_1X + b_0 \in \mathcal{R}$$

and

$$C(X) := L_C(X) + c_0 := X^4 + c_2X^2 + c_1X + c_0 \in \mathcal{R}$$

satisfying the equation

$$A = C \circ B^{-1}.$$

The operator $\circ$ indicates composition modulo $X^{2^n} - X$ and B^{-1} denotes the compositional inverse of B (with respect to the operator $\circ$) given by

$$B^{-1}(X) := {L_B}^{-1}(X) + {L_B}^{-1}(b_0).$$

Here, ${L_B}^{-1}$ denotes the inverse of L_B under composition modulo $X^{2^n} - X$, or in other words, the inverse of L_B in the Betti-Mathieu group. We highlight that the inverse B^{-1} shares the same affine structure with B, i.e. it is composed of a linearised permutation polynomial ${L_B}^{-1}$ and a constant term in $\mathbb{F}_{2^n}$, but has a much higher degree.

One round of JARVIS is shown in Figure 1. Additionally, a whitening key k_0 is applied before the first round.

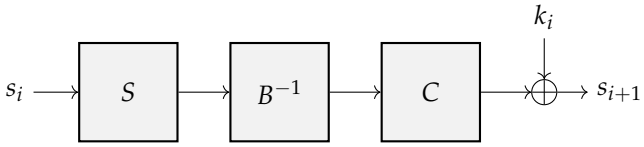


Figure 1: One round of the JARVIS block cipher. For simplicity, the addition of the whitening key is omitted.

Key Schedule

The key schedule of JARVIS shares similarities with the round function itself, the main difference being that the affine transformations are omitted. In the key schedule, the first key k_0 is the master key and the next round key k_{i+1} is calculated by adding a round constant c_i to the (generalised) inverse $S(k_i)$ of the previous round key k_i. One round of the key schedule is depicted in Figure 2.

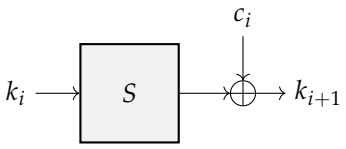


Figure 2: The key schedule used by the JARVIS block cipher.

The first round constant c_0 is randomly selected from $\mathbb{F}_{2^n}$, while subsequent round constants c_i, $1 \leq i \leq r$, are calculated using the relation

$$c_i := a \cdot c_{i-1} + b$$

for random elements $a, b \in \mathbb{F}_{2^n}$.

Instantiations

The authors of [AD18] propose four instances of JARVIS-n, where $n \in \{128, 160, 192, 256\}$. For each of these instances the values c_1, a, b, and the polynomials B and C are specified. Table 2 presents the recommended number of rounds r for each instance, where the claimed security level is equal to the key size (and state size) n. We will use $r \in \mathbb{N}$ throughout this paper to denote the number of rounds of a specific instance.

Instance	n	# of rounds r
JARVIS-128	128	10
JARVIS-160	160	11
JARVIS-192	192	12
JARVIS-256	256	14

Table 2: Instances of the JARVIS block cipher [AD18].

7.2.2 Friday

FRIDAY is a hash function based on a Merkle-Damgård construction, where the block cipher JARVIS is transformed into a compression function using the Miyaguchi-Preneel scheme. In this scheme, a (padded) message block

m_i, $1 \leq i \leq t$, serves as input m to a block cipher $E(m, k)$ and the respective previous hash value h_{i-1} serves as key k. The output of the block cipher is then added to the sum of m_i and h_{i-1}, resulting in the new hash value h_i. The first hash value h_0 is an initialization vector and taken to be the zero element in $\mathbb{F}_{2^n}$ in the case of FRIDAY. The final state h_t is the output of the hash function. The hash function FRIDAY is thus defined by the following iterative formula

$$h_0 := IV := 0,$$
$$h_i := E(m_i, h_{i-1}) + h_{i-1} + m_i,$$

for $1 \leq i \leq t$, as illustrated in Figure 3.

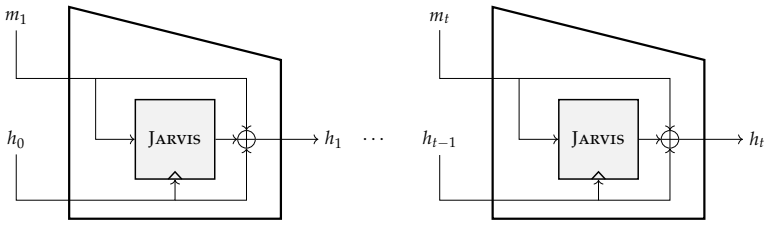


Figure 3: The FRIDAY hash function.

7.3 Overview of Algebraic Attacks on Jarvis and Friday

From an algebraic point of view, JARVIS offers security mainly by delivering a high degree for its linear transformations and for the S-box. In the original proposal, the authors analyse the security against various algebraic attack vectors, such as higher-order differential attacks and interpolation attacks.

7.3.1 Higher-Order Differential Attacks

Higher-order differential attacks [Knu94a] can be regarded as algebraic attacks that exploit the low algebraic degree of a nonlinear transformation. If this degree is low enough, an attack using multiple plaintexts and their corresponding ciphertexts can be mounted. In more detail, if the algebraic degree of a Boolean function f is d, then when applying f to all elements of an affine vector space $\mathcal{V} \oplus c$ of dimension $> d$ and taking the sum of these values, the result is 0, i.e.

$$\bigoplus_{v \in \mathcal{V} \oplus c} v = \bigoplus_{v \in \mathcal{V} \oplus c} f(v) = 0.$$

Finding such a distinguisher possibly allows the attacker to recover the secret key.

However, higher-order differential attacks pose no threat to JARVIS. Indeed, the algebraic degree of $S(X) = X^{2^n-2}$ is the Hamming weight of $2^n - 2$, which is equal to $n - 1$ and thus maximal (note that the S-box is a permutation). This makes higher-order differential attacks and zero-sum distinguishers infeasible after only one round of JARVIS.

7.3.2 Interpolation Attacks

Interpolation attacks were introduced in 1997 [JK97] and are another type of algebraic attack where the attacker constructs the polynomial corresponding

to the encryption (or decryption) function without knowing the secret key. The basis of interpolation attacks is a consequence of the Fundamental Theorem of Algebra: given $d+1$ pairs $(x_0, y_0), \ldots, (x_d, y_d)$ of elements in a certain field $\mathbb{F}$, there is a *unique* polynomial $P(X) \in \mathbb{F}[X]$ of degree at most d which satisfies

$$P(x_i) = y_i$$

for all $0 \leq i \leq d$. To put it another way, the polynomial $P(X)$ *interpolates* the given pairs (x_i, y_i), which is why it deserves the denotation *interpolation polynomial*. There are several approaches for calculating all the coefficients of the interpolation polynomial. A classical technique is to choose Lagrange's basis $(L_0, L_1, \ldots, L_d)$, with

$$L_i(X) := \prod_{\substack{j=0 \\ j \neq i}}^{d} \frac{X - x_j}{x_i - x_j} \in \mathbb{F}[X]\,,$$

as a basis for the $\mathbb{F}$-vector space $\mathbb{F}[X]$ and read off the solution $(p_0, \ldots, p_d)$ from the resulting system of equations

$$y_i = P(x_i) = p_0 L_0(x_i) + p_1 L_1(x_i) + \ldots + p_d L_d(x_i),\ 0 \leq i \leq d.$$

Lagrange's basis leads to a complexity of $\mathcal{O}(d^2)$ field operations and so does Newton's basis $\{N_0, N_1, \ldots, N_d\}$ with

$$N_i(X) := \prod_{j=0}^{i-1} (X - x_j) \in \mathbb{F}[X]\,.$$

A different approach uses the fact that polynomial interpolation can be reduced to polynomial evaluation, as discussed by Horowitz [Hor72] and Kung [Kun73], leading to a complexity of $\mathcal{O}(d \log^2 d)$ field operations. In essence, this approach relies on the Fast Fourier Transform for polynomial multiplication.

From the above complexity estimates, it is thus desirable that the polynomial representation of the encryption function reaches a high degree and forces all possible monomials to appear. In Jarvis, a high word-level degree is already reached after only one round; additionally the polynomial expression of the encryption function is also dense after only two rounds. It follows that interpolation attacks pose no threat to Jarvis.

7.3.3 Gröbner Basis Attacks

The first step in a Gröbner basis attack is to describe the primitive by a system of polynomial equations. Subsequently, a Gröbner basis [Buc65; CLO15] for the ideal defined by the corresponding polynomials is calculated and finally used to solve for specified variables. In more detail, Gröbner basis attacks consist of three phases:

1. Set up an equation system and compute a Gröbner basis (typically for the *degrevlex* term order for performance reasons) using an algorithm such as Buchberger's algorithm [Buc65], F4 [Fau99], or F5 [Fau02].

2. Perform a change of term ordering for the computed Gröbner basis (typically going from the *degrevlex* term order to the *lex* one, which facilitates computing elimination ideals and hence eliminating variables)

using an algorithm such as FGLM [FGL+93]. Note that in our applications all systems of algebraic equations result in zero-dimensional ideals, i.e. the systems have only finitely many solutions.

3. Solve the univariate equation for the last variable using a polynomial factoring algorithm, and substitute into other equations to obtain the full solution of the system.

Cost of Gröbner Basis Computation. For a generic system of n_e polynomial equations

$$F_1(x_1, \ldots, x_{n_v}) = F_2(x_1, \ldots, x_{n_v}) = \cdots = F_{n_e}(x_1, \ldots, x_{n_v}) = 0$$

in n_v variables $x_1, \ldots, x_{n_v}$, the complexity of computing a Gröbner basis [BFP12] is

$$\mathcal{C}_{\text{GB}} \in \mathcal{O}\left(\binom{n_v + D_{\text{reg}}}{D_{\text{reg}}}^{\omega}\right),$$

where $2 \leq \omega < 3$ is the linear algebra exponent representing the complexity of matrix multiplication and D_{reg} is the degree of regularity. The constants hidden by $\mathcal{O}(\cdot)$ are relatively small, which is why $\binom{n_v+D_{\text{reg}}}{D_{\text{reg}}}^{\omega}$ is typically used directly. In general, computing the degree of regularity is a hard problem. However, the degree of regularity for "regular sequences" [BFS+05] is given by

$$D_{\text{reg}} = 1 + \sum_{i=1}^{n_e} (d_i - 1), \tag{23}$$

where d_i is the degree of F_i. Regular sequences have $n_e = n_v$. More generally, for "semi-regular sequences" (the generalisation of regular sequences to $n_e > n_v$) the degree of regularity can be computed as the index of the first non-positive coefficient in

$$H(z) = \frac{1}{(1-z)^{n_v}} \times \prod_{i=1}^{n_e} (1 - z^{d_i}).$$

It is conjectured that most sequences are semi-regular [Frö85]. Indeed, experimental evidence suggests random systems behave like semi-regular systems with high probability. Hence, assuming our target systems of equations behave like semi-regular sequences, i.e. they have no additional structure, the complexity of computing a Gröbner basis depends on (a) the number of equations n_e, (b) the degrees $d_1, d_2, \ldots, d_{n_e}$ of the equations, and (c) the number of variables n_v. Crucially, our experiments described later in the paper indicate that the systems considered in this work do not behave like regular sequences.

Cost of Gröbner Basis Conversion. The complexity of the FGLM algorithm [FGL+93] is

$$\mathcal{C}_{\text{FGLM}} \in \mathcal{O}\left(n_v \cdot \deg(\mathcal{I})^3\right),$$

where $\deg(\mathcal{I})$ is called the *degree of the ideal* and defined as the dimension of the quotient ring $\mathbb{F}[X_1, X_2, \ldots, X_n]/\mathcal{I}$ as an $\mathbb{F}$-vector space. For the systems we are considering in this paper – which are expected to have a unique solution in $\mathbb{F}$ – the dimension of $R/\mathcal{I}$ corresponds to the degree of the unique univariate polynomial equation in the reduced Gröbner basis with respect to the canonical lexicographic order [KR00, Theorem 3.7.25].

Again, the hidden constants are small, permitting to use $n_v \cdot \deg(\mathcal{I})^3$ directly. A sparse variant of the algorithm also exists [FM11b] with complexity $\mathcal{O}\left(\deg(\mathcal{I})(N_1 + n_v \log \deg(\mathcal{I}))\right)$, where N_1 is the number of nonzero entries of a multiplication matrix, which is sparse even if the input system spanning $\mathcal{I}$ is dense. Thus, the key parameter to establish for estimating the cost of this step is $\deg(\mathcal{I})$.

Cost of Factoring. Finally, we need to solve for the last variable using the remaining univariate polynomial equation obtained by computing all necessary elimination ideals. This can be done by using a factorisation algorithm. For example, the complexity of a modified version of the Berlekamp algorithm [Gen07] to factorise a polynomial P of degree D over $\mathbb{F}_{2^n}$ is

$$\mathcal{C}_{\text{Sol}} \in \mathcal{O}\left(D^3n^2 + Dn^3\right).$$

In our context, we can however reduce the cost of this step by performing the first and second steps of the attack for two (or more) (plaintext, ciphertext) pairs and then considering the GCD of the resulting univariate polynomials, which are univariate in the secret key variable k_0. Computing polynomial GCDs is quasi-linear in the degree of the input polynomials. In particular, we expect

$$\mathcal{C}_{\text{Sol}} \in \mathcal{O}\left(D(\log(D))^2\right).$$

We will again drop the $\mathcal{O}(\cdot)$ and use the expressions directly.

Our Algebraic Attacks on MARVELlous. All attacks on MARVELlous presented in this paper are inherently Gröbner basis attacks which, on the one hand, are based on the fact that the S-box $S(X) = X^{2^n-2}$ of Jarvis can be regarded as the function $S : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$, where

$$S(x) = x^{-1}$$

for all elements *except* the zero element in $\mathbb{F}_{2^n}$. As a consequence, the relation

$$y = S(x) = x^{-1}$$

can be rewritten as an equation of degree 2 in two variables, namely

$$x \cdot y = 1,$$

which holds everywhere except for the zero element in $\mathbb{F}_{2^n}$. We will use this relation in our attacks, noting that $x = 0$ occurs with a negligibly small probability for $n \geq 128$.

On the other hand, we exploit the fact that the decomposition of the affine polynomial A originates from two low-degree polynomials B and C. When setting up the associated equations for Jarvis, we introduce intermediate variables in such a way that the low degree of B and C comes into effect, and then show that the particular combination of the inverse S-box $S(X) = X^{2^n-2}$ with the affine layer in Jarvis is vulnerable to Gröbner basis attacks.

Based on the above observations, we describe in the next sections:

- a key-recovery attack on reduced-round Jarvis and an optimised key-recovery attack on full-round Jarvis;
- its extension to a (two-block) preimage attack on full-round Friday;
- a more efficient direct preimage attack on full-round Friday.

7.4 Gröbner Basis Computation for Jarvis

We first describe a straightforward approach, followed by various optimisations which are necessary to extend the attack to all rounds.

7.4.1 Reduced-Round Jarvis

Let $B, C \in \mathcal{R}$ be the polynomials of the affine layer in JARVIS. Furthermore, in round i of JARVIS let us denote the intermediate state between the application of B^{-1} and C as x_i, for $1 \leq i \leq r$ (see Figure 4).

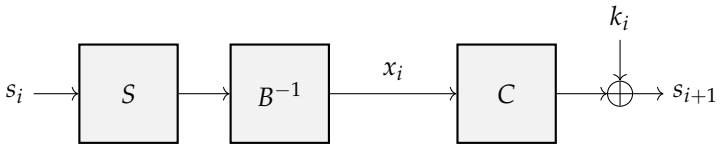


Figure 4: Intermediate state x_i in one round of the encryption path.

As a result, two consecutive rounds of JARVIS can be related by the equation

$$(C(x_i) + k_i) \cdot B(x_{i+1}) = 1 \tag{24}$$

for $1 \leq i \leq r-1$. As both polynomials B and C have degree 4, equation (24) yields a system of $r-1$ polynomial equations, each of degree 8, in the variables $x_1, \ldots, x_r$ and $k_0, \ldots, k_r$. To make the system dependent on the plaintext p and the ciphertext c, we add the two equations

$$B(x_1) \cdot (p + k_0) = 1, \tag{25}$$

$$C(x_r) = c + k_r \tag{26}$$

to this system. Additionally, two successive round keys are connected through the equation

$$(k_{i+1} + c_i) \cdot k_i = 1 \tag{27}$$

for $0 \leq i \leq r-1$. In total, the above description of JARVIS amounts to $2 \cdot r + 1$ equations in $2 \cdot r + 1$ variables, namely:

- $r-1$ equations of degree 8 (equation (24)),
- one equation of degree 5 (equation (25)),
- one equation of degree 4 (equation (26)),
- r equations of degree 2 (equation (27)),

in the $2 \cdot r + 1$ variables $x_1, \ldots, x_r$ and $k_0, \ldots, k_r$. Since the number of equations is equal to the number of variables, we can estimate the complexity of a Gröbner basis attack by using Equation (23). According to this estimate, the computation of a Gröbner basis for the above system of equations is prohibitively expensive for full-round JARVIS. For example, Equation (23) predicts a complexity of ≈ 120 bits (when setting $\omega = 2.8$) for computing a Gröbner basis for $r = 6$. However, we note that we were able to compute such a basis in practice (Section 7.6), which indicates that the above estimate is too pessimistic.

7.4.2 Optimisations for an Attack on Full-Round Jarvis

In order to optimise the computation from the previous section and extend it to full-round JARVIS, we introduce two main improvements. First, we reduce the number of variables and equations used for intermediate states. Secondly, we relate all round keys to the master key, which helps to further reduce the number of variables.

A More Efficient Description of Intermediate States

The main idea is to reduce the number of equations and variables for intermediate states at the expense of an increased degree in some of the remaining equations. By relating a fixed intermediate state x_i to the respective preceding and succeeding intermediate states x_{i-1} and x_{i+1}, we obtain the equations

$$B(x_i) = \frac{1}{C(x_{i-1}) + k_{i-1}}, \tag{28}$$

$$C(x_i) = \frac{1}{B(x_{i+1})} + k_i \tag{29}$$

for $2 \leq i \leq r-1$. Since both B and C are *monic affine* polynomials of degree 4, we claim that it is possible to find *monic affine* polynomials

$$D(X) := X^4 + d_2X^2 + d_1X + d_0$$

and

$$E(X) := X^4 + e_2X^2 + e_1X + e_0,$$

also of degree 4, such that

$$D(B) = E(C).$$

Indeed, comparing corresponding coefficients of $D(B)$ and $E(C)$ yields a system of 5 linear equations in the 6 unknown coefficients $d_0, d_1, d_2, e_0, e_1, e_2$, which can then be solved. We explain the construction of D and E in more detail in Appendix 7.A.

From now on let us assume we have already found appropriate polynomials D and E. After applying D and E to Equation (28) and Equation (29), respectively, we equate the right-hand side parts of the resulting equations and get

$$D\left(\frac{1}{C(x_{i-1}) + k_{i-1}}\right) = E\left(\frac{1}{B(x_{i+1})} + k_i\right) \tag{30}$$

for $2 \leq i \leq r-1$. Eventually we obtain a system of polynomial equations of degree 36 by clearing denominators in Equation (30).

The crucial point is that variables for every second intermediate state may now be dropped out of the description of JARVIS. This is because we can consider either only evenly indexed states or only odd ones, and by doing so, we have essentially halved the number of equations and variables needed to describe intermediate states. We note that in all optimised versions of our attacks we only work with *evenly* indexed intermediate states, as this choice allows for a more efficient description of JARVIS compared to working with odd ones.

Finally we relate the plaintext p and the ciphertext c to the appropriate intermediate state x_2 and x_r, respectively, and set

$$D\left(\frac{1}{p + k_0}\right) = E\left(\frac{1}{B(x_2)} + k_1\right), \tag{31}$$

$$C(x_r) + k_r = c. \tag{32}$$

Here, the degree of Equation (31) is 24, while Equation (32) has degree 4.

Remarks. It is worth pointing out that the above description uses several implicit assumptions. First, it may happen that some intermediate states become zero, with the consequence that our approach will not find a solution. However, this case only occurs with a negligibly small probability, in particular when considering instances with $n \geq 128$. If this event occurs we can use another plaintext-ciphertext pair. Secondly, when we solve the optimised system of equations (i.e. the system we obtain after applying D and E), not all of the solutions we find for this system are guaranteed to be valid solutions for the original system of equations. Lastly, Equation (32) implicitly assumes an even number of rounds. If we wanted to attack an odd number of rounds instead, this equation had to be adjusted accordingly.

Relating Round Keys to the Master Key

Two consecutive round keys in JARVIS are connected by the relation

$$k_{i+1} = \frac{1}{k_i} + c_i$$

if $k_i \neq 0$, which is true with high probability for large state sizes n. As a consequence, each round key is a rational function of the master key k_0 of degree 1, i.e.

$$k_{i+1} = \frac{\alpha_i \cdot k_0 + \beta_i}{\gamma_i \cdot k_0 + \delta_i}.$$

We provide the exact values for α_i, β_i, γ_i, and δ_i in Appendix 7.B. Expressing k_i as a rational function of k_0 in Equation (30) and Equation (32) raises the total degree of these equations to 40 and 5, respectively. On the other hand, the degree of Equation (31) remains unchanged.

7.4.3 Complexity Estimates of Gröbner Basis Computation for Jarvis

Assuming the number of rounds r to be even, the aforementioned two improvements yield

- $\frac{r}{2} - 1$ equations of degree 40 (equation (30)),
- one equation of degree 24 (equation (31)),
- one equation of degree 5 (equation (32)),

in $\frac{r}{2} + 1$ variables (the intermediate states $x_2, x_4, \ldots, x_r$ and the master key k_0). Since the number of equations equals the number of variables, we may calculate the degree of regularity using Equation (23), again assuming the system behaves like a regular sequence.

Our results for the degree of regularity, and thus also for the complexity of computing a Gröbner basis, are listed in Table 3. Note that we assume $\omega = 2.8$. However, this is possibly a pessimistic choice, as the regarded systems are sparse. We therefore also give the complexities for $\omega = 2$ in parentheses.

These values show that we are able to compute Gröbner bases for, and therefore successfully attack, all full-round versions of JARVIS. We note that, even when pessimistically assuming that the memory complexity of a Gröbner basis computation is asymptotically the same as its time complexity (the memory complexity of any algorithm is bounded by its time complexity)

r	n_v	D_{reg}	Complexity in bits
6	4	106	63 (45)
8	5	145	82 (58)
10 (JARVIS-128)	6	184	100 (72)
12 (JARVIS-192)	7	223	119 (85)
14 (JARVIS-256)	8	262	138 (98)
16	9	301	156 (112)
18	10	340	175 (125)
20	11	379	194 (138)

Table 3: Complexity estimates of Gröbner basis computations for r-round JARVIS.

and when considering the time-memory product (which is highly pessimistic from an attacker's point of view), our attacks against JARVIS-256 are still valid.

7.5 Gröbner Basis Computation for Friday

In this section, we let $F : \mathbb{F}_{2^n} \times \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ indicate the application of one block of FRIDAY.

7.5.1 Extending the Key-Recovery Attack on Jarvis to a Preimage Attack on Friday

Using the same equations as for JARVIS described in Section 7.4, a preimage attack on FRIDAY may also be mounted. At its heart, the attack on FRIDAY with r rounds is an attack on JARVIS with $r - 1$ rounds.

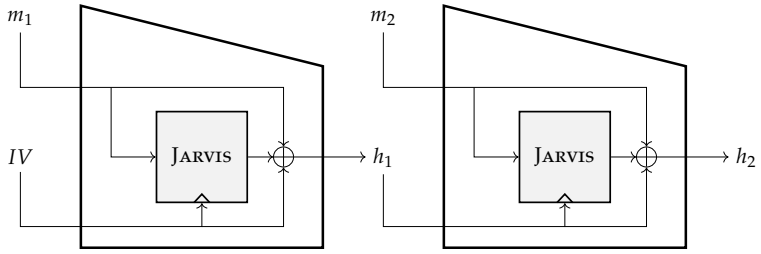


Figure 5: Two blocks of FRIDAY.

We work with two blocks of FRIDAY, hence a message m is the concatenation

$$m = m_1 \,||\, m_2$$

of two message blocks $m_1, m_2 \in \mathbb{F}_{2^n}$. The output of the first block is denoted by h_1 and the known (final) hash value of the second block is denoted by h_2. The hash values h_1 and h_2 can be expressed as

$$h_1 = F(m_1, IV)$$

and

$$h_2 = F(m_2, h_1).$$

The initialization vector IV is just the zero element in $\mathbb{F}_{2^n}$. We refer to Figure 5 for an illustration of the introduced notation.

Our preimage attack proceeds as follows: in the first part, we use random values $\hat{m}_1$ for the input to the first block to populate a table T_1 in which each entry contains a pair $(\hat{m}_1, \hat{h}_1)$, where $\hat{h}_1$ denotes the corresponding intermediate hash value

$$\hat{h}_1 := F(\hat{m}_1, IV).$$

In the second part, we find pairs (m'_2, h'_1) with

$$F(m'_2, h'_1) = h_2,$$

or in other words, pseudo preimages for the known hash value h_2. To find such a pseudo preimage, we fix the sum $m_2 + h_1$ to an arbitrary value $v_0 \in \mathbb{F}_{2^n}$, i.e. we set

$$v_0 := m_2 + h_1.$$

This has two effects:

1. In the second block, the value v_1 entering the first round of Jarvis is fixed and known until the application of the second round key. Essentially, this means that one round of Jarvis can be skipped.
2. Since $v_0 = m_2 + h_1$ is fixed and known, the final output v_2 of Jarvis is defined by
$$v_2 := v_0 + h_2$$
and thus also known.

In the current scenario, the intermediate hash value h_1 serves as master key for the r round keys $k_1, k_2, \ldots, k_r$ applied in the second block. Using v_1 as plaintext and v_2 as ciphertext, an attack on Jarvis with $r - 1$ rounds is sufficient to reveal these round keys. Once one of the round keys is recovered, we calculate the second part h'_1 of a pseudo preimage (m'_2, h'_1) by applying the inverse key schedule to the recovered key. Finally, we set

$$m'_2 := h'_1 + v_0$$

and thereby obtain the remaining part of a pseudo preimage. How the presented pseudo preimage attack on r-round Friday reduces to a key-recovery attack on $(r-1)$-round Jarvis is outlined in Figure 6.

Conceptually, we repeat the pseudo preimage attack many times (for different values of v_0) and store the resulting pairs (m'_2, h'_1) in a table T_2. The aim is to produce matching entries $(\hat{m}_1, \hat{h}_1)$ and (m'_2, h'_1) in T_1 and T_2 such that

$$\hat{h}_1 = h'_1,$$

which implies

$$F(m'_2, F(\hat{m}_1, IV)) = F(m'_2, \hat{h}_1) = F(m'_2, h'_1) = h_2,$$

giving us the preimage $(\hat{m}_1, m'_2)$ we are looking for.

Remark. The (input, output) pairs (v_1, v_2) we use for the underlying key-recovery attack on Jarvis are *not* proper pairs provided by, e.g., an encryption oracle for Jarvis. Thus, it may happen that for some pairs (v_1, v_2) the key-recovery attack does not succeed, i.e. there is no key h'_1 which maps v_1 to v_2. The probability for such an event is

$$P_{\text{fail}} = \left(\frac{2^n - 1}{2^n}\right)^{2^n} = \left(1 - \frac{1}{2^n}\right)^{2^n} \approx \lim_{k \to \infty} \left(1 - \frac{1}{k}\right)^k = \frac{1}{e}$$

for large n.

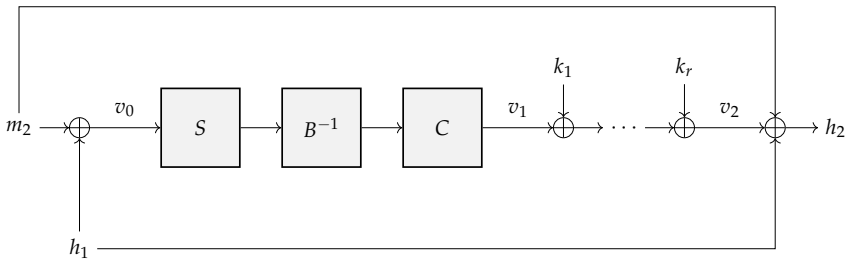


Figure 6: Internals of the second block of FRIDAY. The values v_0, v_1 and v_2 are known.

7.5.2 Complexity of Generating Pseudo Preimages

The cost of generating pseudo preimages is not negligible. Hence, we cannot afford to generate tables T_1 and T_2, each with $2^{\frac{n}{2}}$ entries, and then look for a collision. However, given the attack complexities for JARVIS in Table 3, an attack on 9-round JARVIS has a complexity of around 83 bits (assuming $\omega = 2.8$). Considering JARVIS-128, for example, this means we can generate up to 2^{45} pseudo preimages.

Let us assume we calculate 2^{10} pseudo preimages $(\hat{m}_1, m'_1)$ and $2^{\frac{n}{2}}$ intermediate pairs $(\hat{m}_1, \hat{h}_1)$, in both cases for FRIDAY instantiated with JARVIS-128. This leaves us with a table T_1 containing $2^{\frac{n}{2}}$ $(\hat{m}_1, \hat{h}_1)$ pairs and a table T_2 containing 2^{10} (m'_2, h'_1) pairs.

Assuming that all hash values in T_1 are pairwise distinct and that also all hash values in T_2 are pairwise distinct, the probability that we find at least one hash collision between a pair in T_1 and a pair in T_2 is

$$P = 1 - \prod_{i=0}^{|T_2|-1} \left(1 - \frac{|T_1|}{2^{128} - i}\right), \tag{33}$$

which is, unfortunately, too low for $|T_1| = 2^{\frac{n}{2}}$. However, we can increase this probability by generating more entries for T_1. Targeting a total complexity of, e.g., ≈ 120 bits, we can generate 2^{118} such entries. Note that the number of expected collisions in a table of m random n-bit entries is

$$N_c = m - 2^n + 2^n \cdot \left(\frac{2^n - 1}{2^n}\right)^m.$$

Therefore, the expected number of unique values in such a table is

$$N_u = \left(1 - \frac{N_c}{m}\right) \cdot m = m - N_c = 2^n - 2^n \cdot \left(\frac{2^n - 1}{2^n}\right)^m.$$

We want that $N_u \geq 2^{118}$, and by simple computation it turns out that 2^{119} hash evaluations are sufficient with high probability. Using these values in Equation (33) yields a success probability of around 63 percent.

7.5.3 Direct Preimage Attack on Friday

The preimage attack we present in this section works with *one* block of FRIDAY, as shown in Figure 7.

The description of the intermediate states $x_1, \ldots, x_r$ yields the same system of equations as before; however, in contrast to the optimised attack on JARVIS described in Section 7.4.2, in the current preimage attack on FRIDAY

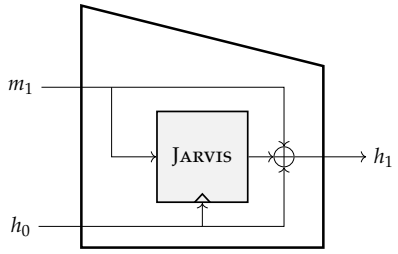


Figure 7: Preimage attack on FRIDAY using one message block.

the master key k_0 and thus all subsequent round keys $k_1, \ldots, k_r$ are known. As an effect, we do not need to express round keys as a rational function of k_0 anymore. For the sake of completeness, we give Equation (30) once more and note that the degree now decreases to 32 (from formerly 40). It holds that

$$D\left(\frac{1}{C(x_{i-1}) + k_{i-1}}\right) = E\left(\frac{1}{B(x_{i+1})} + k_i\right)$$

for $2 \leq i \leq r-1$. Moreover, an additional equation is needed to describe the structure of the Miyaguchi-Preneel compression function (see Figure 6), namely

$$B(x_1) \cdot (C(x_r) + k_r + h_1) = 1.$$

Again, we assume an even number of rounds r and work with intermediate states $x_2, x_4, \ldots, x_r$, which is why we need to apply the transformations D and E to cancel out the state x_1 in the above equation. Thus, eventually we have

$$D\left(\frac{1}{C(x_r) + k_r + h_1}\right) = E\left(\frac{1}{B(x_2)} + k_1\right). \tag{34}$$

Here, h_1 denotes the hash value for which we want to find a preimage m_1' such that

$$F(m_1', h_0) = h_1.$$

To obtain m_1' we solve for the intermediate state x_r and calculate

$$m_1' := C(x_r) + k_r + h_1 + h_0.$$

The value $h_0 = k_0$ can be regarded as the initialisation vector and is the zero element in $\mathbb{F}_{2^n}$. The above attack results in:

- $\frac{r}{2} - 1$ equations of degree 32 coming from Equations (30) when considering even intermediate states, and
- one equation of degree 32 coming from Equation (34),

in the $\frac{r}{2}$ variables $x_2, x_4, \ldots, x_r$. The number of equations is the same as the number of variables, and we can again use Equation (23) to estimate the degree of regularity. The complexities of the Gröbner basis computations are summarised in Table 4, where we pessimistically assume $\omega = 2.8$, but also give the complexities for $\omega = 2$ in parentheses.

7.6 Behaviour of the Attacks against Jarvis and Friday

Recall that our attack has three steps:

1. Set up an equation system and compute a Gröbner basis using, e.g., the F4 algorithm [Fau99], with cost $\mathcal{C}_{\text{GB}}$.

r	n_v	D_{reg}	Complexity in bits
6	3	94	48 (34)
8	4	125	65 (47)
10 (JARVIS-128)	5	156	83 (59)
12 (JARVIS-192)	6	187	101 (72)
14 (JARVIS-256)	7	218	118 (85)
16	8	249	136 (97)
18	9	280	154 (110)
20	10	311	172 (123)

Table 4: Complexity estimates for the Gröbner basis step in preimage attacks on FRIDAY using r-round JARVIS.

2. Perform a change of term ordering for the computed Gröbner basis using the FGLM algorithm [FGL+93], with cost $\mathcal{C}_{\text{FGLM}}$.
3. Solve the remaining univariate equation for the last variable using a polynomial factoring algorithm, substitute into other equations, with cost $\mathcal{C}_{\text{Sol}}$.

For the overall cost of the attack we have[3]:

$$\begin{aligned} \mathcal{C} &:= 2\,\mathcal{C}_{\text{GB}} + 2\,\mathcal{C}_{\text{FGLM}} + \mathcal{C}_{\text{Sol}}, \\ \mathcal{C} &:= 2\left(\binom{n_v + D}{D}^{\omega}\right) + 2\left(n_v \cdot {D_u}^3\right) + \left(D_u \log^2 D_u\right). \end{aligned}$$

We can estimate $\mathcal{C}_{\text{GB}}$ if we assume that our systems behave like regular sequences. For the $\mathcal{C}_{\text{FGLM}}$ and $\mathcal{C}_{\text{Sol}}$ we need to establish the degree D_u of the univariate polynomial recovered, for which however we do not have an estimate. We have therefore implemented our attacks on JARVIS and FRIDAY using Sage v8.6 [Ste+19] with Magma v2.20–5 [Mag] as the Gröbner basis engine. In particular, we implemented both the unoptimised and the optimised variants of the attacks from Sections 7.4.2 and 7.5.3.

We observed that our attacks performed significantly better in our experiments than predicted. On the one hand, our Gröbner basis computations reached significantly lower degrees D than the (theoretically) expected D_{reg}. Furthermore, the degrees of the univariate polynomials seem to grow as $\approx 2 \cdot 5^r$ (JARVIS) and as $\approx 2 \cdot 4^r$ (FRIDAY), respectively, suggesting the second and third steps of our attack are relatively cheap.

We therefore conclude that the complexities given in Tables 3 and 4 are conservative upper bounds for our attacks on JARVIS and FRIDAY. We summarise our findings in Table 5, and the source code of our attacks on MARVELLOUS is available on GitHub[4].

In Table 5, r denotes the number of rounds, D_{reg} is the expected degree of regularity under the assumption that the input system is regular, n_v is the number of variables, $2 \cdot \log_2 \binom{n_v + D_{\text{reg}}}{D_{\text{reg}}}$ is the expected bit security for $\omega = 2$ under the regularity assumption, D is the highest degree reached during the Gröbner basis computation, and $2 \cdot \log_2 \binom{n_v + D}{D}$ is the expected bit security for $\omega = 2$ based on our experiments. The degree of the recovered univariate polynomial used for solving the system is denoted as D_u.

[3] As suggested in Section 7.3.3, our attack proceeds by running steps 1 and 2 twice, and recovering the last variable via the GCD computation, thus reducing the complexity of step 3.

[4] https://github.com/IAIK/marvellous-attacks

r	n_v	D_{reg}	$2\log_2\binom{n_v+D_{\text{reg}}}{D_{\text{reg}}}$	D	$2\log_2\binom{n_v+D}{D}$	$D_u = \deg(\mathcal{I})$	Time
JARVIS (optimised)							
3	2	47	20	26	17	256	0.3s
4	3	67	31	40	27	1280	9.4s
5	3	86	34	40	27	6144	891.4s
6	4	106	45	41	34	28672	99989.0s
JARVIS (unoptimised)							
3	4	25	29	10	20	256	0.5s
4	5	33	38	11	24	1280	23.9s
5	6	41	47	13	29	6144	2559.8s
6	7	47	55	14	34	28672	358228.6s
FRIDAY							
3	2	39	19	32	18	128	3.6s
4	2	63	22	36	19	512	0.5s
5	3	70	32	36	26	2048	36.5s
6	3	94	34	48	29	8192	2095.2s

Table 5: Experimental results using Sage.

7.6.1 Comparison with MiMC

We note that the same attack strategy, namely a direct Gröbner basis computation to recover the secret key, also applies, in principle, to MiMC, as pointed out by [Ash19]. In particular, it is easy to construct a multivariate system of equations for MiMC with degree 3 that is already a Gröbner basis by introducing a new state variable per round[5]. This makes the first step of a Gröbner basis attack free.[6] However, then the change of ordering has to essentially undo the construction to recover a univariate polynomial of degree $D_u \approx 3^r$. Performing this step twice produces two such polynomials from which we can recover the key by applying the GCD algorithm with complexity $\tilde{\mathcal{O}}(3^r)$. In [AGR+16], the security analysis implicitly assumes that steps 1 and 2 of our attack are free by constructing the univariate polynomial directly and costing only the third and final step of computing the GCD.

The reason our Gröbner basis attacks are so effective against FRIDAY and JARVIS is that the particular operations used in the ciphers – finite field inversion and low-degree linearised polynomials – allow us to construct a polynomial system with a relatively small number of variables, which can in turn be efficiently solved using our three-step attack strategy. We have not been able to construct such amenable systems for MiMC.

7.7 Comparing the S-Boxes of Jarvis and the AES

The non-linear operation in JARVIS shows similarities with the AES S-box $S_{\text{AES}}(X)$. In particular, $S_{\text{AES}}(X)$ is the composition of an $\mathbb{F}_2$-affine function A_{AES} and the multiplicative inverse of the input in $\mathbb{F}_{2^8}$, i.e.

$$S_{\text{AES}}(X) = A_{\text{AES}}(X^{254}),$$

[5] This property was observed by Tomer Ashur and Alan Szepieniec and shared with us during personal communication.

[6] We note that this situation is somewhat analogous to the one described in [BPW06].

where

$$A_{\text{AES}}(X) = \texttt{0x8F} \cdot X^{128} + \texttt{0xB5} \cdot X^{64} + \texttt{0x01} \cdot X^{32} + \texttt{0xF4} \cdot X^{16} + \texttt{0x25} \cdot X^8 + \texttt{0xF9} \cdot X^4 + \texttt{0x09} \cdot X^2 + \texttt{0x05} \cdot X + \texttt{0x63}.$$

In JARVIS, we can also view the S-box as

$$S(X) = A(X^{254}),$$

where

$$A(X) = (C \circ B^{-1})(X)$$

and both B and C are of degree 4. In this section we show that A_{AES} *cannot* be split into

$$A_{\text{AES}}(X) = (\hat{C} \circ \hat{B}^{-1})(X),$$

with both $\hat{B}$ and $\hat{C}$ of low degree. To see this, first note that above decomposition implies

$$\hat{B}(X) = A_{\text{AES}}^{-1}(\hat{C}(X)),$$

where

$$A_{\text{AES}}^{-1}(X) = \texttt{0x6E} \cdot X^{128} + \texttt{0xDB} \cdot X^{64} + \texttt{0x59} \cdot X^{32} + \texttt{0x78} \cdot X^{16} + \texttt{0x5A} \cdot X^8 + \texttt{0x7F} \cdot X^4 + \texttt{0xFE} \cdot X^2 + \texttt{0x5} \cdot X + \texttt{0x5}$$

is the compositional inverse polynomial of A_{AES} satisfying the relation

$$A_{\text{AES}}^{-1}(A_{\text{AES}}(x)) = x,$$

for every $x \in \mathbb{F}_{2^8}$. Hence, to show that at least one of $\hat{B}, \hat{C}$ is of degree > 4, it suffices to compute $A_{\text{AES}}^{-1}(\hat{C})$ assuming a degree 4 for $\hat{C}$, and to show that then the corresponding $\hat{B}$ has degree > 4.

Remark. First of all, note that since A_{AES} has degree 128, it is always possible to find polynomials $\hat{C}$ and $\hat{B}$ of degree 8 such that the equality $A_{\text{AES}}(X) = \hat{C}(\hat{B}^{-1}(X))$ is satisfied. Indeed, if both $\hat{C}$ and $\hat{B}$ have degree 8, then each one of them have all monomials of degrees $1, 2, 4$ and 8. The equality $A_{\text{AES}}(X) = \hat{C}(\hat{B}^{-1}(X))$ is then satisfied if 8 equations (one for each monomial of A_{AES}) in 8 variables (both $\hat{C}$ and $\hat{B}$ have 4 monomials each) are satisfied. Hence, a random polynomial A_{AES} satisfies the equality $A_{\text{AES}}(x) = \hat{C}(\hat{B}^{-1}(x))$ with negligible probability if both $\hat{C}$ and $\hat{B}$ have degree at most 4.

Property of A_{AES}. Let us assume a degree-4 polynomial

$$\hat{C}(X) = \hat{c}_4 X^4 + \hat{c}_2 X^2 + \hat{c}_1 X + \hat{c}_0.$$

We can now write down $A_{\text{AES}}^{-1}(\hat{C}(X))$, which results in $\hat{B}(X)$. However, we want $\hat{B}$ to be of degree at most 4, so we set all coefficients for the degrees $8, 16, 32, 64, 128$ to 0. This results in a system of five equations in the three variables $\hat{c}_1, \hat{c}_2, \hat{c}_4$, given in Appendix 7.C. We tried to solve this system and confirmed that no solutions exist. Thus, the affine part of the AES S-box cannot be split into $\hat{C}(\hat{B}^{-1}(X))$ such that both $\hat{B}$ and $\hat{C}$ are of degree at most 4, whereas in JARVIS this is possible.

As a result, from this point of view, the main difference between AES and JARVIS/FRIDAY is that the linear polynomial used to construct the AES

S-box does not have the splitting property used in our attacks, while the same is not true for the case of JARVIS/FRIDAY. In this latter case, even if $B(C^{-1})$ has high degree, it depends only on 9 variables instead of $n+1$ as expected by a linearised polynomial of degree 2^n (where $n \geq 128$). Thus, a natural question to ask is what happens if we replace B and C with other polynomials of higher degree.

7.8 Conclusion and Future Work

We have demonstrated that JARVIS and FRIDAY are insecure against Gröbner basis attacks, mainly due to the algebraic properties of concatenating the finite field inversion with a function that is defined by composing two low-degree affine polynomials. In our attacks we modelled both designs as a system of polynomial equations in several variables. Additionally, we bridged equations over two rounds, with the effect of significantly reducing the number of variables needed to describe the designs.

Following our analysis, the area sees a dynamic development. Authors of JARVIS and FRIDAY have abandoned their design. Their new construction [ACG+19] is substantially different, although it still uses basic components which we were able to exploit in our analysis. Whether our particular method of bridging internal state equations can be applied to the new hash functions is subject to future work. A broader effort is currently underway to identify designs practically useful for a range of modern proof systems. A noteable competition compares three new designs (Marvelous [ACG+19], Poseidon/Starkad [GKR+21], and GMiMC [AGP+19]) with the more established MiMC.

Acknowledgements

We thank Tomer Ashur for fruitful discussions about JARVIS, FRIDAY, and a preliminary version of our analysis. The research described in this paper was supported by the Royal Society International Exchanges grant "Domain Specific Ciphers" (IES\R2\170211).

7.a Polynomials of Section 7.4.2

In Section 7.4.2, we search for monic affine polynomials D, E such that the equality

$$D(B) = E(C)$$

is satisfied, where B, C are monic affine polynomials of degree 4. In particular, given

$$B(X) = X^4 + b_2X^2 + b_1X + b_0 \quad \text{and} \quad C(X) = X^4 + c_2X^2 + c_1X + c_0$$

the goal is to find

$$D(X) = X^4 + d_2X^2 + d_1X^1 + d_0 \quad \text{and} \quad E(X) = X^4 + e_2X^2 + e_1X + e_0$$

such that $D(B) = E(C)$. By comparing the corresponding coefficients of $D(B)$ and $E(C)$, we obtain a system of 5 linear equations in the 6 variables $d_0, d_1, d_2, e_0, e_1, e_2$:

$$\begin{aligned} d_2 + e_2 &= b_2^4 + c_2^4, \\ d_1 + b_2^2 \cdot d_2 + e_1 + c_2^2 \cdot e_2 &= b_1^4 + c_1^4, \\ b_2 \cdot d_1 + b_1^2 \cdot d_2 + c_2 \cdot e_1 + c_1^2 \cdot e_2 &= 0, \\ b_1 \cdot d_1 + c_1 \cdot e_1 &= 0, \\ d_0 + b_0 \cdot d_1 + b_0^2 \cdot d_2 + e_0 + c_0 \cdot e_1 + c_0^2 \cdot e_2 &= b_0^4 + c_0^4. \end{aligned}$$

This system can be solved to recover D and E.

7.b Constants α_i, β_i, γ_i, and δ_i for the Round Keys

Each round key $k_{i+1} = \frac{1}{k_i} + c_i$ in JARVIS can be written as

$$k_{i+1} = \frac{\alpha_i \cdot k_0 + \beta_i}{\gamma_i \cdot k_0 + \delta_i},$$

where α_i, β_i, γ_i, and δ_i are constants. By simple computation, note that:

- $i = 0$:
$$k_1 = \frac{1}{k_0} + c_0 = \frac{c_0 k_0 + 1}{k_0},$$
and $\alpha_0 = c_0, \beta_0 = 1, \gamma_0 = 1, \delta_0 = 0$;
- $i = 1$:
$$k_2 = \frac{1}{k_1} + c_1 = \frac{(c_0 c_1 + 1)k_0 + c_1}{c_0 k_0 + 1},$$
and $\alpha_1 = 1 + c_0 c_1, \beta_1 = c_1, \gamma_1 = c_0, \delta_1 = 1$;
- $i = 2$:
$$k_3 = \frac{1}{k_2} + c_2 = \frac{(c_0 c_1 c_2 + c_0 + c_2)k_0 + c_1 c_2 + 1}{(c_0 c_1 + 1)k_0 + c_1},$$
and $\alpha_2 = c_0 c_1 c_2 + c_0 + c_2, \beta_2 = c_1 c_2 + 1, \gamma_2 = c_0 c_1 + 1, \delta_2 = c_1$;

and so on. Thus, we can derive recursive formulas to calculate the remaining values for generic $i \geq 0$:

$$\begin{aligned} \alpha_{i+1} &= \alpha_i \cdot c_{i+1} + \gamma_i, \\ \beta_{i+1} &= \beta_i \cdot c_{i+1} + \delta_i, \\ \gamma_{i+1} &= \alpha_i, \\ \delta_{i+1} &= \beta_i. \end{aligned}$$

7.c System of Equations from Section 7.7

The system of equations is constructed by symbolically computing

$$A_{\text{AES}}^{-1}(\hat{C}(x)),$$

as described in Section 7.7, and setting all coefficients for degrees 8, 16, 32, 64, 128 to 0. These are five possible degrees and the following equations are the sum of all coefficients belonging to each of these degrees:

$$\begin{aligned}
\texttt{0x5a} \cdot \hat{c}_1^8 + \texttt{0x7f} \cdot \hat{c}_2^4 + \texttt{0xfe} \cdot \hat{c}_4^2 &= 0, \\
\texttt{0x78} \cdot \hat{c}_1^{16} + \texttt{0x5a} \cdot \hat{c}_2^8 + \texttt{0x7f} \cdot \hat{c}_4^4 &= 0, \\
\texttt{0x59} \cdot \hat{c}_1^{32} + \texttt{0x78} \cdot \hat{c}_2^{16} + \texttt{0x5a} \cdot \hat{c}_4^8 &= 0, \\
\texttt{0xdb} \cdot \hat{c}_1^{64} + \texttt{0x59} \cdot \hat{c}_2^{32} + \texttt{0x78} \cdot \hat{c}_4^{16} &= 0, \\
\texttt{0x6e} \cdot \hat{c}_1^{128} + \texttt{0xdb} \cdot \hat{c}_2^{64} + \texttt{0x59} \cdot \hat{c}_4^{32} &= 0.
\end{aligned}$$

By practical tests we found that no (nontrivial) coefficients $\hat{c}_1, \hat{c}_2, \hat{c}_4$ satisfy all previous equalities, which means that there are no polynomials $\hat{B}$ and $\hat{C}$ both of degree 4 that satisfy $A_{\text{AES}}(X) = (\hat{C} \circ \hat{B}^{-1})(X)$.

8

INFLUENCE OF THE LINEAR LAYER ON THE ALGEBRAIC DEGREE IN SP-NETWORKS

Based on the peer-reviewed conference publication

[CGG+22] Carlos Cid, Lorenzo Grassi, Aldo Gunsing, Reinhard Lüftenegger, Christian Rechberger, and Markus Schofnegger. "Influence of the Linear Layer on the Algebraic Degree in SP-Networks." In: *IACR Transactions on Symmetric Cryptology* 2022.1 (2022), pp. 110–137. DOI: 10.46586/tosc.v2022.i1.110-137

Abstract We consider SPN schemes, i.e., schemes whose non-linear layer is defined as the parallel application of $t \geq 1$ independent S-Boxes over $\mathbb{F}_{2^n}$ and whose linear layer is defined by the multiplication with a $(n \cdot t) \times (n \cdot t)$ matrix over $\mathbb{F}_2$. Even if the algebraic representation of a scheme depends on all its components, upper bounds on the growth of the algebraic degree in the literature usually only consider the details of the non-linear layer. Hence a natural question arises: (how) do the details of the linear layer influence the growth of the algebraic degree? We show that the linear layer plays a crucial role in the growth of the algebraic degree and present a new upper bound on the algebraic degree in SP-networks. As main results, we prove that in the case of low-degree round functions with large S-Boxes: *(a)* an initial exponential growth of the algebraic degree can be followed by a linear growth until the maximum algebraic degree is reached; *(b)* the rate of the linear growth is proportional to the degree of the linear layer over $\mathbb{F}_{2^n}^t$. Besides providing a theoretical insight, our analysis is particularly relevant for assessing the security of cryptographic permutations designed to be competitive in applications like MPC, FHE, SNARKs, and STARKs, including permutations based on the Hades design strategy. We have verified our findings on small-scale instances and we have compared them against the currently best results in the literature, showing a substantial improvement of upper bounds on the algebraic degree in case of low-degree round functions with large S-Boxes.

Keywords Higher-Order Differential Cryptanalysis, Algebraic Degree, SPN, Linear Layer

8.1 Introduction

Most modern block ciphers and cryptographic permutations over $\mathbb{F}_2^N$, for $N = n \cdot t$, are based on the iteration of a round function. In many cases, the round function is composed of two main components, a non-linear layer S and a linear layer M (including the addition of round constants). The non-linear layer S is defined as the parallel application of t independent non-linear functions over $\mathbb{F}_2^n$. The linear layer M is defined via the multiplication with a $(n \cdot t) \times (n \cdot t)$ matrix over $\mathbb{F}_2$. This design strategy is called a Substitution-Permutation-Network (SPN).

The particular combination of these two building blocks, their details and the number of rounds are chosen to guarantee security against all possible

means of analysis present in the literature, while at the same time achieving good performance in the target applications. Regarding the security aspect, the analysis of symmetric schemes can be divided into statistical and algebraic cryptanalysis. Subsuming statistical analysis, we can identify all methods that exploit statistical properties of the analyzed scheme, including differential [BS91; BS93] and linear [Mat93] cryptanalysis, and all their variants, like truncated differential [Knu94a], impossible differential [Knu98; BBS99] and zero-correlation [BR14] analysis. In contrast, algebraic analysis exploits algebraic properties of the analyzed schemes such as degrees and/or the different algebraic representations. In this category, we include interpolation cryptanalysis [JK97], higher-order differential analysis [Lai94; Knu94a], cube attacks[DS09] and methods employing Groebner bases [Buc76]. While the influence of the linear layer on statistical analysis has been largely analyzed in the literature [DR01; DR02a; BDK+21], the same is not true for the case of algebraic analysis.

Influence of the Linear Layer on Statistical Analysis. For statistical analysis, the impact of the linear layer on the security against this means of analysis is well studied in the literature. If the linear layer of a scheme is defined by the multplication with a $t \times t$ matrix over $\mathbb{F}_{2^n}$, an upper bound of the probability of differential trails can be found by considering both the maximum differential probability of the involved S-Boxes (namely, the maximum probability that a non-zero input difference is mapped into an output difference) and the branch number of the linear layer (that is, the maximum number of active S-Boxes over two consecutive rounds). This is known as the wide-trail design strategy [DR01; DR02a]. Analogous results hold for the case of linear trails.

If the linear layer does not admit an equivalent representation as a $t \times t$ matrix over $\mathbb{F}_{2^n}$, statistical analysis that makes use of this alignment is frustrated after a few rounds, but, e.g., the wide trail design strategy does not apply anymore. In this scenario, differential/linear bounds are often obtained by computer-aided proofs.

Influence of the Linear Layer on Algebraic Analysis. Contrary to statistical analysis, the influence of the linear layer on the security against algebraic analysis is not well researched in the literature. Focusing on schemes over $\mathbb{F}_2^N$, let's consider, e.g., higher-order differential cryptanalysis [Lai94; Knu94a], probably one of the most powerful cryptanalytic methods for symmetric primitives over $\mathbb{F}_2^N$ with low-degree building blocks. Given an instance of a (keyed or keyless) cryptographic permutation $P : \mathbb{F}_2^N \to \mathbb{F}_2^N$, higher-order differential cryptanalysis exploits the fact that if the algebraic degree of P is strictly smaller than $N - 1$ then for any (proper) vector subspace $\mathcal{V} \subseteq \mathbb{F}_2^N$ with dimension strictly greater than the algebraic degree of P and for any $v \in \mathbb{F}_2^N$, we have $\bigoplus_{x \in \mathcal{V} \oplus v} P(x) = 0$. Since the same property does not, in general, hold for a permutation drawn at random, it is possible to distinguish a given (keyed or keyless) permutation from a random permutation. The idea was first introduced by Lai [Lai94], albeit without a concrete application. Knudsen [Knu94a] then used higher-order differentials to analyze low-degree ciphers which were deemed secure against standard differential cryptanalysis.

The crucial problem in higher-order differential distinguishers against iterated constructions is the analysis of the growth of the algebraic degree. Currently, the best generic upper bound for the growth of the algebraic

degree is given in [BCC11], where authors upper bound the algebraic degree of the composition of two functions over $\mathbb{F}_{2^n}^t$. More recently, for the particular case in which the round function is defined as a low-degree polynomial over $\mathbb{F}_{2^N}$, a more accurate estimate on the minimum number of rounds to reach maximum algebraic degree has been proposed in [EGL+20]. However, in all these cases, the details of the linear layer are not taken into account.

The Scope of our Results. We pick up this problem, and we show how the details of the linear layer influence the growth of the algebraic degree in SPN schemes. As main results

- we generalize the upper bound given in [EGL+20] (only valid for Even–Mansour schemes, a subset of all SPN schemes) to the whole class of SPN schemes and prove a linear upper bound on the growth of the degree that improves the exponential one proposed in [BCC11];
- we analyze the impact of the linear layer on the growth of the degree. That is, we prove that the rate of the linear growth is proportional to the degree of the linear layer when written as a linear function over $\mathbb{F}_{2^n}^t$.

We point out that this is not only of theoretical interest. Indeed, motivated by new applications such as secure Multi-Party Computation (MPC), Fully Homomorphic Encryption (FHE) and Zero-Knowledge proofs (ZKP), the need for symmetric encryption schemes with a simple natural algebraic description has become ever more apparent. This is an active area of research, and many dedicated symmetric encryption schemes that aim for simple arithmetization or directly aim for a small number of multiplications in $\mathbb{F}_{2^n}$ or $\mathbb{F}_p$, for large n and prime p (usually, $2^n, p \approx 2^{128}$), have recently been proposed in the literature. They include permutations, block ciphers, and hash functions such as MiMC [AGR+16; GRR+16], GMiMC [AGP+19], HadesMiMC [GLR+20] (and its hash variant Poseidon [GKR+21]), Jarvis & Friday [AD18], Vision & Rescue [AAB+20], and Ciminion [DGG+21]. Many of these proposed schemes use "algebraically simple" S-Boxes, e.g., based on a power mapping $x \mapsto x^d$ for a small odd integer $d \geq 3$. In these schemes, our bounds are most competitive against other state-of-the-art bounds and, furthermore, they help to establish a more accurate estimate for the number of rounds that guarantee security in future MPC-/FHE-/ZKP-friendly designs.

Nomenclature. Since we do not make any assumption about the round-keys, our results equally apply to keyed and keyless permutations. Thus in this paper we refer to both by using the term "schemes". In this nomenclature, e.g., an *SPN scheme* is a family of permutations built from an SPN construction parametrized by secret keys or publicly known constants.

8.1.1 Related Work in the Literature

We focus on the case of *iterated* schemes, that is, schemes consisting of several iterations of the same round function. Algebraic analysis, like interpolation or higher-order differential and integral distinguishers, is based on bounding the (algebraic) degree of the analyzed scheme, which is in general a difficult task. Here, we recall the main results in the literature that focus on this problem. For a more detailed discussion and comparison of different approaches to bounding the algebraic degree we refer to [CXZ+21].

Theoretical Bounds on the Algebraic Degree

A naive bound for the algebraic degree of the composition of two functions $F, G : \mathbb{F}_2^N \to \mathbb{F}_2^N$ is given by $\deg(G \circ F) \leq \deg(G) \cdot \deg(F)$. If iterated, this bound leads to an exponential bound on the algebraic degree for the composition of more than two functions and a first estimate about the minimum number of rounds to reach maximum algebraic degree in SPN schemes. For an SPN scheme defined over $\mathbb{F}_{2^n}^t$ with S-Box layer of algebraic degree δ, it follows that at least

$$\log_\delta(n \cdot t - 1) \approx \log_\delta(n) + \log_\delta(t)$$

rounds are required to reach maximum degree (note that the affine layer does not increase the algebraic degree).

Result by Boura, Canteaut and De Cannière [BCC11]. The naive exponential bound, however, does not reflect the real growth of the algebraic degree when considering iterated schemes, and the problem of estimating the growth of the algebraic degree has therefore been studied in the literature. After the initial work of Canteaut and Videau [CV02], a tighter upper bound was presented by Boura, Canteaut, and De Cannière in [BCC11]. In there, the authors deduce a new bound for the algebraic degree of iterated permutations for SPN schemes over $\mathbb{F}_{2^n}^t$, which includes functions that have a number of $t \geq 1$ balanced S-Boxes over $\mathbb{F}_{2^n}$ as their non-linear layer. The bound in [BCC11] only relies on the algebraic degree of the S-Box, and no assumption on the linear layer is made. To apply the result presented in [BCC11], one has to determine a particular parameter γ, that depends on the details of the S-Box. As we discuss in Section 8.4.1, for an S-Box over $\mathbb{F}_{2^n}$ the cost for computing γ is exponential in n. This means, for large S-Boxes (e.g., $n \geq 64$) it is infeasible to determine γ computationally and a further study of the analyzed scheme is necessary. However, theoretically bounding γ is in general a difficult task. Apart from the bound of Boura, Canteaut and De Cannière, in a follow-up work Boura and Canteaut studied the influence of F^{-1} on the algebraic degree of $\deg(G \circ F)$ [BC13]. As main result, they discuss how the algebraic degrees of F^{-1} and F affect each other, which subsequently allows them to bound the algebraic degree of $G \circ F$ by means of the degrees of G and F^{-1}.

Result by Carlet [Car20]. More recently, Carlet [Car20] presented a bound on the algebraic degree of $G \circ F$ by working with the indicators of the graphs $\mathcal{G}_F$ and $\mathcal{G}_G$ (where $\mathcal{G}_F = \{(x, F(x)) : x \in \mathbb{F}_2^N\}$). In this work, Carlet bounds the algebraic degree of $G \circ F$ via the degree of G and the degree of the indicator function of $\mathcal{G}_F$. However, the bounds in [Car20] require evaluating the degree of large quantities of products of coordinate functions (see [Car20, Theorem 5]) and, to the best of our knowledge, it is unclear if the bounds in [Car20] practically improve upon the ones in [BCC11] if the function F in $G \circ F$ is bijective. In this scenario, the deduced bound on the algebraic degree of $G \circ F$ is essentially the same as in [BC13] (see discussion after Corollary 5 in [Car20]).

Division Property. A generalization of integral and higher-order differential distinguishers is the division property [Tod15], proposed by Todo at Eurocrypt 2015. Given $u = (u_0, u_1, ..., u_{n-1}) \in \mathbb{F}_2^n$, let $x^u := \prod_{i=0}^{n-1} x_i^{u_i}$ for each $x \in \mathbb{F}_2^n$. The division property generalizes integral cryptanalysis and

Parameter	Explanation
$\mathbb{F}_{2^n}$	Finite field with 2^n elements
$\mathbb{F}_{2^n}^t$	t-fold cartesian product of $\mathbb{F}_{2^n}$
n	S-Box size in bits
t	Number of words in the SPN
$N := n \cdot t$	State size in bits
d	Word-level degree (over $\mathbb{F}_{2^n}$) of the S-Boxes
δ	Algebraic degree (over $\mathbb{F}_2$) of the S-Boxes
$l := 2^{l'}$	Degree of the linear layer (over $\mathbb{F}_{2^n}$)
	Word-level degree (over $\mathbb{F}_{2^n}$) of the round function

Table 6: Nomenclature and parameters in our results for SPN schemes over $\mathbb{F}_{2^n}^t$

higher-order differential distinguishers in the sense that it is interested in the sum of this quantity taken over all vectors of $X \subseteq \mathbb{F}_2^n$. To the best of our knowledge and at the current state of the art, the division property can only provide useful bounds on the algebraic degree for *small* n. Indeed, currently it is infeasible to apply the two-/three-subset bit-based division property [TM16; FTI+17; WHT+18; HSW+20] to large S-Boxes (i.e., of size bigger than 12 bits to the best of our knowledge). Hence, such a tool does not seem to be useful in the case of schemes defined over $\mathbb{F}_{2^n}^t$ for large n (as targeted in this paper), and a theoretical estimation is hence crucial.

Algebraic Degree in MiMC-Like Schemes. MiMC [AGR+16; GRR+16] is a scheme natively defined over $\mathbb{F}_{2^N}$, where the S-Box is given by the cube function $x \mapsto x^3$. Only recently a new upper bound on the algebraic degree of MiMC-like schemes (that is, of schemes defined over $\mathbb{F}_{2^N}$ via a round function of degree $d \geq 3$) has been proposed in [EGL+20] at Asiacrypt 2020. More precisely, the authors show that when the round function can be described as a low-degree polynomial function over $\mathbb{F}_{2^N}$ of degree at most d, the algebraic degree $\delta(r)$ of r iterations of the round function grows linearly with the number of rounds, i.e., $\delta(r) \leq \log_2(d^r + 1)$. This observation implies that at least $\log_d(2^{N-1} - 1)$ rounds are required for reaching maximum algebraic degree. As a concrete application, [EGL+20] shows that the number of rounds in MiMC needs to be increased by several percent to resist all known cryptanalysis. Nevertheless, the authors of [EGL+20] do not provide any statements about how to generalize their findings to SPN schemes.

8.1.2 Our Contribution

As main contribution, we present a new theoretical upper bound on the algebraic degree for SPN schemes over $\mathbb{F}_{2^n}^t$ in Theorem 17. In more detail, we consider SPN schemes over $\mathbb{F}_{2^n}^t$ for $n \geq 3$ and $t \geq 2$, where

- the S-Boxes are defined via invertible non-linear polynomial functions over $\mathbb{F}_{2^n}$ of univariate degree $d \geq 3$ and algebraic degree $\delta \geq 2$;
- the linear layer is defined as the multiplication with an invertible matrix in $\mathbb{F}_2^{n \cdot t \times n \cdot t}$. We denote by $l = 2^{l'}$ the degree of the corresponding function over $\mathbb{F}_{2^n}$.

In Section 8.2.2 we give more details about the definition of an SPN scheme and the involved degrees δ, d, l and . As a quick reference, Table 6 provides a more comprehensive overview about the parameters in our results. In

Theorem 17 we prove that the algebraic degree $\delta(r)$ after r rounds is upper-bounded by

$$\delta(r) \leq \begin{cases} \delta^r & \text{if } r \leq R_{\text{exp}} = 1 + \lfloor \log_\delta(t) \rfloor, \\ t \cdot \log_2\left(\frac{r-1 \cdot d}{t} + 1\right) & \text{if } R_{\text{exp}} < r \leq R_{\text{SPN}}. \end{cases} \tag{35}$$

It follows that at least

$$R_{\text{SPN}} = 1 + \lceil \log\left(t \cdot (2^n - 1) - 2^{n-1}\right) - \log(d) \rceil \approx \log\left(t \cdot (2^n - 1)\right)$$

rounds are necessary to reach maximum algebraic degree $n \cdot t - 1$, see Section 8.3.1. Our results have been practically verified on small-scale schemes. Section 8.5 is devoted to a more detailed discussion of our practical experiments. Moreover, our results match the ones given in [EGL+20] for the particular case $t = 1$.

Comparison with Related Work. As discussed above, there are two possible approaches for estimating the growth of the algebraic degree in SPN schemes: theoretical bounds, like the one by Boura, Canteaut and De Cannière [BCC11] and tool-based bounds, like the division property. However, both approaches have inherent limitations when applied to SPN schemes defined over $\mathbb{F}_{2^n}^t$ for large n (as targeted in this paper and important for MPC-/FHE-/ZKP-friendly schemes): in the first approach, the degree of the S-Box over $\mathbb{F}_{2^n}$ and the alignment of the scheme (hence, the degree of the linear layer over $\mathbb{F}_{2^n}$) are not taken into account. While this could be an advantage in the sense that such results apply to a large class of schemes, the resulting estimation of the growth of the algebraic degree is far from being optimal when applied to schemes over $\mathbb{F}_{2^n}^t$ with large and low-degree S-Boxes; in the second approach, the tools cannot tackle large S-Boxes (i.e., $n \geq 12$). Our new results include both scenarios.

A concrete comparison between our new bound on the algebraic degree and the one proposed in [BCC11] for an SPN scheme over $\mathbb{F}_{2^{33}}^8$ with cube S-Box $S(x) = x^3$ for several values of l is presented in Fig. 8.

8.2 Preliminaries

In this section, we recall the most important results about polynomial representations of Boolean functions and we recall the definition of SPN and iterated Even–Mansour schemes. We also introduce the classification of *weak-arranged* and *strong-arranged* SPN schemes.

8.2.1 Polynomial Representations over Binary Extension Fields

We denote addition (and subtraction) in binary extension fields and polynomial rings over binary extension fields by the symbol $\oplus$. For $n, t \in \mathbb{N}$, every function $F : \mathbb{F}_{2^n}^t \to \mathbb{F}_{2^n}$ can be uniquely represented by a polynomial over $\mathbb{F}_{2^n}$ in t variables with maximum degree $2^n - 1$ in each variable, i.e., as

$$F(X_1, \ldots, X_t) = \bigoplus_{v=(v_1,\ldots,v_t) \in \{0,1,\ldots,2^n-1\}^t} \varphi(v) \cdot X_1^{v_1} \cdot \ldots \cdot X_t^{v_t}, \tag{36}$$

for certain $\varphi(v) \in \mathbb{F}_{2^n}$. We refer to this representation as the *word-level representation*. At the same time, the function F can be written as an n-tuple $(F_1, \ldots, F_n)$ of functions $F_i : \mathbb{F}_2^N \to \mathbb{F}_2$ and thus admits a unique

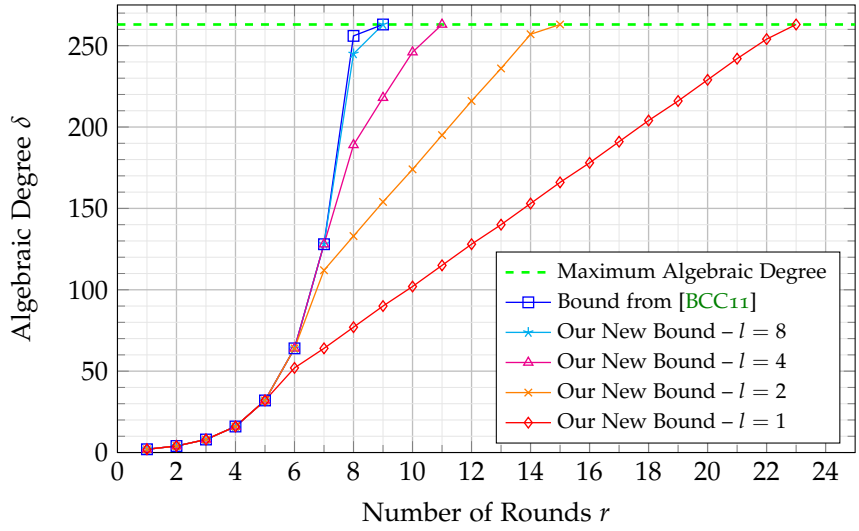


Figure 8: Comparison between our new bound and the one proposed in [BCC11] for the case of an SPN scheme instantiated over $(\mathbb{F}_{2^{33}})^8$ with a cube S-Box $S(x) = x^3$ for several values of l (where $n = 33$, $t = 8$, $d = 3, \delta = 2$ and $= d \cdot l = 3 \cdot l$, $\gamma = (n+1)/2 = 17$). γ is a constant for the bound in [BCC11] that depends on the details of the S-Box function S.

representation as an n-tuple $(F_1, \ldots, F_n)$ of polynomials over $\mathbb{F}_2$ in $N := n \cdot t$ variables with maximum degree 1 in each variable. Here, F_i takes the form

$$F_i(Y_1, \ldots, Y_N) = \bigoplus_{u=(u_1,\ldots,u_N)\in\{0,1\}^N} \rho_i(u) \cdot Y_1^{u_1} \cdot \ldots \cdot Y_N^{u_N}, \tag{37}$$

where the coefficients $\rho_i(u) \in \mathbb{F}_2$ can be computed by the *Moebius transform* with a time complexity of $\mathcal{O}(N \cdot 2^N)$ additive operations. We call this alternative description the *bit-level representation* of F. Combining Equations (37), for $1 \leq i \leq n$, into a single polynomial representation leads to a description of F as a single polynomial in $N = n \cdot t$ variables, but now with coefficients in $\mathbb{F}_2^n$, instead of $\mathbb{F}_2$.

Whenever we refer to the degree of a single variable in F (or F_i), we shall speak of the *univariate degree*. In contrast, the degree of F (or F_i) as a multivariate polynomial shall be called its *multivariate degree*, or just its *degree*.

We denote functions $F : \mathbb{F}_2^n \to \mathbb{F}_2$ as *Boolean functions* and hence functions of the form $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$, for $n \in \mathbb{N}$, as *vectorial Boolean functions*. We only work with vectorial Boolean functions where $n = m$. The unique polynomial representation of a Boolean function is called its *algebraic normal form* (ANF), which we emphasize with the following definition.

Definition 36. Let $F : \mathbb{F}_2^n \to \mathbb{F}_2$ be a Boolean function. The *algebraic normal form* (ANF) of F is the unique representation as a polynomial over $\mathbb{F}_2$ in n variables and with maximum univariate degree 1, as given in Eq. (37). The *algebraic degree* $\delta(F)$ of F is the degree of this representation as a multivariate polynomial over $\mathbb{F}_2$.

When the function F is clear from the context, we also write δ instead of $\delta(F)$. If $G : \mathbb{F}_2^n \to \mathbb{F}_2^n$ is a vectorial Boolean function and $(G_1, \ldots, G_n)$ is its representation as an n-tuple of multivariate polynomials over $\mathbb{F}_2$, then its algebraic degree $\delta(G)$ is defined as the maximal algebraic degree of its

coordinate functions G_i, i.e., as $\delta(G) = \max_{1 \le i \le n} \delta(G_i)$. The link between the algebraic degree and the univariate degree of a vectorial Boolean function is well-known, e.g., it is established in [CCZ98, Sect. 2.2]: due to the isomorphism of $\mathbb{F}_2$-vector spaces $\mathbb{F}_{2^n} \cong \mathbb{F}_2^n$, every function over $\mathbb{F}_2^n$ can be considered as a function over $\mathbb{F}_{2^n}$ and thus admits a representation as a univariate polynomial over $\mathbb{F}_{2^n}$. Hence, the algebraic degree of a vectorial Boolean function can be computed from its univariate representation. Eq. (38) makes this link explicit: Let $F : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ be a function over $\mathbb{F}_{2^n}$ and let $F(X) = \sum_{i=0}^{2^n-1} \varphi_i \cdot X^i$ denote the corresponding univariate polynomial description over $\mathbb{F}_{2^n}$. The algebraic degree $\delta(F)$ of F as a vectorial Boolean function is the maximum over all Hamming weights[1] of exponents of non-vanishing monomials, that is

$$\delta(F) = \max_{0 \le i \le 2^n-1} \{\mathrm{hw}(i) \mid \varphi_i \neq 0\}. \tag{38}$$

Lastly, we recall that the algebraic degree of an invertible function F over $\mathbb{F}_2^n$ is at most $n-1$, while the univariate polynomial representation of F over $\mathbb{F}_{2^n}$ has degree at most $2^n - 2$.

8.2.2 SPN Schemes

Here we recall the concept of SPN schemes, and we fix the notation used in the rest of the article. Let $E_k^r : \mathbb{F}_{2^n}^t \to \mathbb{F}_{2^n}^t$ denote the application of r rounds of an SPN scheme under a fixed (secret or publicly known) key $k \in \mathbb{F}_{2^n}^t$ with $n \ge 3$, $t \ge 2$, and $N := n \cdot t$. For every $x = (x_1, \ldots, x_t) \in \mathbb{F}_{2^n}^t$ we write

$$E_k^r(x) := (F_r \circ \cdots \circ F_1)(x \oplus k_0), \tag{39}$$

where each $F_i : \mathbb{F}_{2^n}^t \to \mathbb{F}_{2^n}^t$ is defined as $F_i(x) := R(x) \oplus k_i$. The subkeys $k_0, \ldots, k_r \in \mathbb{F}_{2^n}^t$ may be derived from the master key $k \in \mathbb{F}_{2^n}^t$ by means of a key schedule, or they may just as well be randomly chosen elements. Here, R denotes the composition of the S-Box and the linear layer, i.e., we have $R : \mathbb{F}_{2^n}^t \to \mathbb{F}_{2^n}^t$ with

$$R(x) := (M \circ S)(x) := M(S_1(x_1), \ldots, S_t(x_t)), \tag{40}$$

where all $S_i : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ are assumed to be invertible non-linear polynomial S-Boxes of degree $d \ge 3$ defined as

$$S_i(x) := \bigoplus_{j=0}^{d} c_j^{(i)} \cdot x^j, \tag{41}$$

for $c_j^{(i)} \in \mathbb{F}_{2^n}$ and $c_d^{(i)} \neq 0$. Finally, M denotes an invertible linear layer $M : \mathbb{F}_2^{n \cdot t} \to \mathbb{F}_2^{n \cdot t}$ defined by the multiplication with an invertible $(n \cdot t) \times (n \cdot t)$ matrix M with coefficients in $\mathbb{F}_2$. We remark, every $(n \cdot t) \times (n \cdot t)$ matrix M over $\mathbb{F}_2$ gives rise to an $\mathbb{F}_{2^n}$-linear function over $\mathbb{F}_{2^n}^t$. Moreover, every $\mathbb{F}_{2^n}$-linear function over $\mathbb{F}_{2^n}^t$ can be written as a function

$$M(x) = (M_1(x), M_2(x), \ldots, M_t(x)),$$

where $M_i : \mathbb{F}_{2^n}^t \to \mathbb{F}_{2^n}$, for $i \in \{1, 2, \ldots, t\}$, is a function of the form

$$M_i(x) = \bigoplus_{j=1}^{t} M_{i,j}(x_j) = \bigoplus_{h=0}^{l'} M_{i,j;h} \cdot x_j^{2^h}, \tag{42}$$

[1] Given $x = \sum_{i=0}^{s} x_i \cdot 2^i \in \mathbb{N}$, for $x_i \in \{0, 1\}$, then $\mathrm{hw}(x) = \sum_{i=0}^{s} x_i$.

with $M_{i,j;h} \in \mathbb{F}_{2^n}$ for each i, j, h. In other words, each $M_{i,j}$ is a linearized polynomial over $\mathbb{F}_{2^n}$ with respect to the variable x_j, and M_i is a sum of linearized polynomials over $\mathbb{F}_{2^n}$. In the following, we denote by $l := 2^{l'}$ the degree of M as a function over $\mathbb{F}_{2^n}^t$, i.e.,

$$l := \deg M := \max_{1 \le i \le t} \deg(M_i) = \max_{1 \le i,j \le t} \deg(M_{i,j}),$$

and by the degree of the round function satisfying $2^\delta - 1 \le \ := \min\{d \cdot l, 2^n - 2\}$.

We always assume that the linear layer M ensures *full diffusion after a finite number of rounds*, in the sense that there exists an $r \in \mathbb{N}$ such that every output word after r rounds depends on every input word $x_1, \ldots, x_t$. E.g., the smallest integer r that satisfies the previous condition for an MDS matrix is 1, for the AES MixLayer it is 2, while it does not exist for a diagonal matrix. We refer to [BJK+16] for a more detailed analysis of this concept. A particular subclass of SPN schemes are *iterated Even–Mansour schemes*. An iterated Even–Mansour (EM) scheme is an SPN scheme with only one word, i.e., with $t = 1$.

Under above definition, examples of SPN schemes include SHARK [RDP+96], AES [DR02b] and AES-like schemes in general, SHA-3/Keccak [BDP+11; BDP+13], Present [BKL+07], MiMC [AGR+16], LowMC [ARS+15a], and so on. Examples of non-SPN schemes include Feistel and Lai-Massey [LM90] schemes.

Classification: Strong-Arranged vs. Weak-Arranged SPN Schemes

We recall that for each $n, t \ge 1$, every matrix in $\mathbb{F}_{2^n}^{t \times t}$ admits an equivalent representation as a matrix in $\mathbb{F}_2^{n \cdot t \times n \cdot t}$, while the opposite does not hold in general. Let us introduce the following definition.

Definition 37. Let $t \ge 2$ and let $n \ge 3$, and let $M : \mathbb{F}_{2^n}^t \to \mathbb{F}_{2^n}^t$ be an invertible $\mathbb{F}_{2^n}$-linear function, represented as in Eq. (42). We say that M is (n,t)-*reducible* if there exist invertible $\mathbb{F}_{2^n}$-linear functions $M', L_1, L_2 : \mathbb{F}_{2^n}^t \to \mathbb{F}_{2^n}^t$ with $L_1, L_2 \ne M$, $\deg(M') < \deg(M)$ such that for $i = 1, 2$ it holds

$$L_i(x_1, \ldots, x_t) = (L_{i,1}(x_1), \ldots, L_{i,t}(x_t))$$

and

$$M = L_1 \circ M' \circ L_2. \tag{43}$$

We note, $\deg(L_1), \deg(L_2)$ are the degrees of L_1, L_2 when represented as in Eq. (42). If M is not (n,t)-reducible, we call it (n,t)-*irreducible*.

With the requirement $\deg(M') < \deg(M)$ we want to exclude trivial decompositions with $M' = L_1^{-1} \circ M \circ L_2^{-1}$, for any linear functions $L_1, L_2 : \mathbb{F}_{2^n}^t \to \mathbb{F}_{2^n}^t$. The same remark applies for the condition $L_1, L_2 \ne M$. Thereby, we exclude decompositions with $L_1 = M$ and $M' = \mathrm{Id}$ (Id being the identity function). We often just say that M is (ir)reducible instead of (n,t)-(ir)reducible, the context will provide enough clarification. Every SPN scheme admits an equivalent representation in which the defining matrix M for the linear layer is irreducible. Indeed, if this is not the case, it is sufficient to incorporate L_1 and L_2 from Eq. (43) into the non-linear layer S, that is

$$S \leftarrow L_2 \circ S \circ L_1, \tag{44}$$

and to adjust the round constants. We point out that this procedure may change the degrees d and l, but *not* the degree of the round function.

As a concrete example, consider the AES. Its S-Box over $\mathbb{F}_{2^8}$ is defined as

$$x \mapsto c + \hat{L} \circ x^{-1} = c + \hat{L} \circ (x^{127})^2,$$

for a certain linear function $\hat{L}$ over $\mathbb{F}_{2^8}$ of degree strictly bigger than 1. In the equivalent representation in which $\hat{L}$ and $x \mapsto x^2$ would be incorporated in the linear layer of AES (and so the AES S-Box would be $x \mapsto x^{127}$ over $\mathbb{F}_{2^8}$), the obtained linear layer would not be irreducible anymore with respect to the definition just given. Motivated by above discussion, we can assume that the linear layer M in an SPN scheme over $\mathbb{F}_{2^n}^t$ is (n,t)-irreducible.

Definition 38. Let $E^r : \mathbb{F}_{2^n}^t \to \mathbb{F}_{2^n}^t$ be an r-round SPN scheme with (n,t)-irreducible linear layer M (otherwise, consider an equivalent representation of E^r in which M is irreducible). The SPN scheme is called *strong-arranged* if the linear layer M has degree 1 over $\mathbb{F}_{2^n}^t$; *weak-arranged* otherwise.

Among the previous examples, AES, MiMC, HadesMiMC, and Vision are strong-arranged SPNs, while Keccak, Present and LowMC are weak-arranged SPNs.

On the Degree of the Linearized Polynomial. Given a matrix $M \in \mathbb{F}_2^{(n\cdot t)\times(n\cdot t)}$, the naive way to find its polynomial representation over $\mathbb{F}_{2^n}$ is by interpolation. The polynomial $M_{i,j}$ contains only n different monomials (see Eq. (42)). Hence, $t \cdot n + 1$ input/output pairs suffice to recover the polynomial representation of each M_i, and thus M. Moreover, given the polynomial representation of an $\mathbb{F}_{2^n}$-linear function over $\mathbb{F}_{2^n}^t$ (as in Eq. (42)), the simplest possible way to check if it is invertible or not is by finding the corresponding matrix over $\mathbb{F}_2^{(n\cdot t)\times(n\cdot t)}$, and check if its determinant is non-zero.

8.3 Growth of the Algebraic Degree in SPN Schemes

In this section we prove a new upper bound on the growth of the algebraic degree in SPN schemes. Our proof proceeds analogously for SPN-derived block ciphers and permutations, respectively, by assuming fixed and publicly known constants in the latter case and fixed secret keys in the former one.

8.3.1 Minimum Number of Rounds for Preventing Higher-Order Differential Distinguishers

Here, we provide a minimum number of rounds to reach maximum algebraic degree in SPN schemes. We show that this number matches the minimum number of rounds needed to provide security against the interpolation analysis [JK97].

Proposition 24. *Let $n \geq 3$. Consider r rounds of an SPN scheme E_k^r over $\mathbb{F}_{2^n}^t$ as defined in Eq. (39), where $l = 2^{l'} \geq 1$ is the degree of the linear layer and with the additional assumption that all S-Boxes $S_1, \ldots, S_t$ are defined via non-linear polynomial functions with equal univariate degree $d \geq 3$. Let be the degree of the round function.*

A lower bound on the number of rounds to prevent higher-order differential distinguishers is given by

$$\mathcal{R}_{\text{SPN}} := 1 + \lceil \log\left(t \cdot (2^n - 1) - 2^{n-1}\right) - \log(d) \rceil, \tag{45}$$

independent of the (secret or publicly known) key k.

Note that

$$\mathcal{R}_{\text{SPN}} \approx \log(2^n - 1) + \log(t), \tag{46}$$

especially for $t, n \gg 1$ and small $d \geq 3$ (where $\log(d) = 1$ if $l = 1$ and $0 < \log(d) < 1$ otherwise).

Proof. To reach maximum algebraic degree $n \cdot t - 1$ the polynomial representation of E_k^r over $\mathbb{F}_{2^n}$ must contain a monomial with algebraic degree n in $t - 1$ variables and algebraic degree $n - 1$ in one variable. This happens if E_k^r contains a word-level monomial with univariate degree $2^n - 1$ in $t - 1$ variables and univariate degree $2^{n-1} - 1$ in one variable. Since the multivariate degree of E_k^r after $r \geq 1$ rounds is upper bounded by $^{r-1} \cdot d$ (we note, the final linear layer does not affect the algebraic degree), we obtain

$$^{r-1} \cdot d \geq (t-1) \cdot (2^n - 1) + 2^{n-1} - 1 = t \cdot (2^n - 1) - 2^{n-1}$$

as a necessary condition on the number of rounds to reach maximum algebraic degree $n \cdot t - 1$. Rearranging for r yields

$$r \geq 1 + \log\left(t \cdot (2^n - 1) - 2^{n-1}\right) - \log(d).$$

□

8.3.2 Algebraic Degree of SPN Schemes

As main result of this paper, we prove the following upper bound on the growth of the degree for SPN schemes.

Theorem 17. *Let $n \geq 3$ and $t \geq 1$. Consider r rounds of an SPN scheme E_k^r over $\mathbb{F}_{2^n}^t$ as defined in Eq. (39), where $l = 2^{l'} \geq 1$ is the degree of the linear layer and with the additional assumption that all S-Boxes $S_1, \ldots, S_t$ are defined via the same invertible non-linear function S of univariate degree $d \geq 3$ and algebraic degree $\delta \geq 2$. Let be the degree of the round function.*

Let $R_{exp} := 1 + \lfloor \log_\delta(t) \rfloor$. Then, the algebraic degree of E_k^r after r rounds, denoted by $\delta(r)$, is upper-bounded by

$$\delta(r) \leq \begin{cases} \delta^r & \text{if } r \leq R_{exp}\,, \\ \min\left\{\delta^r,\ t \cdot \log_2\left(\frac{r-1 \cdot d}{t} + 1\right)\right\} & \text{if } r > R_{exp}, \end{cases} \tag{47}$$

independent of the (secret or publicly known) key k and until the maximum algebraic degree $n \cdot t - 1$ is reached.

This means that after an initial exponential growth for the first $R_{\text{exp}} := 1 + \lfloor \log_\delta(t) \rfloor$ rounds, the growth of the degree is upper bounded by a linear growth of the form

$$t \cdot \log_2\left(\frac{r-1 \cdot d}{t} + 1\right) \approx r \cdot t \cdot \log_2() + t \cdot \log_2\left(\frac{d}{\cdot\, t}\right),$$

where the linear rate $t \cdot \log_2()$ is proportional to the number of words t and to the degree of the round function, which is related to the degrees d and l of the S-Boxes and of the linear layer over $\mathbb{F}_{2^n}$.

Idea of the proof. The roadmap for the proof of Theorem 17 reads as follows:

1. Lemma 7 makes a statement about which monomials can occur in the polynomial representation of the encryption function;
2. In Lemma 8 we prove that the algebraic degree grows as fast as δ^r in the first $R_{\text{exp}} := 1 + \lfloor \log_\delta(t) \rfloor$ rounds; this shows that the naive exponential bound can indeed be achieved;
3. Lemma 9 provides the linear growth for the latter rounds by involving the logarithmic function instead of the hamming weights, resulting in the bound $\delta(r) \leq t \cdot \log_2\left(\frac{r-1 \cdot d}{t} + 1\right)$.

8.3.3 Proof of Theorem 17

About the (Initial) Exponential Growth

Lemma 7. *Let $t \geq 1$ and let $d' \geq 3$ be an integer and let $d' = \sum_{i=1}^{\delta} 2^{d_i}$ be the base-2 expansion of d for certain $d_i \in \mathbb{N}$. Given a polynomial $P = \bigoplus_{i \in \{1,\ldots,u\}} c_i \cdot m_i \in \mathbb{F}_{2^n}[X_1, \ldots, X_t]$ that contains the monomials $m_1, m_2, \ldots, m_u \in \mathbb{F}_{2^n}[X_1, \ldots, X_t]$ for a certain $u \geq 1$, the monomials in $P^{d'}$ are of the form*

$$m_{i_1}^{2^{d_1}} \cdot m_{i_2}^{2^{d_2}} \cdot \ldots \cdot m_{i_\delta}^{2^{d_\delta}} \tag{48}$$

where $i_1, i_2, \ldots, i_\delta \in \{1, 2, \ldots, u\}$.

Proof. We obtain

$$\begin{aligned} P^{d'} &= \left(\bigoplus_{i \in \{1,\ldots,u\}} c_i \cdot m_i \right)^{2^{d_1} + \cdots + 2^{d_\delta}} = \prod_{j=1}^{\delta} \left(\bigoplus_{i \in \{1,\ldots,u\}} c_i^{2^{d_j}} \cdot m_i^{2^{d_j}} \right) \\ &= \bigoplus_{i_1, i_2, \ldots, i_\delta \in \{1,2,\ldots,u\}} \left(\prod_{j=1}^{\delta} c_{i_j}^{2^{d_j}} \cdot m_{i_j}^{2^{d_j}} \right). \end{aligned}$$

where the second equality holds since $(x \oplus y)^{2^k} = x^{2^k} \oplus y^{2^k}$ for each $x, y \in \mathbb{F}_{2^n}$ and each $k \in \mathbb{N}$. Hence, we conclude that only monomial products of the form

$$m_{i_1}^{2^{d_1}} \cdot m_{i_2}^{2^{d_2}} \cdot \ldots \cdot m_{i_\delta}^{2^{d_\delta}}$$

may occur in P^d, where $i_1, i_2, \ldots, i_\delta \in \{1, 2, \ldots, u\}$. The monomials $m_{i_1}, \ldots, m_{i_\delta}$ are not necessarily different, therefore the exponents in Eq. (48) are either powers of 2 or sums of powers of 2. □

The next lemma shows that the naive exponential bound δ^r for the algebraic degree is not only a trivial bound but can indeed be achieved.

Lemma 8. *Let the same conditions as in Theorem 17 hold. Furthermore, let $S(x) = \sum_{i=0}^{d} c_i \cdot x^i$ for $c_i \in \mathbb{F}_{2^n}$, and let d' be a degree for which $hw(d') = \delta$ and $c_{d'} \neq 0$. Let $d' = \sum_{i=1}^{\delta} 2^{d_i}$ be the base-2 expansion of d' for appropriate $d_i \in \mathbb{N}$. In the first $R_{exp} = 1 + \lfloor \log_\delta(t) \rfloor$ rounds the algebraic degree grows as fast as δ^r.*

Proof. The idea is to observe the growth of the algebraic degree with the help of Lemma 7. After the first round, all monomials $X_1^{d'}, \ldots, X_t^{d'}$ are present in the polynomial representation of E_k^r and have algebraic degree δ.

According to Lemma 7, after one more round all monomials of the form $(i_1, \ldots, i_\delta \in \{1, \ldots, t\})$

$$(X_{i_1}^{d'})^{2^{d_1}} \cdot (X_{i_2}^{d'})^{2^{d_2}} \cdot \ldots \cdot (X_{i_\delta}^{d'})^{2^{d_\delta}},$$

are present in the encryption polynomial and have algebraic degree δ^2 if $i_1, \ldots, i_\delta$ are pairwise different. To see why they have algebraic degree δ^2, we note that: *(a)* raising a (word-level) monomial of E_k^r to the power of 2^k, $k \in \mathbb{N}$, does not change its algebraic degree, and *(b)* if two (word-level) monomials $m_{\alpha_1}, m_{\alpha_2}$ of E_k^r do not contain any shared variable, the algebraic degree of the product $m_{\alpha_1} \cdot m_{\alpha_2}$ is the sum of the respective algebraic degrees.

In the same way as before, after another round, all monomials of the form $(i_1, \ldots, i_{\delta^2} \in \{1, \ldots, t\})$

$$(X_{i_1}^{d' \cdot 2^{d_1}} \cdots X_{i_\delta}^{d' \cdot 2^{d_\delta}})^{2^{d_1}} (X_{i_{\delta+1}}^{d' \cdot 2^{d_1}} \cdots X_{i_{2\delta}}^{d' \cdot 2^{d_\delta}})^{2^{d_2}} \cdots (X_{i_{\delta^2-(\delta-1)}}^{d' \cdot 2^{d_1}} \cdots X_{i_{\delta^2}}^{d' \cdot 2^{d_\delta}})^{2^{d_\delta}}$$

appear in the encryption polynomial and have algebraic degree δ^3 if $i_1, \ldots, i_{\delta^2}$ are pairwise different. Continuing this way, we conclude that the algebraic degree grows as fast as δ^r until all t variables are exhausted, i.e., until $\delta^r = \delta \cdot t$, or equivalently, for the first $\lfloor \log_\delta(\delta \cdot t) \rfloor = 1 + \lfloor \log_\delta(t) \rfloor$ rounds. □

About the Linear Growth

Lemma 9. *Let the same conditions as in Theorem 17 hold. Then, the algebraic degree of E_k^r after r rounds, denoted by $\delta(r)$, is upper-bounded by*

$$\delta(r) \leq t \cdot \log_2 \left(\frac{r-1 \cdot d}{t} + 1 \right). \tag{49}$$

Proof. Since the word-level degree of a single output word of E_k^r after r rounds is upper bounded by $^{r-1} \cdot d$ (we note, the final linear layer does not affect the algebraic degree) the algebraic degree $\delta(r)$ of E_k^r after r rounds can be upper bounded by

$$\delta(r) \leq \max_{\{(e_1, \ldots, e_t) \in \mathbb{N}^t \,:\, \sum_{i=1}^t e_i \leq {}^{r-1} \cdot d\}} \sum_{i=1}^{t} \mathrm{hw}(e_i),$$

where we use the fact that the algebraic degree of a monomial $X_1^{e_1} \cdot \ldots \cdot X_t^{e_t}$ is given by $\sum_{i=1}^t \mathrm{hw}(e_i)$.

Let $(e_1, \ldots, e_t) \in \mathbb{N}^t$ be arbitrary with $\sum_{i=1}^t e_i \leq {}^{r-1} \cdot d$. We observe that $2^w - 1$ is the smallest number with hamming weight $w \in \mathbb{N}$. This means that $2^{\mathrm{hw}(e_i)} - 1 \leq e_i$, hence $\mathrm{hw}(e_i) \leq \log_2(e_i + 1)$ and

$$\sum_{i=1}^{t} \mathrm{hw}(e_i) \leq \sum_{i=1}^{t} \log_2(e_i + 1).$$

Let $(e_1, \ldots, e_t) \in \mathbb{N}^t$ such that $\sum_{i=1}^t e_i \leq {}^{r-1} \cdot d$. The logarithm is concave, which means that

$$a \cdot \log_2(x) + (1 - a) \cdot \log_2(y) \leq \log_2(a \cdot x + (1 - a) \cdot y)$$

for $a \in [0, 1]$. This is commonly generalized by induction to

$$\sum_{i=1}^{t} a_i \cdot \log_2(x_i) \leq \log_2 \left(\sum_{i=1}^{t} a_i \cdot x_i \right)$$

whenever $\sum_{i=1}^{t} a_i = 1$ and $a_i \in [0,1]$ for all i. Therefore

$$\sum_{i=1}^{t} \log_2(e_i+1) = t \cdot \sum_{i=1}^{t} \frac{1}{t} \log_2(e_i+1)$$
$$\leq t \cdot \log_2\left(\sum_{i=1}^{t} \frac{e_i+1}{t}\right) \leq t \cdot \log_2\left(\frac{r-1 \cdot d}{t}+1\right),$$

where the last inequality holds because $\sum_{i=1}^{t} e_i \leq {}^{r-1} \cdot d$ and the fact that the logarithm is an increasing function. Combining this with the initial equation results in the desired

$$\delta(r) \leq t \cdot \log_2\left(\frac{r-1 \cdot d}{t}+1\right). \qquad \square$$

8.3.4 Discussion of Theorem 17

Forward versus Backward Direction. As originally proved in Corollary 3 of [BC13], given a fixed key k, the algebraic degrees of E_k^r and its compositional inverse E_k^{-r} are related in a particular way: the algebraic degree of E_k^r is maximal (i.e. $n \cdot t - 1$) if and only if the algebraic degree of E_k^{-r} is maximal. As an immediate consequence we state the following observation: *the number of rounds to reach maximal algebraic degree in the forward and in the backward direction is the same.* This fact is particularly surprising if one direction of an SPN scheme is defined via low-degree S-Boxes, while the inverse direction is built from S-Boxes of high degree. For example, for the S-Box function $S(x) = x^3$ over $\mathbb{F}_{2^n}$ the inverse function is given by $S^{-1}(x) = x^{(2^{n+1}-1)/3}$. Here, S has algebraic degree 2, while S^{-1} has algebraic degree $(n+1)/2$.

Remarks on implicit assumptions. According to the remark about the connection of forward and backward direction below, it suffices to focus only on one direction of the scheme when attempting to reach maximal algebraic degree. We focus on the forward direction. Furthermore, our analysis is independent of the concrete instantiation of the linear layer, besides assuming it is invertible and it ensures full diffusion after a finite number of rounds. Implicitly, our proof assumes the strongest possible linear layer, i.e., a linear layer that guarantees full diffusion after one round and whose corresponding linearized polynomial is full. Therefore, depending on the instantiation of the linear layer, the algebraic degree might grow slower than we predict, but never faster. Theorem 17 can easily be generalized to the case in which the S-Boxes are defined via different invertible functions, under the assumption that they all have the same univariate degree d and the same algebraic degree δ.

Relation to Iterated Even–Mansour Schemes. The authors of [EGL+20] state in Section 3.3 that for an iterated Even–Mansour scheme whose round function can be described by a low-degree polynomial that

> *"[...] if the round function can be described by a polynomial of low univariate degree d over $\mathbb{F}_{2^n}$, we expect a linear behavior in* [the algebraic degree] $\delta_{lin}(r)$*:* $\delta_{lin}(r) \leq \lfloor \log_2(d^r+1) \rfloor \approx r \cdot \log_2(d)$".

However, no formal proof of this expectation is given in [EGL+20]. Our Theorem 17 comprises this situation as special case $t = 1$ and $l = 1$; thus we not only prove but also generalize the result in [EGL+20]. Indeed, in

Theorem 17 the case $t = l = 1$ corresponds to iterated Even–Mansour schemes and hence the algebraic degree $\delta(r)$ after r rounds is upper bounded by $\log_2(d^r + 1)$.

Comparison with Interpolation Analysis. The previous bound on the necessary number of rounds matches the number of rounds needed to guarantee security against the interpolation analysis [JK97] introduced by Jakobsen and Knudsen at FSE 1997. The goal of an interpolation analysis is to construct the polynomial that describes the encryption or decryption function. Hence, if the number of monomials is too large, such a polynomial cannot be constructed faster than via a brute force search. Since the number of monomials can be estimated by means of the given the degree of the function, the designers must guarantee that the polynomial that represents the scheme is of maximum degree and full (or at least dense) to guarantee security against this type of cryptanalysis.

8.4 Comparison of Theorem 17 with the Results in [BCC11]

8.4.1 Iterative Application of the Bound in [BCC11]

The bounds on the algebraic degree in [BCC11] are stated for the composition of two functions which means that the application to iterated SPN schemes (which often comprise the composition of several dozen functions) requires an ad-hoc analysis of the analyzed scheme. Here, we first provide a closed formuala for the bound in [BCC11, Theorem 2] when extended to the composition of more than two functions, which provides the basis for our comparisons in Section 8.5.

The bound given by Boura, Canteaut, and De Cannière in [BCC11, Theorem 2] states the following: Let F be a function from $\mathbb{F}_2^N$ to $\mathbb{F}_2^N$ corresponding to the concatenation of t smaller balanced[2] S-Boxes $S_1, \ldots, S_t$ defined over $\mathbb{F}_2^n$. Then, for any function G from $\mathbb{F}_2^N$ to $\mathbb{F}_2^N$, it holds

$$\deg(G \circ F) \leq N - \frac{N - \deg(G)}{\gamma}, \tag{50}$$

where

$$\gamma := \max_{i=1,\ldots,n-1} \frac{n-i}{n-\delta_i} \leq n-1, \tag{51}$$

and δ_i is defined as the maximal algebraic degree of the product of any i coordinates of any of the smaller S-Boxes.

We emphasize that γ and δ_i depend on the details of the S-Box. Namely, two S-Boxes with the same algebraic degree can have in general different γ. The result in [BC13, Theorem 2] uses the algebraic degree of the compositional inverses S_j^{-1}, $1 \leq j \leq t$, for a bound on the algebraic degree of $G \circ F$. Under the same assumptions as above this result leads to the same bound as stated in Eq. (50), with the additional upper bound on γ

$$\gamma \leq \max_{1 \leq j \leq t} \max \left\{ \frac{n-1}{n - \deg(S_j)}, \frac{n}{2} - 1, \deg\left(S_j^{-1}\right) \right\}. \tag{52}$$

Using an upper bound on γ for bounding the algebraic degree of $G \circ F$ in Eq. (50) could lead to a less tight bound on $\deg(G \circ F)$ than using the exact

[2] A function $f : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is said to be *balanced* if each element in $\mathbb{F}_2^m$ has exactly 2^{n-m} preimages. For $n = m$, an S-Box is balanced iff it is invertible.

value of γ. However, Eq. (52) has the advantage that it only uses known facts about the involved functions and thus a bound on $\deg(G \circ F)$ can be computed straight away. The same remark applies to another bound in [BC13, Corollary 2], which works with the algebraic degree of F^{-1} and is given by

$$\deg(G \circ F) < N - \left\lfloor \frac{N - 1 - \deg(G)}{\deg(F^{-1})} \right\rfloor.$$

In Proposition 25, we derive a *direct* upper bound of the algebraic degree of SPN schemes in the simple but most common case where all S-Boxes are equal. With "direct" upper bound we mean that we iteratively apply (50) to the round functions of an SPN scheme and thus obtain a closed-form statement about the algebraic degree after a certain number of rounds (and not only for the composition of two functions as stated in [BCC11]).

Proposition 25. *Let F be a function from $\mathbb{F}_2^N$ to $\mathbb{F}_2^N$ corresponding to the concatenation of t copies of a balanced S-Box S over $\mathbb{F}_{2^n}$ with algebraic degree $\delta \geq 2$. For any affine functions $L_1, L_2, \ldots, L_r$ from $\mathbb{F}_2^N$ to $\mathbb{F}_2^N$ and any integer $r \geq 1$ consider the SPN scheme E_r from $\mathbb{F}_2^N$ to $\mathbb{F}_2^N$ defined as*

$$E_r := L_r \circ F \circ L_{r-1} \circ F \circ \cdots \circ L_1 \circ F.$$

Then the algebraic degree $\delta(r)$ of E after r rounds is upper-bounded by

$$\delta(r) \leq \begin{cases} \delta^r & \text{if } r \leq R_0 := \left\lfloor \log_\delta \left(N \cdot \frac{\gamma - 1}{\gamma \cdot \delta - 1} \right) \right\rfloor, \\ \frac{\delta^{R_0}}{\gamma^{r - R_0}} + N \cdot \left(1 - \frac{1}{\gamma^{r - R_0}}\right) & \text{if } R_0 < r \leq \mathcal{R}_{[\text{BCD11}]}, \end{cases} \tag{53}$$

independent of the (secret or publicly known) key k, where

$$\mathcal{R}_{[\text{BCD11}]} := \underbrace{\left\lfloor \log_\delta \left(N \cdot \frac{\gamma - 1}{\gamma \cdot \delta - 1} \right) \right\rfloor}_{= R_0} + \left\lceil \log_\gamma \left(N - \delta^{R_0} \right) \right\rceil \tag{54}$$

is the minimum number of rounds for security against higher-order differential distinguishers and where γ is defined as in Eq. (51).

The proof of Proposition 25 can be found in Appendix 8.A. The strategy we adopt to prove Proposition 25 is similar to the one proposed by Biryukov, Khovratovich, and Perrin [BKP16]. In there, authors focused on the case in which all S-Boxes have maximum algebraic degree $\delta = n - 1$, while here we do not need this restriction. We point out one more time that the details of the linear layer are not taken into account and do not influence the bound just given.

Cost of Computing γ. The growth of the degree predicted in (50) depends on the value of γ. Computing γ can be very expensive for large S-Boxes. Indeed, one has to consider all possible combinations of the product of any i coordinates of the given S-Boxes, which implies a lower bound on the cost of order

$$\Omega\left(\sum_{i=1}^{n} \binom{n}{i} \right) \approx \Omega(2^n).$$

In the case in which t different S-Boxes are used, the previous cost must be multiplied by t. This means that for large S-Boxes (e.g., $n \geq 64$) it is infeasible to determine γ computationally and a further analysis of the scheme is necessary. Our results in Section 8.3 do not have this limitation. They depend on known parameters of the scheme and can be computed straight away.

8.4.2 Comparison and Impact of the Linear Layer

Comparison. For a better insight when the bound $\mathcal{R}_{\text{SPN}}$ improves upon the one given by $\mathcal{R}_{[\text{BCD11}]}$ we ask the following question: *For which values of* n, t, d, l *and* δ *is*

$$\mathcal{R}_{\text{SPN}} \geq \mathcal{R}_{[\text{BCD11}]}$$

satisfied? Substituting the corresponding expressions we obtain the following inequality

$$1 + \log\left(t \cdot (2^n - 1)\right) - \log(d) \geq \left\lfloor \log_\delta\left(N \cdot \frac{\gamma - 1}{\gamma \cdot \delta - 1}\right)\right\rfloor + \left\lceil \log_\gamma\left(N \cdot \frac{\gamma \cdot (\delta - 1)}{\gamma \cdot \delta - 1}\right)\right\rceil.$$

Using the relations $\gamma \cdot \delta - 1 \geq \gamma - 1$ and $\gamma \cdot \delta - 1 \geq \delta - 1$ (note that $\delta \geq 2$), an upper bound for $\mathcal{R}_{[\text{BCD11}]}$ is given by

$$\mathcal{R}_{[\text{BCD11}]} \leq 1 + \lfloor \log_\delta(N) \rfloor + \lceil \log_\gamma(N) \rceil \leq 1 + \lceil \log_\delta(N) \rceil + \lceil \log_2(N) \rceil.$$

Focusing on the case $n \gg 1$, the condition $\mathcal{R}_{\text{SPN}} \geq \mathcal{R}_{[\text{BCD11}]}$ is satisfied if (approximately)

$$\begin{aligned} 1 + \log\left(t \cdot (2^n - 1)\right) - \log(d) &\approx n \cdot \log(2) + \log(t) \\ &\geq 1 + \log_\delta(n \cdot t) + \log_2(n \cdot t), \end{aligned}$$

or to put it another way, if

$$\underbrace{n \cdot \log(2) + \log(t)}_{\in \mathcal{O}(n)} \geq \underbrace{(\log_2(n) + \log_2(t)) \cdot (1 + \log_\delta(2)) + 1}_{\in \mathcal{O}(\log_2(n))}. \tag{55}$$

It is easy to see that for any fixed values of d, δ, l and t, the previous inequality can be satisfied if n is large enough.

Impact of the Linear Layer. According to Theorem 17, after an exponential growth, the algebraic degree grows at most linearly with a rate equal to $t \cdot \log_2()$. If $l = 1$ (and thus $= d$) the degree l of the linear layer does not infuence the algebraic degree. However, if $l \geq 2$, the initial exponential growth can take place for more than R_{exp}; as an extreme case, if l is close to its maximum possible value 2^{n-1}, the linear growth may never occur. A concrete example of these facts is given in Fig. 8. Concluding, the details of the linear layer play a crucial role in the growth of the (algebraic) degree.

8.5 Practical Results

In this section, we present our practical results on SPN schemes over $(\mathbb{F}_{2^n})^t$ (defined as in Section 8.3) with low-degree and large S-Boxes. Assuming $= d \cdot l$, we focus on the two cases *(1)* $l = 1$, $t \geq 2$; and *(2)* $l \geq 2$, $t = 1$. This allows us to emphasize the impact of t and l independently. Since the approach we take is the same for all of our tests, we will first describe it.

8.5.1 Test Methodology

Instead of computing the ANF of a (keyed or keyless) permutation (which is quite expensive already for small field sizes[3]), we evaluate the zero-sum

[3] For example, the computation of the Möbius transform is exponential in the bit size [BCB20], and other methods (like the symbolic evaluation of the multiplication) are only feasible for small n or large n with small d (i.e., a small number of multiplications).

Algorithm 14: Evaluating the zero sum property of an SPN scheme E_k^r over $(\mathbb{F}_{2^n})^t$ using different input subspaces.

Data: SPN scheme E_k^r using r rounds, with S-Box size n and t words, dimension D of the subspace, number of tests n_T.
Result: *True* if a zero sum is found in all tests, *False* otherwise.

```
for i ← 1 to n_T do
    Randomly distribute D active bits among the N = n · t possible
     positions, resulting in the input vector space V ⊆ F_2^N.;
    Randomly sample round constants c_1, ..., c_r and v.;
    Randomly sample key k.;
    Fix E_k^r using c_1, ..., c_r and k.;
    s ← 0.;
    foreach x ∈ V ⊕ v do
        s ← s ⊕ E(x).;
    if s ≠ 0 then
        return False.;
return True.;
```

property for multiple random input vector spaces. For this purpose, we wrote a custom program in C++.[4]

For random keys and constants, given an input subspace of dimension $D \leq N - 1$, where $N = n \cdot t$, we look for the minimum number of rounds r for which the corresponding sum of the outputs is different from zero. Such a number corresponds to

(1) the minimum number of rounds for reaching algebraic degree $\delta = D + 1$, and

(2) the minimum number of rounds for preventing higher-order differential distinguishers for $D = N - 1$.

To avoid a bias by weak keys or "bad" round constants, we have repeated the tests multiple times (with new random keys, round constants, and input subspaces).

We illustrate the approach in Algorithm 14 using a keyed permutation.

Number of Subspaces of Dimension D. We emphasize, if the algebraic degree of an SPN scheme E_k^r after r rounds is $\delta(r)$, then summing over all evaluations from any vector space of dimension $D \geq \delta(r) + 1$ always results in a zero sum, i.e., $\bigoplus_{x \in \mathcal{V}} E_k^r(x \oplus v) = 0$ for a generic (fixed) v. However, the converse is not true in general. That is, having a zero sum over a vector space of dimension D, does in general not imply that the algebraic degree is $\delta(r) = D - 1$. Indeed, $\delta(r)$ could be higher, and the zero sum could occur merely due to the specific structure of the vector space and the analyzed function.

Evaluating the zero sum property for all affine subspaces of dimension D is actually infeasible. Indeed, when working over $(\mathbb{F}_p)^N$, for any prime p and $N \in \mathbb{N}$, the number of different subspaces of dimension $D \leq N$ is

$$\frac{(p^N - 1) \cdot (p^N - p) \cdot (p^t - p^2) \cdot \cdots \cdot (p^N - p^{D-1})}{(p^D - 1) \cdot (p^D - p) \cdot (p^D - p^2) \cdot \cdots \cdot (p^D - p^{D-1})} \in \mathcal{O}\left(p^{D \cdot (N-D)}\right)$$

[4] The code we used for the practical tests can be found on GitHub: https://github.com/IAIK/higher-order-differential

as shown, e.g., in [Hog06], which is out of practical range even for small values of p, N, D. For this reason, we have to limit ourselves to evaluate the zero sum property for a limited number of subspaces only. However, in our practical tests we observed that a small number of tests *for each of the possible combinations* of active bits is sufficient to derive a stable number (e.g., around 10 tests for each combination). Indeed, for example, we observed no differences when using an input subspace of dimension $N - 1$ and changing the position of the single inactive bit in multiple tests.

The practical number of rounds to prevent higher-order differential distinguishers we report is *the smallest number of rounds among all tested keys and round constants*. This means that potentially a higher number of rounds can be cryptanalyzed by choosing the keys and round constants in a particular way.

Randomization of Active Bits. Depending on the position of the active bits, the final results may be very different. For example, significant differences arise when considering a fixed number of active bits in a single word and the same number of active bits split over multiple words. In order to counteract this problem, we choose the input subspaces randomly such that the position of active bits is also randomized. As a concrete example, consider $t = 2$ with $d = 3$ and arbitrary n. Clearly, after one round the algebraic degree is upper-bounded by $\delta = 2$, and indeed, when activating 2 bits in the same word, we do not get a zero sum. However, if we activate one bit in each of the two words (i.e., in total also 2 bits), we do get a zero sum, since only products of at most $\delta = \mathrm{hw}(d) = 2$ bit variables from the *same* word occur in the polynomial representation. Hence, we randomize the input subspaces in our tests.

Computational Cost in Practice. In our practical tests we observed that with very few trials we already reach a stable number for the algebraic degree after a certain number of rounds. It is however crucial to test every possible combination of active words, since this has a significant impact on the final result. Concretely, we fix the number of tests to 100 for "feasible" numbers of active bits (i.e., around 30). For the larger tests, we fix the number to 10. While this may seem like a small sample size, we could not observe any differences when testing more often with lower numbers of bits. As for the concrete runtime, it largely depends on the number of active bits, but also on additional properties like the tested degree. E.g., x^3 can be evaluated faster than x^7 for a given S-Box input x. Practically, a test with 30 active bits can thus take several hours depending on the concrete tested construction.

8.5.2 Results for SPN Schemes with $t \geq 2$, $l = 1$ and S-Boxes of the form $S(x) = x^d$

In our experiments, we focus on a SHARK-like scheme [RDP+96] with power maps as S-Box functions. More specifically, we focus on SPN schemes over $(\mathbb{F}_{2^n})^t$ where the S-Box function $S : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ is given by $S(x) = x^d$ and the mixing layer is defined as the multiplication of the t state words with an invertible $t \times t$ matrix over $\mathbb{F}_{2^n}$. The choice of n and d is governed by the requirement $\gcd(d, 2^n - 1) = 1$, ensuring that $S(x) = x^d$ is a permutation of $\mathbb{F}_{2^n}$.

For the S-Box $S(x) = x^3$, we report our results on the minimum number of rounds to prevent higher-order differential distinguishers in Table 7. We

Table 7: Theoretical *lower* bound and practical number of rounds for preventing higher-order differential distinguishers on SPN schemes over $(\mathbb{F}_{2^n})^t$ for several values of n and $t \geq 2$ (where $N = n \cdot t$). The chosen S-Box is the cube function $S(x) = x^3$. For the practical number of rounds, we consider both the case of an MDS matrix and the case of a matrix that provides the "worst" possible diffusion (e.g., a sparse matrix as in Eq. (57)). $\mathcal{R}_{[\text{BCD11}]}$ is computed assuming $\gamma = (n+1)/2$.

Parameters			Theoretical # of Rounds		Practical # of Rounds	
N	n	t	$\mathcal{R}_{\text{SPN}}$	$\mathcal{R}_{[\text{BCD11}]}$	MDS matrix	Sparse matrix
35	5	7	5	6	8	15
35	7	5	6	6	8	12
36	9	4	7	6	9	11
33	11	3	8	5	10	10
39	13	3	10	6	11	12
34	17	2	12	6	12	12
38	19	2	13	6	14	14
66	11	6	9	7	-	-
65	13	5	10	6	-	-
60	15	4	11	6	-	-
66	17	4	12	7	-	-
63	21	3	15	6	-	-
66	33	2	22	7	-	-
132	11	12	10	8	-	-
135	15	9	12	8	-	-
133	19	7	14	7	-	-
132	33	4	22	8	-	-
129	43	3	28	7	-	-
130	65	2	42	8	-	-

observe that the number of rounds that can be covered by a higher-order differential distinguisher is always close to the one predicted by our formula (in some cases a little higher, but never smaller). Moreover, especially when the size of the S-Box is not too small, the round number $\mathcal{R}_{\text{SPN}}$ predicted by our formula is significantly larger than $\mathcal{R}_{[\text{BCD11}]}$. Furthermore, our results of small-scale experiments on the growth of the algebraic degree (according to the test methodology in Section 8.5.1) for $S(x) = x^3$ and $S(x) = x^7$ are depicted in Fig. 9 and Fig. 10, respectively.

Note that the tests made for Table 7 and, e.g., Fig. 9 use different approaches: in the former case we maximize the number of active bits and see how many rounds we can distinguish, whereas in the latter case we want to estimate the algebraic degree via the number of active bits. For this reason, more test runs are needed to determine the degree growth, especially in order to take care of the different positions of the active bits (where the number of choices is lower for Table 7, since $N-1$ bits are active in all tests).

Determining γ. To use the results from [BCC11] for our comparisons we need to determine the parameter γ (see also Eq. (51)). Since an exact computation of γ is too expensive for most instances we use, we derive an

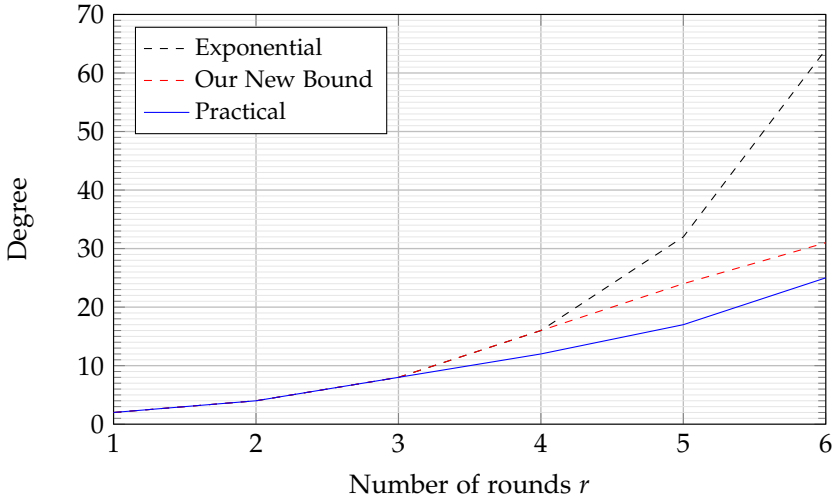


Figure 9: Degree growth for an SPN scheme over $(\mathbb{F}_{2^{33}})^4$ instantiated with the S-Box $f(x) = x^3$.

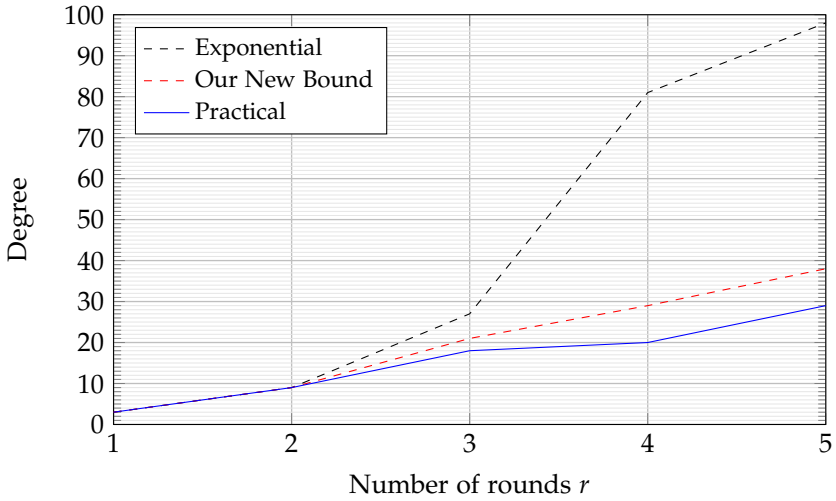


Figure 10: Degree growth for an SPN scheme over $(\mathbb{F}_{2^{33}})^3$ instantiated with the S-Box $S(x) = x^7$.

upper bound on γ and use this upper bound as a benchmark. By definition of γ, it holds

$$\begin{aligned}\gamma &= \max_{1\le i\le n-1} \frac{n-i}{n-\delta_i} = \max\left\{\max_{1\le i\le q} \frac{n-i}{n-\delta_i}, \max_{q+1\le i\le n-1} \frac{n-i}{n-\delta_i}\right\} \\ &\le \max\left\{\max_{1\le i\le q} \frac{n-i}{n-i\cdot\delta}, \max_{q+1\le i\le n-1} \frac{n-i}{n-(n-1)}\right\} \\ &= \max\left\{\frac{n-q}{n-q\cdot\delta}, n-(q+1)\right\}.\end{aligned}$$

where $q = \lfloor (n-1)/\delta \rfloor$ and $\delta = \mathrm{hw}(d)$ is the algebraic degree of the S-Box. For the particular case $S(x) = x^3$ only odd values for n are allowed (to guarantee $\gcd(2^n - 1, 3) = 1$) and thus we obtain $n - 1 = q \cdot 2$. Hence,

$$\gamma \le \max\left\{\frac{n-\frac{n-1}{2}}{n-2\cdot\frac{n-1}{2}}, n-\frac{n-1}{2}-1\right\} = \frac{n+1}{2}. \tag{56}$$

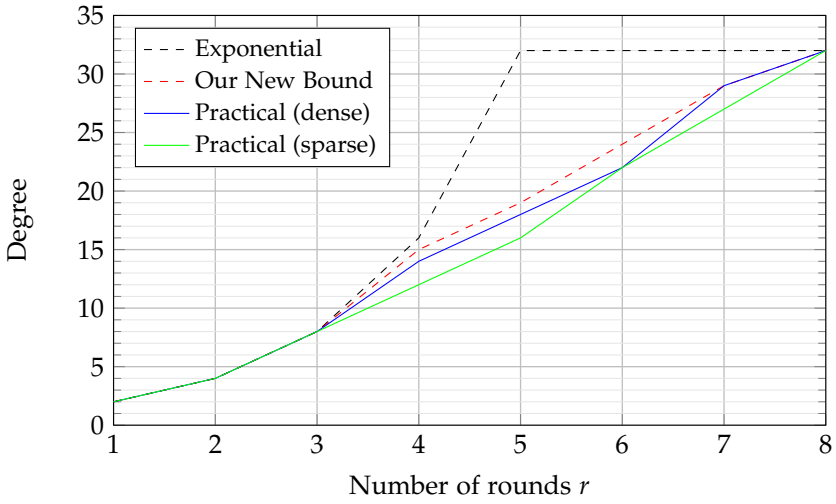


Figure 11: Degree growth for an iterated Even–Mansour scheme over $\mathbb{F}_{2^{33}}$ with a linearized polynomial of degree $l = 2^3$ as linear layer and instantiated with the S-Box $S(x) = x^3$.

We assume $\gamma = (n+1)/2$ to compute the theoretical values for $\mathcal{R}_{[\text{BCD11}]}$. We also refer to [EGL+20, Lemma 3], where authors support this assumption by practical experiments for each odd $n \leq 33$.

Influence of the Linear Layer. To understand how the linear layer influences the minimum number of rounds to prevent higher-order differential distinguishers, in our practical tests we consider two extreme cases: *(1)* we evaluate the case in which the linear layer is defined as the multiplication with an MDS matrix (for parameters n and t that allow us to do so[5]), which corresponds to the case of the "strongest" linear layer from a diffusion point of view; *(2)* we also evaluate the case in which the linear layer is "weak", which could happen if it is defined by the multiplication with a matrix containing a large number of zero coefficients. For this second case, we used a $t \times t$ matrix M with coefficients $M_{r,c}$ given by

$$M_{r,c} = \begin{cases} 1 & \text{if } r = 0 \text{ or } c \equiv r+1 \bmod t, \\ 0 & \text{otherwise.} \end{cases} \tag{57}$$

We note, using M from Eq. (57) we need t rounds to have full diffusion (at word level), instead of just one round as for the MDS case. Hence, especially for large t, we expect that more rounds than previously predicted may be necessary to guarantee security against higher-order differential distinguishers. In Table 7 we report empirical evidence for this expectation: the gap between the number of rounds predicted by our formula and the one found by practical tests in the case of a sparse matrix is close to zero for "small" t, and grows for "large" t.

8.5.3 Results for Iterated Even–Mansour Schemes ($t = 1$) with $l \geq 2$ and S-Boxes of the form $x \mapsto x^d$

We focus on an iterated Even-Mansour scheme with a power map as S-Box function. More specifically, we focus on a scheme over $\mathbb{F}_{2^n}$ where the S-Box

[5] An MDS matrix over $\mathbb{F}_{2^n}^{t \times t}$ exists if the condition $\log_2(2t+1) \leq n$ (i.e., $t \leq 2^{n-1} - 1$) is satisfied.

Table 8: Theoretical lower bound and practical number of rounds for preventing higher-order differential distinguishers on iterated Even–Mansour schemes over $\mathbb{F}_{2^n}$ for several values of n and $l \geq 1$. The chosen S-Box is the cube function $S(x) = x^3$. For the practical number of rounds, we consider two cases regarding the linearized polynomial M, namely, M dense and M sparse. $\mathcal{R}_{[\text{BCD11}]}$ is computed assuming $\gamma = (n+1)/2$.

Parameters		Theoretical # of Rounds		Practical # of Rounds	
n	l	$\mathcal{R}_{\text{SPN}}$	$\mathcal{R}_{[\text{BCD11}]}$	Dense M	Sparse M
33	1	21	5	21	21
33	2	13	5	13	13
33	4	10	5	10	10
33	8	8	5	8	8
33	16	7	5	7	7
33	32	6	5	6	7
65	1	41	6	-	-
65	2	26	6	-	-
65	4	19	6	-	-
65	8	15	6	-	-
65	16	13	6	-	-
65	32	11	6	-	-
129	1	81	7	-	-
129	2	50	7	-	-
129	4	37	7	-	-
129	8	29	7	-	-
129	16	24	7	-	-
129	32	21	7	-	-

function $S : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ is given by $S(x) = x^d$ and the linear layer is defined as a linearized permutation polynomial of degree $l := 2^{l'}$. As in Section 8.5.2, n and d are chosen such that $S(x) = x^d$ is a permutation of $\mathbb{F}_{2^n}$.

We consider two different cases for the linearized polynomial:

- A dense linearized polynomial. In this case our polynomial is equal to $M(x) = \sum_{i=0}^{l'} \lambda_i \cdot x^{2^i}$ for $\lambda_i \in \mathbb{F}_{2^n} \setminus \{0\}$ that guarantee invertibility;
- A sparse linearized polynomial. In this case our polynomial is equal to $M(x) = \lambda \cdot x^l + \lambda' \cdot x^{l_0}$ for small $l_0 = 2^{\tilde{l_0}}$ (usually, $l_0 = 1$) and $\lambda, \lambda' \in \mathbb{F}_{2^n} \setminus \{0\}$ that guarantee invertibility.

For the S-Box $S(x) = x^3$, we report our results on the minimum number of rounds to prevent higher-order differential distinguishers in Table 8 and depict the growth of the algebraic degree for smaller number of rounds in Fig. 11. We observe that the algebraic degree grows close to our bound for both the sparse and dense cases, where the sparse case grows slightly slower than the dense case. In fact, when only looking at the minimum number of rounds required to prevent higher-order differential distinguishers as in Table 8, almost all results coincide: the only exception is the case of $n = 33$, $l = 32$ where a sparse linear polynomial requires one extra round. A more substantial difference is found between the round number $\mathcal{R}_{\text{SPN}}$ predicted by our formula and $\mathcal{R}_{[\text{BCD11}]}$, where the latter does not depend on l and is significantly smaller.

For the difference in test methodology regarding Table 8 and the graph in Fig. 11 the same remark as in Section 8.5.2 applies.

Special Case: $M(x) = \mu \cdot x^l$**.** Finally, we discuss the case in which the linearized polynomial is of the form $M(x) = \mu \cdot x^l$ for $l = 2^{l'}$ and $\mu \in \mathbb{F}_{2^n} \setminus \{0\}$. We remember that this function is always invertible over $\mathbb{F}_{2^n}$ ($x \mapsto x^2$ is always invertible, due to $\gcd(2, 2^n - 1) = 1$). Here, the value of l does not have any influence on the tests and the results are the same as for strong-arranged SPN schemes (i.e., for $l = 1$). This becomes evident when having a look at the relation between word-level degree and algebraic degree in Eq. (38). Exponentiating a monomial $m^e = X_1^{e_1} \cdot \ldots \cdot X_t^{e_t}$ to the power of $2^{l'}$ is in fact only an l'-shift of all (non-zero) digits in the base-2 expansion of e, hence

$$\delta\left(m^e\right) = \sum_{i=1}^{t} \mathrm{hw}(e_i) = \sum_{i=1}^{t} \mathrm{hw}\left(e_i \cdot 2^{l'}\right) = \delta\left(\left(m^e\right)^{2^{l'}}\right).$$

This means, the word-level degree is increased by a factor of $l = 2^{l'}$, but the algebraic degree remains the same. While the case $M(x) = \mu \cdot x^l$, for $l = 2^{l'}$, can be considered a degenerate case of a linear layer, the results of our experiments for this case do not contradict Theorem 17. We emphasize once more, the statement in Theorem 17 is an upper bound, and that the growth of the degree can be slower than predicted (which is true for every upper bound in the literature).

8.6 Possible Applications of Theorem 17

After the last advances in [BCC11], [BC13], and in [Car20], our findings extend the canon of theoretical bounds for the growth of the algebraic degree in SPN schemes by an improved bound, see Theorem 17. While the currently best bounds are more generic than our bound, our results substantially improve existing state-of-the-art bounds when considering SPN schemes with large S-Boxes and for which the degrees of both the non-linear layer and the linear layer are low, as is often the case in schemes for MPC-/FHE-/ZKP-applications. In these domain specific schemes, it is most often algebraic cryptanalysis, in particular higher-order differential distinguishers, that dominates the overall security arguments. Thus, a better understanding of the growth of the algebraic degree is not only vital for the security assessment of schemes for MPC-/FHE-/ZKP-applications but also for navigating design choices towards a more solid theoretical foundation.

HadesMiMC, Poseidon and Starkad. As a concrete application, HadesMiMC [GLR+20] is probably the most suitable candidate to apply our results. In particular, even if both HadesMiMC and POSEIDON are designed over $(\mathbb{F}_p)^t$, there is no reason why a scheme based on the Hades strategy cannot be designed over $(\mathbb{F}_{2^n})^t$. As a concrete example, we refer to STARKAD [GKR+21], a variant of POSEIDON defined over $(\mathbb{F}_{2^n})^t$.

Moreover, our upper bound for the growth of the algebraic degree plays an important role in higher-order differential distinguishers of SPN schemes over $\mathbb{F}_{2^n}^t$ that do not exploit the largest non-trivial vector subspace (i.e., $\mathbb{F}_2^{n \cdot t - 1}$), but subspaces of smaller dimension than the state size $n \cdot t$. This is not only of theoretical interest, but it applies to all cases in which the security level is smaller than the size of the full scheme, a scenario that is common for schemes recently proposed for MPC/FHE/ZKP-applications.

Schemes for MPC-/FHE-/ZKP-Applications. As we have seen in Section 8.2.2, the degree of a generic invertible $(n \cdot t) \times (n \cdot t)$ matrix with coefficients in $\mathbb{F}_2$ is in general very high when represented as a linearized polynomial over $\mathbb{F}_{2^n}$. In this case (namely, $l \approx 2^{n-1}$), our bound does not improve the naive exponential bound.

However, the situation is different for schemes used in MPC-/FHE-/ZKP-applications. In such applications, both the linear layer and the non-linear one are naturally defined over $\mathbb{F}_{2^n}$. One performance metric of schemes for MPC-/FHE-/ZKP-applications is, e.g., a minimal number of multiplications in $\mathbb{F}_{2^n}$, which is why usually linearized polynomials of low degree over $\mathbb{F}_{2^n}$ are used as linear layers. Concrete examples are JARVIS, and more recently the follow-up design VISION. JARVIS is an EM scheme over $\mathbb{F}_{2^n}$ (analyzed in [ACG+19]) with a linearized polynomial of degree 4 as linear layer. Compared to the possible maximum degree 2^{127}, the degree of this linearized polynomial is low. In a similar way, the linear layer of VISION is defined.

Consequently, in the case of SPN schemes with $l \geq 2$ designed for MPC-/FHE-/ZKP-applications , we expect that our results provide a better estimation of the algebraic degree than the naive exponential bound and the bound in [BC13], since in this scenario the linear layer usually has low degree when represented as a linearized polynomial over $\mathbb{F}_{2^n}$.

Acknowledgement. The authors thank anonymous reviewers for their valuable comments. We also thank Christina Boura for shepherding this final version of our manuscript. Lorenzo Grassi is supported by the European Research Council under the ERC advanced grant agreement under grant ERC-2017-ADG Nr. 788980 ESCADA. Aldo Gunsing is supported by the Netherlands Organisation for Scientific Research (NWO) under TOP grant TOP1.18.002 SCALAR.

8.a Proof of Proposition 25

Proof. Applying the naive exponential bound and the bound from [BCC11, Theorem 2] (see Eq. (50)) to $E_1 = L_1 \circ F$ yields

$$\deg(L_1 \circ F) \leq \min\left\{\delta, N \cdot \left(1 - \frac{1}{\gamma}\right) + \frac{1}{\gamma}\right\} = \delta.$$

The last equality is justified as follows: for $t = 1$, this is obvious (δ is exactly the degree of 1 round). For $t \geq 2$, this follows from the fact that the non-linear layer has degree δ (since we have parallel independent S-Boxes with algebraic δ) and that the linear layer does not change the algebraic degree.

In other words, for at least one round the naive exponential bound for the growth of the algebraic degree is better than the bound in [BCC11]. Therefore, we now look for the maximum number of rounds R_0 with this behavior. This corresponds to solving the following equation for R_0

$$\delta^{R_0} = N \cdot \left(1 - \frac{1}{\gamma}\right) + \frac{\delta^{R_0 - 1}}{\gamma},$$

which gives

$$R_0 = \log_\delta\left(N \cdot \frac{\gamma - 1}{\gamma \cdot \delta - 1}\right).$$

To put it another way, for any number of rounds $r \leq R_0$, the degree of E_r is upper-bounded by δ^r. As a next step, we find the minimum additional

number of rounds to prevent higher-order differential distinguishers, i.e., the minimum additional number of rounds R_1 such that the algebraic degree after $R_0 + R_1$ rounds is $N-1$ (the biggest non-trivial subspace of $\mathbb{F}_2^N$ has dimension $N-1$).

For $r > R_0$, the bound in [BCC11] is better than the naive bound, hence, the algebraic degree of E_r after $r = R_0 + 1$ rounds is upper-bounded by

$$\deg\left(E_{R_0+1}\right) \leq \underbrace{N \cdot \left(1 - \frac{1}{\gamma}\right)}_{=:C} + \frac{\delta^{R_0}}{\gamma} = C + \frac{\delta^{R_0}}{\gamma},$$

and after $r = R_0 + 2$ rounds by

$$\deg\left(E_{R_0+2}\right) \leq C + \frac{1}{\gamma} \cdot \left(C + \frac{\delta^{R_0}}{\gamma}\right) = C + \frac{C}{\gamma} + \frac{\delta^{R_0}}{\gamma^2}.$$

Continuing this way, we conclude that after $r = R_0 + s$ rounds, for an integer $s \geq 1$, the algebraic degree is upper bounded by

$$\deg\left(E_{R_0+s}\right) \leq \frac{\delta^{R_0}}{\gamma^s} + C \cdot \sum_{i=0}^{s-1} \frac{1}{\gamma^i} = \frac{\delta^{R_0}}{\gamma^s} + C \cdot \frac{1 - \frac{1}{\gamma^s}}{1 - \frac{1}{\gamma}} = \frac{\delta^{R_0}}{\gamma^s} + N \cdot \frac{\gamma^s - 1}{\gamma^s}.$$

This means, the minimum additional number of rounds R_1 to prevent higher-order differential distinguishers is given by the implicit condition

$$\frac{\delta^{R_0}}{\gamma^{R_1}} + \frac{N \cdot (\gamma^{R_1} - 1)}{\gamma^{R_1}} = N - 1,$$

which gives

$$R_1 = \log_\gamma\left(N - \delta^{R_0}\right).$$

We conclude, the minimum number of rounds $\mathcal{R}_{[\text{BCD11}]}$ to prevent higher-order differential distinguishers is given by

$$\mathcal{R}_{[\text{BCD11}]} = \left\lfloor \log_\delta\left(N \cdot \frac{\gamma - 1}{\gamma \cdot \delta - 1}\right)\right\rfloor + \left\lceil \log_\gamma\left(N - \delta^{R_0}\right)\right\rceil.$$

□

9

REINFORCED CONCRETE: A FAST HASH FUNCTION FOR VERIFIABLE COMPUTATION

Based on the peer-reviewed conference publication[1]

[GKL+22] Lorenzo Grassi, Dmitry Khovratovich, Reinhard Lüftenegger, Christian Rechberger, Markus Schofnegger, and Roman Walch. "Reinforced Concrete: A Fast Hash Function for Verifiable Computation." In: *SIGSAC Computer and Communications Security - CCS 2022*. Ed. by Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi. ACM, 2022, pp. 1323–1335. DOI: 10.1145/3548606.3560686

Abstract We propose a new hash function `Reinforced Concrete`, which is the first generic purpose hash that is fast both for a zero-knowledge prover and in native x86 computations. It is suitable for a various range of zero-knowledge proofs and protocols, from set membership to generic purpose verifiable computation. Being up to 15x faster than its predecessor Poseidon hash, `Reinforced Concrete` inherits security from traditional time-tested schemes such as AES, whereas taking the zero-knowledge performance from a novel and efficient decomposition of a prime field into compact buckets.

The new hash function is suitable for a wide range of applications like privacy-preserving cryptocurrencies, verifiable encryption, protocols with state membership proofs, or verifiable computation. It may serve as a drop-in replacement for various prime-field hashes such as variants of MiMC, Poseidon, Pedersen hash, and others.

Keywords Hash Functions, Verifiable Computation, ZKSNARKs, Finite Fields

9.1 Introduction

SNARKs and hash functions. The recent years have been marked as a thrive of distributed verifiable computation, where the outcome of some algorithm $\mathcal{A}$ is accompanied with a *succinct* proof of correctness, widely known as a *SNARK* [PHG+13; Gro16; MBK+19]. Performance of those protocols, however, remains a major bottleneck for applications. The reasons are manyfold, but one crucial point is that SNARKs are constructed for statements formulated over *prime fields* whereas regular computer programs are written for and executed over bitstrings. The necessary translation of code into finite field arithmetic carries a significant overhead. A notable example is the cost of computing 70 SHA-256 hash function calls, which were needed to transfer Zcash [BHH+22] cryptocurrency privately back in 2017, and which took over 40 seconds to create such a SNARK, compared to 10 microseconds of native computation on a PC. Thus, the design of various cryptographic primitives tailored for operating over finite fields is an active research area [AGR+16; GKK+19; AAB+20].

In this paper we remove one of such bottlenecks by offering a hash function that is fast both for SNARKs and native computation. There already exist

[1] A full version of this article can be found at https://eprint.iacr.org/2021/1038.pdf.

functions that excel in either of those areas, but not in both. The motivation for such a swissarmy tool is the following. To scale, parallelize, and aggregate proofs we employ what is called a *recursive proof* protocol [BCM+20; COS20; BDF+21; BCL+21], where a party can prove their share of computation *together* with a verification of proof coming from the predecessor. This also enables wrapping multiple proofs into a single succinct check. Notably, however, many such recursive protocols require both hashing the input of a party with Merkle tree and proving some openings of the tree in zero-knowledge (ZK). Thus, whatever hash function is selected for the tree, it must be fast in both scenarios. To make a concrete example, one of the most ZK-efficient hash functions to date, Poseidon [GKK+19], when plugged into the Fractal recursive protocol, makes the prover 100 times more expensive just because it is slow in the native x86 computation [COS20, Section 13.2].

Summary of use cases. In more details, our new hash function will address, among others, the following use cases:

- **Fast and efficient set membership proofs** based on Merkle tree accumulators. Immensely popular in cryptocurrency protocols [OWW+20; BHH+22; PSS21], this case requires a hash function for the tree. Parties $P_1, P_2, \ldots, P_n$ add entries $V_1, V_2, \ldots, V_k$ to some public accumulator $\mathfrak{A}$. Then at any point any party P_j can prove that $V_i \in \mathfrak{A}$. For instance, in Zcash [BHH+22] V_i are unspent transactions and $\mathfrak{A}$ is a Merkle tree over them, so that in order to spend transaction V an owner is required to provide a proof of knowledge that $V \in \mathfrak{A}$ as well as a proof of knowledge of some secret committed within V. Its ZK circuit should minimize the proof creation time.
- **Verifiable computation** based on recursive proofs. Here the entire computation is a chain of functions $F_1, F_2, \ldots, F_k$ applied consecutively to some state. Starting with X, for each i Party P_i computes F_i and carries an intermediate result and a proof of correctness to the next P_{i+1} so that the last P_k provides Y and attests $X \xrightarrow{F_k \circ F_{k-1} \circ \cdots \circ F_1} Y$ being actually aware only of their own computation and the proof of correctness π_{k-1} from P_{k-1}. Verifiable computation frameworks such as Halo Infinite [BDF+21] or Fractal [COS20] instruct that the proof π_k asserts the correctness of F_k and that the code C_k that verifies π_{k-1} outputs a success. If the inner commitment scheme is Merkle-tree-based (such as FRI [BBH+18a]), then π_{k-1} consists of several Merkle tree openings, so that C_k makes a number of calls to the hash function that comprises the tree. Here we minimize both native computation time and the prover time.

Both use cases require a cryptographically secure hash function, i.e., it should resist preimage and collision attacks.

Summary of requirements. We summarize the requirements stemming from the use cases as follows.

- Minimal prover time. For many ZK proof systems it is a (super)linear function of the gate count, where each gate is usually a basic field arithmetic operation or, in some systems, a table lookup [PHG+13; Gro16; BCR+19; GWC19; GW20]. Though the actual performance depends significantly on the proof system chosen and an application, the mere number of standard gates is a good approximation. It is known

that custom gates (lookup high-degree polynomials) may increase the performance up to the factor of 10, but those are function-specific and can't be reasonably compared across distinct proof systems. In Table 9 we provide a count in R1CS constraints (roughly, the number of field multiplications), in standard Plookup gates (each gate contains either a single multiplication and an arity-4 addition, or a table lookup), and in area-degree product (each custom gate contributes to the cost additively with the product of input size and the degree of the polynomial that describes the gate constraint). Unfortunately, we can't provide a sound prover time benchmark since at the moment of submission no production-ready proof system that supports lookups is available though specifications exist [PFM+22].

- Native performance. A hash function is supposed to run as fast as possible on typical hardware where proofs are created, which are regular laptops and desktops nowadays. The Fractal use case [COS20] implies that it should be at least 10x faster than Poseidon.
- Security. The common approach [AGR+16; GKK+19] is to provide evidence that the existing attacks fail. However, as algebraic attacks [ACG+19; GKR+22] are the most natural for finite-field-based designs, it becomes increasingly difficult to estimate the security as the performance of those attacks is highly volatile [BBL+22; SS21]. It is thus desirable to base the security of a new hash function on a more traditional [MRS+09] rather than algebraic security analysis.

State of the art. There already exist several hash functions crafted for the first use-case with the number of circuit gates (or equivalently low-degree polynomial constraints) being the primary metric. Examples include prime-field (Feistel) MiMC versions [AGR+16; AGP+19], Friday [AD18], Poseidon [GKK+19], Rescue [AAB+20] (and its updated version Rescue-Prime [AAB+20]), Griffin [GHR+23], Grendel [Sze21], Neptune [GOP+22]. Many of these hash functions share some common features, as the fact that the non-linear layer is instantiated via a simple power map. Focusing on Poseidon, it is based on the Hades design strategy [GLR+20], which makes use of an uneven distribution of the S-boxes, namely, full S-box layers in the external rounds and partial S-box layers in the middle ones, in order to minimize the multiplicative complexity. The external rounds provide security against statistical attacks, while the internal rounds have the goal of increasing the degree of the permutation. A rather recent addition to this set is Sinsemilla [BHH+22, Sec. 5.4.1.9], an instance of the Pedersen hash function[BHH+22, Sec. 5.4.1.7] optimized for table lookups in custom gates.

While most of them have withstood public scrutiny [ACG+19; BCD+20a; EGL+20; KR21; GRS21; BBL+22], the plain performance is not satisfactory (see last column of Table 9), since each round of such schemes requires a finite field multiplication, which is relatively expensive (hundreds of CPU cycles) compared to bit operations utilized in traditional hash functions.

Our design: Reinforced Concrete. We present a new sponge hash function `Reinforced Concrete`, in short `RC`, over $\mathbb{F}_p$ exploiting all the advantages of lookup-equipped proof systems and suitable for both membership proofs and verifiable computation use cases. The permutation that instantiates `RC` is composed of two types of components:

1. outer ones for preventing statistical attacks;

2. an inner one for preventing algebraic attacks.

The inner part strengthens the whole construction like steel bars strengthen concrete, hence the name of the function and its components.

For the inner component, instead of using simple power maps as in POSEIDON and RESCUE, we use a single building block with a complex algebraic structure, which we call `Bars`. A `Bars` layer can be seen as a non-linear layer composed of independent high-degree and dense S-boxes. The `Bars` function combines a layer of S-boxes (such as in AES) with a field element decomposition in just a handful of small operations (or table gates in the circuit), and it admits a very simple representation when using look-up tables, as e.g. in the case of AES [DR02b] and AES-like ciphers. As a result, the security argument we propose for preventing algebraic attacks including interpolation [JK97] and Grobner basis attacks [CLO15] resembles the one well known and accepted in the literature for AES and more generally AES-like ciphers, for which the algebraic attacks can attack only a tiny fraction of the rounds compared to the statistical attacks [CP02; CL05].

Even if it prevents algebraic attacks, it can be broken by more traditional statistical attacks such as rebound attacks [MRS+09; LMR+09]. As those are much better studied, we instantiated the external rounds with other layers which are known to protect against statistical attacks, including affine layers called `Concrete` that provides full diffusion and low-degree non-linear layer called `Bricks`, which both provides (non-linear) diffusion and ensure security against statistical attacks.

Our approach to performance. We tackle the performance issue by making the `Bars` layer fast in the native computation. For this we managed to avoid field multiplications altogether in this layer and do only a bunch modular reductions by small moduli instead, followed by compact S-boxes. The performance of our design varies for different fields we operate on, but is in the range of 2-9x overhead over the popular SHA-256.

Our approach to compactness. We tackle the prover time issue by providing an efficient lookup-based implementation of highly-nonlinear `Bars`, which is therefore one of *main contributions* of this submission. Concretely, it is the first primitive that is highly nonlinear, compact, and fast at the same time. For S-boxes of size $2^{9.5}$, we make only 126 lookups to process 510 bits of data, which is not far from the optimal $510/9.5 \approx 53$.

Comparison to other designs. When compared to the hash functions tailored to the same use cases, we are on par in the gate metric and are much faster in the native performance.

The performance can be improved in certain fields, and we show how to craft a prime to increase performance further. Even over generic prime fields (such as the scalar fields of the BLS12-381 or BN254 elliptic curves) `RC` is faster by a factor of 5 compared to POSEIDON and by a factor of 140 compared to RESCUE and 120 compared to RESCUE-Prime. Using specially crafted fields increases these factors to 16, 357, and 289 respectively. `RC` is, thereby, only by a factor of 5 slower than Blake2, the fastest traditional hash algorithm we benchmarked, but requires 7 times less gates when encoded into a circuit.

Compared to Pedersen hash/SINSEMILLA we provide pre-image resistance in addition to collision resistance. Also we rely on the public scrutiny rather than on (pre-quantum) hardness assumptions.

	Performance			
	Zero knowledge			Native
	R1CS eq-s	Plookup reg. gates	Area-degree product	(μs)
Poseidon	243	633	9495	19
Rescue	288	480	7200	480
Rescue-Prime	252	420	6300	415
Feistel-MiMC	1326	1326	19890	38
Griffin	96	186	2790	115
Neptune	228	1137	17055	20
SHA-256	27534	3000	60000	0.32
Blake2s	21006	2000	40000	0.21
Pedersen hash	869		13035	54
Sinsemilla		510	1530	137
Reinforced Concrete-BN/BLS	-	**378**	**5670**	**3.4**
Reinforced Concrete-ST	-	**360**	**5400**	**1.09**

Table 9: Performance of various hash functions in the zero knowledge (preimage proof) and native (hashing 512 bits of data) settings. All native benchmarks are ours (Section 9.8.2). Poseidon, Rescue, Rescue-Prime, Feistel-MIMC, Neptune, and Griffin gate counts are ours (Section 9.8.1). SHA-256 and Blake2s R1CS gate counts are from Hopwood's notes [Hop19], and their Plookup costs as well as the area-degree product is taken from the report by Williamson [Wil20]. Pedersen hash gate count is taken from the Zcash protocol [BHH+22], and the area-degree product is calculated using the same factor of 15 as for Poseidon. The Sinsemilla regular gate count by us is Section 9.8.1, whereas the area-degree optimized version is from [BH21].

From the design perspective, one can view the collision resistant but slower Sinsemilla as an alternative to the `Bars` layer, as both are not preimage resistant in isolation. Whether it is possible to take the best from both designs, remains the subject of future work.

Regarding security analysis, the new design offers reasonably big security margin against statistical attacks, but at the same time much bigger margin against algebraic attacks. Since the latter are less explored, we conclude that `RC` is more robust against possible breakthroughs in algebraic analysis. On the other hand, the most recent algebraic cryptanalysis of weakened Poseidon and Rescue-Prime [BBL+22] has proven to be memory-intensive and thus less practical than can be expected.

Supported proof systems. Whereas some ZK proof systems explicitly work with arithmetic gates (i.e. field additions and multiplications) only [PHG+13; Gro16], a number of protocols also support lookup tables. Those include Arya [BCG+18], Plookup [GW20; PFM+22], Halo2 [BHH+22], Cairo [GPR21]. As lookup gates also speed up traditional hash functions like SHA-2, we expect such protocols to become widespread in the near future.

Restrictions and Future Work. Whereas `RC` clearly brings high native and ZK performance, it also has its own restrictions. First of all, a proof system should support lookup gates, as otherwise the `RC` circuit would be quite big (we estimate it to be around 5000 constraints). Secondly the `Bars` component is specific for each field, which implies a bit of work when carrying it to a proof system with a new curve. Devising a more generic `Bars` is the subject of the future work. Another interesting direction is non-sponge instances of `RC`.

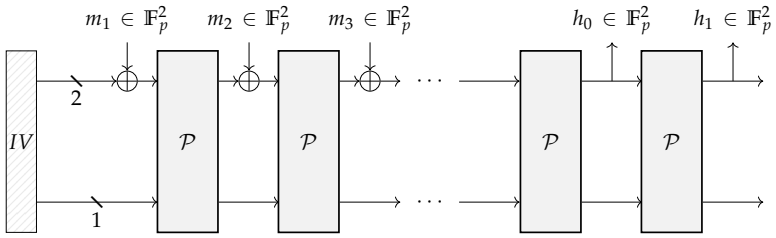


Figure 12: A sponge hash function with a fixed-size output. In our case IV is a 3-tuple of zero $\mathbb{F}_p$ elements, m_i are message chunks to be hashed (2 $\mathbb{F}_p$ elements each), $\oplus$ is the elementwise addition in the field, h_i are hash outputs.

Summary of the paper. We describe RC on a high level in Section 9.2. Then we give formal security definitions and claims regarding the security of RC in Section 9.3. A more detailed rationale and specification follows in Section 9.4. We proceed with a summary of our own cryptanalysis in Section 9.5 (which is detailed in Appendix). Then we present a constraint system (needed to build a circuit for ZK proofs) for RC and prove its correctness and soundness (Section 9.6). We conclude the main body of the paper with the benchmarks. Details of RC instances for different fields and details of cryptanalysis are presented in Appendix.

9.2 RC in a Nutshell

The RC hash function operates in the sponge framework (Fig. 12). The sponge converts a fixed length bijective function (called RC permutation) to a variable-length hash function, which is collision- and preimage-resistant as long as the underlying permutation does not exhibit any 'non-random' properties up to the bound defined by the security level 2^λ (in our case λ is universally set to 128).

The RC permutation illustrated in Fig. 13, can be considered as a modified 7-round SP network, where input, output and intermediate state elements are from $\mathbb{F}_p^3$ for a prime number p. More formally,

$$\begin{aligned}\mathtt{RC} := \mathtt{Concrete}^{(8)} &\circ \mathtt{Bricks} \circ \mathtt{Concrete}^{(7)} \\ &\circ \mathtt{Bricks} \circ \mathtt{Concrete}^{(6)} \circ \mathtt{Bricks} \\ &\circ \mathtt{Concrete}^{(5)} \circ \mathtt{Bars} \circ \mathtt{Concrete}^{(4)} \\ &\circ \mathtt{Bricks} \circ \mathtt{Concrete}^{(3)} \circ \mathtt{Bricks} \\ &\circ \mathtt{Concrete}^{(2)} \circ \mathtt{Bricks} \circ \mathtt{Concrete}^{(1)}\end{aligned}$$

In the following, we refer to $\mathtt{Concrete} \circ \mathtt{Bricks}$ as "round".

We define RC for different p, with two (-BN and -BLS) being scalar fields of the curves BN254 [Woo14] and BLS12-381[2] and another one (-ST) crafted for a specially chosen field in order to deliver the highest performance. We elaborate how to craft an instance in Section 9.4.

We reserve 1 field element for the capacity in sponge, thus aiming for the 128-bit security against collision and preimage attacks for all instances. A single call to RC thus suffices for a 2-to-1 compression function.

[2] https://electriccoin.co/blog/new-snark-curve/

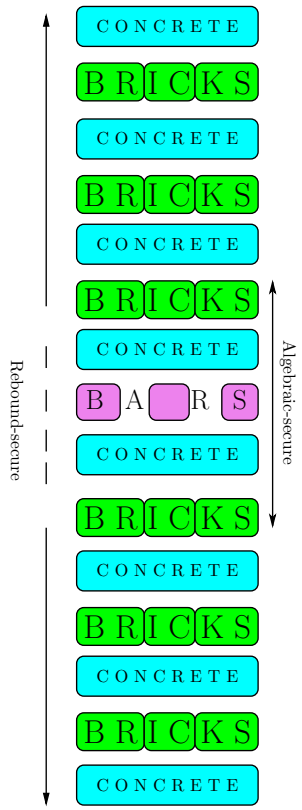


Figure 13: The RC permutation. The middle Br-C-B-C-Br part is secure against algebraic attacks whereas C-Br-C-Br-C-Br-C-Br-C-Br-C is secure against rebounds (more generally, statistical) attacks.

Design. The RC design depicted in Figure 13 is a modification of a traditional word-oriented SP-network (SPN) for constructing (keyed or keyless) cryptographic permutations. The RC design differs from a traditional SPN in two aspects:

- the middle layer of the SP network is replaced by a special component called Bars. This special component effectively reinforces the permutation against cryptanalytic approaches that would cover many more rounds without Bars. It does not admit a low-degree polynomial description but can be implemented as a circuit with reasonable costs in ZK.
- instead of applying independent non-linear transformations on single words, RC uses (low-degree) non-linear layers, called Bricks, that additionally mix different words. Bricks used the same construction as Horst [GHR+23]. It provides resistance against statistical cryptanalysis and is cheap in the zero knowledge, i.e. via gate counting.

The third component, Concrete, is an analog of the traditional affine layer but over $\mathbb{F}$. It ensures diffusion to make statistical or algebraic properties expand to the entire state, and is also cheap in ZK.

Layout. The Bricks and Concrete layers interleave exactly as in traditional SPN designs [DR02b]. As RC is used in a sponge framework, the Bricks components at either end would bring no security against collision or preimage

attacks, so we start and end with `Concrete`. The middle call to `Bricks` is replaced with `Bars`. The rationale behind putting all `Bar` into a single layer is that start-from-the-middle attacks are somewhat easier to find and thus we plan to detect them all in the design phase.

9.3 Security Requirements and Claims

Our high-level security claims, which determine the parameter selection for `RC`, are the following.

- For the sponge hash function with `RC`, we aim for a collision and preimage resistance up to 2^{128} field operations for 256-bit fields. We want to be able to instantiate a random oracle in protocols up to 2^{128} calls.
- For the authenticated encryption scheme using `RC`, we aim for confidentiality and integrity up to 2^{128} encrypted messages for 256-bit fields.
- When using the `RC` in other future schemes, we aim for a 1-element CICO security [GJM+11] up to 2^{128} field operations. More concretely, it should be infeasible to find such x_1, x_2, y_1, y_2 such that

$$\texttt{RC}(0, x_1, x_2) = (0, y_1, y_2)$$

9.4 Specification and Rationale

The story behind the design of `RC`, which has determined its inner components is as follows:

- We wanted to design a hash function which has a high degree as a polynomial and would not allow a treatment with algebraic methods such as Grobner basis.
- We were aware how table lookups can be used to implement hash functions that are highly non-linear and resistant to algebraic attacks – such as Blake2 and SHA-256. We seek to have similar functionality but applied to finite field elements rather than 32/64/128/256-bit words. For this we had to design an efficient way to decompose a field element into smaller chunks, apply some nonlinear transformation, and then wrap it back (composition). This was to become `Bars`.
- It turned out that in order to avoid overflows at composition, the nonlinear transformation within `Bars` should have a certain number of fixed points, and there must not be many of them for security. This yielded an heuristic method for finding a decomposition.
- In order to protect against non-algebraic attacks, we had to wrap `Bars` with additional confusion and diffusion layers. The number of those was derived from traditional attacks on SPN-based designs such as rebound [MRS+09].

9.4.1 The Bricks function

The function $\texttt{Bricks} : \mathbb{F}_p^3 \to \mathbb{F}_p^3$ is a non-linear permutation of degree $d = 5$ (with the requirement $\gcd(p-1, d) = 1$). Following [GHR+23], we define Bricks as

$$\texttt{Bricks}(x_1, x_2, x_3) = (x_1^d, x_2(x_1^2 + \alpha_1 x_1 + \beta_1), x_3(x_2^2 + \alpha_2 x_2 + \beta_2)),$$

where $\alpha_1, \alpha_2, \beta_1, \beta_2 \in \mathbb{F}_p$ such that $\alpha_i^2 - 4\beta_i$ is not a quadratic residue modulo p. We refer to [GHR+23, Section 3] for a proof regarding its invertibility, which relies on the fact that $z^2 + \alpha z + \beta \neq 0$ for each $z \in \mathbb{F}_p$.

9.4.2 The Concrete function

The function $\texttt{Concrete}^{(j)} : \mathbb{F}_p^3 \to \mathbb{F}_p^3$ denotes the multiplication of the state by a 3×3 MDS matrix $M = \mathrm{circ}(2, 1, 1)$ with subsequent addition of the j-th round constant vector $c^{(j)} \in \mathbb{F}_p^3$, that is

$$\texttt{Concrete}^{(j)}(x) := \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + c^{(j)}.$$

Note that M is invertible and MDS for each $p \geq 3$. The elements $c_1^{(j)}, c_2^{(j)}, c_3^{(j)}$ are certain pseudo-random constants, generated using e.g. SHAKE-128 with rejection sampling.

9.4.3 The Bars Function

The function $\texttt{Bars} : \mathbb{F}_p^3 \to \mathbb{F}_p^3$ is defined as

$$\texttt{Bars}(x_1, x_2, x_3) = (\texttt{Bar}(x_1), \texttt{Bar}(x_2), \texttt{Bar}(x_3)).$$

The function $\texttt{Bar} : \mathbb{F}_p \to \mathbb{F}_p$ is designed to be a permutation of $\mathbb{F}_p$ coming from n smaller permutations acting *independently* on n smaller domains $\mathbb{Z}_{s_1}, \ldots, \mathbb{Z}_{s_n}$, where $s_1, \ldots, s_n$ are defined for each prime p separately, see Section 9.7. The independence requirement is crucial for the performance of Bar. For this we decompose a field element $x \in \mathbb{F}_p$ into n smaller *digits* $x_1, \ldots, x_n$ with $x_i \in \mathbb{Z}_{s_i}$ with the function Comp, and then compose it back with Decomp. Overall, $\texttt{Bar} : \mathbb{F}_p \to \mathbb{F}_p$ is defined as

$$\texttt{Bar} = \texttt{Comp} \circ \texttt{SBox} \circ \texttt{Decomp}. \tag{58}$$

In the following, we define all these components. The invertibility of Bar is proved in Section 9.A.

Decomposition and Composition

We choose the standard representation $\mathbb{F}_p = \{0, 1, \ldots, p-1\}$ for $\mathbb{F}_p$, thus identifying an element $x \in \mathbb{F}_p$ with an integer $0 \leq x \leq p-1$. Our decomposition $\texttt{Decomp} : \mathbb{F}_p \to \mathbb{Z}_{s_1} \times \ldots \times \mathbb{Z}_{s_n}$ expands $x \in \mathbb{F}_p$ as

$$\begin{aligned} x = x_1 \cdot s_2 s_3 \cdots s_n + x_2 \cdot s_3 s_4 \cdots s_n + \cdots \\ + x_{n-1} \cdot s_n + x_n = \sum_{i=1}^{n} x_i \prod_{j>i} s_j. \end{aligned}$$

with $0 \leq x_i < s_i$ and where the s_i are chosen such that $\prod_{i=1}^{n} s_i > p$. The digits $x_i \in \mathbb{Z}_{s_i}$ are determined similarly to ordinary base-b expansion:

$$\begin{aligned} x_n &:= x \bmod s_n, \\ x_i &:= \frac{x - \sum_{j>i} x_j \prod_{k>j} s_k}{\prod_{j>i} s_j} \bmod s_i. \end{aligned} \tag{59}$$

It follows directly from the definition in Eq. (59) that the digits x_i are *unique*. Because of the strong analogy with ordinary base-b expansion and for ease of notation in the following part, we define for $1 \leq i \leq n$ the elements

$$b_i := \prod_{j>i} s_j = s_{i+1} s_{i+2} \dots s_n,$$

where b_n is defined by the empty product and thus $b_n := 1$. The inverse process, the composition $\texttt{Comp} : \mathbb{Z}_{s_1} \times \dots \times \mathbb{Z}_{s_n} \to \mathbb{F}_p$ is computed as

$$\texttt{Comp}(y_1, \dots, y_n) := \sum_{i=1}^{n} y_i b_i \bmod p. \tag{60}$$

SBox

Let $(v_1, v_2, \dots, v_n) = \text{Decomp}(p-1)$ and let $p' \leq \min_{1 \leq i \leq n} v_i$. Then x_i is converted as follows:

$$y_i := S(x_i) = \begin{cases} f(x_i) & \text{if } x_i < p', \\ x_i & \text{if } x_i \geq p', \end{cases} \tag{61}$$

where f denotes a permutation of $\mathbb{Z}_{p'}$. In Lemma 12 we prove that Bar is indeed a permutation of $\mathbb{F}_p$. The value p' is selected for each p separately.

The f function is derived from the MiMC cipher (which implicitly requires p' being prime). Reference values of p' for various p and tables for f are given in full in the Appendix.

9.4.4 Sponge framework parameters

We suggest the bijective transformation RC being used in the sponge framework [BDP+08] similarly to Poseidon [GKK+19] and Rescue [AAB+20]. The parameters are as follows:

- Rate is two $\mathbb{F}_p$ elements, capacity is one $\mathbb{F}_p$ element.
- Claimed preimage and collision security level of 128 bits.
- The padding rule is simply to add the 0 element to any input of odd length. The very first capacity value is initialized by the length-depending constant, e.g. just length l. This does not violate the sponge security proof as long as only short lengths (say up to 2^{32}) are allowed.

9.5 Security Analysis

In this section we summarize our own analysis of RC security and connect it with the requirements outlined in Section 9.3.

For the latter, we customarily reduce the security of RC hash to its resistance against known cryptanalytic attacks. In particular, we focus on

the following two classes of attacks, respectively statistical and algebraic attacks. As already mentioned in the introduction, we make use of the HADES/POSEIDON design strategy in order to provide security:

- Statistical attacks (including differential, linear, rebound, truncated, impossible, MiTM, boomerang) cannot be mounted on RC even with the middle component Bricks-Concrete-Bars-Concrete-Bricks replaced with a single Bricks layer up 2^{128} field operations.
- The middle component Bricks-Concrete-Bars-Concrete-Bricks resists invariant subspace and algebraic (e.g., Gröbner basis) attacks up to 2^{128} field operations. Due to the high degree and because we are working over prime fields, we also expect ample resistance against higher-order differential attacks (e.g., zero-sum distinguishers or cube attacks).

We provide a detailed overview of our cryptanalysis in the Appendix of the full version of this article under the link https://eprint.iacr.org/2021/1038.pdf. The short summary is the following:

- Differential and linear attacks do not work as long as the Bricks layer is involved.
- We cannot mount rebound attacks for 5 or more rounds thus having at least 2 rounds of security margin.
- No invariant subspace attacks have been found.
- Groebner basis cryptanalysis fails at greatly weakened versions (10-bit fields) already.

9.6 Lookup Tables and System of Constraints for Bar

In this section we create tables and a set of constraints such that for $x, y \in \mathbb{F}_p$ it holds $y = \texttt{Bar}(x)$ if and only if this set of constraints is satisfied. We face two challenges:

1. The S-box S_i acts on a domain of size s_i, which makes each S-box potentially unique. If we specify the behavior of each S-box separately, the table would have $\sum_i s_i$ entries, which renders it inefficient.
2. Since $\prod_i s_i > p$, there exist distinct elements $(x_1, \ldots, x_n) \neq (x'_1, \ldots, x'_n)$ in $\mathbb{Z}_{s_1} \times \ldots \mathbb{Z}_{s_n}$ that produce the same $x \in \mathbb{F}_p$, i.e., for which it holds

$$x = \texttt{Comp}(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i b_i \bmod p =$$
$$= \sum_{i=1}^{n} x'_i b_i \bmod p = \texttt{Comp}(x'_1, \ldots, x'_n).$$

   We have to ensure that our table and set of constraints prevents this collision from happening.

We address these challenges with two additional sets of variables

$$(z_1, \ldots, z_n) \text{ and } (c_1, \ldots, c_n),$$

respectively. The variable z_i encodes if $x_i < p'$ (S_i is non-linear function) or $x_i \geq p'$ (S_i is identity function) and is defined as

$$z_i := \begin{cases} 0, \text{ if } x_i < p'; \\ 1, \text{ if } x_i \geq p'. \end{cases} \tag{62}$$

The purpose of variables $(c_1, \ldots, c_n)$ is to indicate if a tuple $(x_1, \ldots, x_n) \in \mathbb{Z}_{s_1} \times \ldots \times \mathbb{Z}_{s_n}$ has the property $\sum_{i=1}^{n} x_i b_i \geq p$, or not. If $\sum_{i=1}^{n} x_i b_i \geq p$, the tuple $(x_1, \ldots, x_n)$ "overflows" p and thus it is a potential candidate for a collision since by definition composition is unique for all $(x_1, \ldots, x_n)$ with $\sum_{i=1}^{n} x_i b_i < p$. With our set of constraints we need to exclude all those tuples "overflowing" p. For $(v_1, \ldots, v_n) = \texttt{Decomp}(p-1)$, we therefore define

$$c_i := \begin{cases} 0, \text{ if } x_j = v_j \text{ for all } 1 \leq j \leq i; \\ 1, \text{ if } x_i < v_i; \\ 2, \text{ if } x_i \geq v_i \text{ and } x_j \neq v_j \text{ for some } 1 \leq j \leq i; \end{cases} \tag{63}$$

By definition of c_i, only sequences $c_1, c_2, \ldots, c_n$ of length n output by the finite-state automaton $\mathcal{A}$ in Fig. 14 are allowed; they characterize all tuples $(x_1, \ldots, x_n) \in \mathbb{N}^n$ with $\sum_{i=1}^{n} x_i b_i < p$.

We create the following 4-ary tables for our set of constraints:

- Table T_2 contains all binary sequences of length 4 (Fig. 15) thus providing a means to encode all possible sequences $(z_1, \ldots, z_n)$ by concatenating as many 4-ary sequences as needed;
- Table T_3 contains all outputs of length 4 of the finite-state automaton $\mathcal{A}$ in Fig. 14. They are chained together with the last element of one 4-ary sequence matching the first element of the next 4-ary sequence to encode all possible outputs of $\mathcal{A}$ of length n, see constraints (65),(66);
- Table T_1 encodes the output of the S-Boxes $S_1, \ldots, S_n$ and indicates whether for an input to S-Box S_i the non-linear function f or the identity function is applied (Fig. 16).

We claim that $y = \texttt{Bar}(x)$ holds if and only if for $x, y \in \mathbb{F}_p$ and $(x_1, \ldots, x_n)$, $(y_1, \ldots, y_n)$ in $\mathbb{N}^n$ the following constraints are satisfied:

$$\forall n \geq i \geq 1 : \ (x_i, i \cdot z_i, y_i, c_i) \in T_1, \tag{64}$$

$$\forall \lceil (n-1)/3 \rceil - 1 \geq i \geq 1 : \quad (c_{3i-2}, c_{3i-1}, c_{3i}, c_{3i+1}) \in T_3, \tag{65}$$

$$(c_{n-3}, c_{n-2}, c_{n-1}, c_n) \in T_3, \tag{66}$$

$$\forall \lceil n/4 \rceil - 1 \geq i \geq 1 : \quad (z_{4i-3}, z_{4i-2}, z_{4i-1}, z_{4i}) \in T_2, \tag{67}$$

$$(z_{n-3}, z_{n-2}, z_{n-1}, z_n) \in T_2, \tag{68}$$

$$x = \sum_{i=1}^{n} x_i b_i \bmod p, \tag{69}$$

$$y = \sum_{i=1}^{n} y_i b_i \bmod p. \tag{70}$$

In particular, we claim for $x \in \mathbb{F}_p$ there doesn't exist any collision in $\mathbb{Z}_{s_1} \times \ldots \mathbb{Z}_{s_n}$. I.e., there is exactly one element $(x_1, \ldots, x_n)$ in $\mathbb{Z}_{s_1} \times \ldots \mathbb{Z}_{s_n}$ with

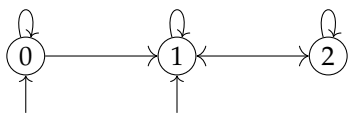


Figure 14: Finite-state automaton $\mathcal{A}$ representing all valid sequences $c_1, c_2, \ldots, c_n$.

$$T_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ \dots & & & \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix},$$

Figure 15: Lookup Table T_2.

$$T_1 = \begin{bmatrix} 0 & 0 & f(0) & 1 \\ 1 & 0 & f(1) & 1 \\ \dots & & & \\ p'-1 & 0 & f(p'-1) & 1 \\ p' & 1 & p' & 1 \\ p'+1 & 1 & p'+1 & 1 \\ \dots & & & \\ v_1-1 & 1 & v_1-1 & 1 \\ v_1 & 1 & v_1 & 0 \\ p' & 2 & p' & 1 \\ \dots & & & \\ v_2-1 & 2 & v_2-1 & 1 \\ v_2 & 2 & v_2 & 0 \\ v_2 & 2 & v_2 & 2 \\ v_2+1 & 2 & v_2+1 & 2 \\ \dots & & & \\ s_2-1 & 2 & s_2-1 & 2 \\ \dots & & & \\ p' & n & p' & 1 \\ \dots & & & \\ v_n-1 & n & v_n-1 & 1 \\ v_n & n & v_n & 0 \\ v_n & n & v_n & 2 \\ v_n+1 & n & v_n+1 & 2 \\ \dots & & & \\ s_n-1 & n & s_n-1 & 2 \end{bmatrix}, \quad T_3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 2 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 0 & 1 & 2 & 1 \\ 0 & 1 & 2 & 2 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 2 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 2 & 2 \\ 1 & 2 & 1 & 1 \\ 1 & 2 & 1 & 2 \\ 1 & 2 & 2 & 1 \\ 1 & 2 & 2 & 2 \\ 2 & 1 & 1 & 1 \\ 2 & 1 & 1 & 2 \\ 2 & 1 & 2 & 1 \\ 2 & 1 & 2 & 2 \\ 2 & 2 & 1 & 1 \\ 2 & 2 & 1 & 2 \\ 2 & 2 & 2 & 1 \\ 2 & 2 & 2 & 2 \end{bmatrix}$$

Figure 16: Lookup Tables T_1 and T_3.

$\texttt{Comp}(x_1, \dots, x_n) = x$. We prove these assertions in Lemma 10 and Lemma 11. As a result, the total number of lookup constraints is

$$n + \lceil (n-1)/3 \rceil + \lceil n/4 \rceil \approx n + n/3 + n/4 \approx 1.59n$$

table lookups with tables of total size $p' + \sum_i (s_i - p' + 1) + 16 + 23$.

9.6.1 Soundness and Completeness

Lemma 10. *The set of constraints (64) – (70) is complete, i.e., for any $x, y \in \mathbb{F}_p$ with $y = \texttt{Bar}(x)$ it is possible to construct $\{x_i, y_i, c_i, z_i : 1 \le i \le n\}$ that satisfy them.*

Proof. We work with the standard representation of $\mathbb{F}_p$, that is, $\mathbb{F}_p = \{0, 1, \dots, p-1\}$. Suppose for $x, y \in \mathbb{F}_p$ it holds $y = \texttt{Bar}(x)$. Our proof works as follows:

1. We construct x_i, y_i and show that constraints (69) and (70) are satisfied;
2. we define z_i that satisfy constraints (67) and (68) regarding Table T_2;

3. we define c_i that satisfy constraints (65) and (66) regarding Table T_3;
4. we show that $(x_i, i \cdot z_i, y_i, c_i)$ satisfy the constraints (64) regarding Table T_1.

1st Step. We define

$$(x_1, \ldots, x_n) := \texttt{Decomp}(x)$$

and

$$(y_1, \ldots, y_n) := \texttt{SBox}(x_1, \ldots, x_n) = (\texttt{SBox} \circ \texttt{Decomp})(x);$$

then constraint (69) holds by definition of Decomp and constraint (70) by definition of Bar, i.e.,

$$\begin{aligned} y &= (\texttt{Comp} \circ \texttt{SBox} \circ \texttt{Decomp})(x) \\ &= \texttt{Comp}\,(\texttt{SBox} \circ \texttt{Decomp}(x)) \\ &= \texttt{Comp}(y_1, \ldots, y_n) = \sum_{i=1}^{n} y_i b_i \bmod p. \end{aligned}$$

2nd Step. Let p' be according to the definition of the Bar function, i.e., p' is the largest prime smaller than or equal to $v = \min_{1 \le i \le n} v_i$, where $(v_1, \ldots, v_n) = \texttt{Decomp}(p-1)$. For $1 \le i \le n$ we define

$$z_i := \begin{cases} 0, & \text{if } x_i < p'; \\ 1, & \text{if } x_i \ge p'; \end{cases}$$

that indicate if $x_i < p'$ or $x_i \ge p'$. The sequence $(z_1, \ldots, z_n)$ is a binary sequence of length n, where all 2^n combinations are possible: every digit x_i can be strictly smaller or greater than p'. Since T_2 contains all binary sequences of length 4, we have that the constraints (67) and (68) regarding T_2 are satisfied .

3rd Step. If $x = p - 1$, or equivalently, if $x_i = v_i$ for all $1 \le i \le n$, we define $c_i := 0$, for all $1 \le i \le n$. Thus $(c_1, \ldots, c_n) = (0, \ldots, 0)$ and the corresponding constraints (65) and (66) in Table T_3 are satisfied. If $x < p - 1$, there exists at least one index $1 \le i \le n$ with $x_i < v_i$. Let j be the minimal index with that property. We set

$$c_i := \begin{cases} 0, & \text{if } i < j; \\ 1, & \text{if } i \ge j \text{ and } x_i < v_i; \\ 2, & \text{if } i > j \text{ and } x_i \ge v_i. \end{cases}$$

Note that the case $i = j$ and $x_i \ge v_i$ cannot happen, since this would on the one hand mean $x_j \ge v_j$ and on the other hand $x_j < v_j$ (by definition of j), a contradiction. Thus, the above three cases cover all possible situations regarding i. Next, we list all subsequences of $c_1, \ldots, c_n$ that are *not* possible:

(a) $(2, \ldots)$; since $c_1 = 2$ this would mean $1 \le j < i = 1$, a contradiction.

(b) $(\ldots, 0, 2, \ldots)$; this would imply $i < j$ ($c_i = 0$) and $i + 1 > j$ ($c_{i+1} = 2$), a contradiction.

(c) $(\ldots, 1, 0, \ldots)$; a contradiction, since $i \ge j$ ($c_i = 1$) and $i + 1 < j$ ($c_{i+1} = 0$).

(d) $(\ldots, 2, 0, \ldots)$; a contradiction, since $i > j$ ($c_i = 2$) and $i + 1 < j$ ($c_{i+1} = 0$).

We explicitly note, all other subsequences are valid. In a next step, we model a finite-state automaton $\mathcal{B}$ whose outputs of length n characterize all possible sequences $(c_1, \ldots, c_n)$. Clearly, $\mathcal{B}$ has the states $0, 1, 2$ with only $0, 1$ being accepting states: due to (a) no sequence can start with 2. According to (b), (c) and (d), all possible transitions are given by

$$\{(0,0), (0,1), (1,1), (1,2), (2,1), (2,2)\}.$$

But this means, that automaton $\mathcal{B}$ is identical to automaton $\mathcal{A}$ depicted in Fig. 14. Hence we conclude, all possible sequences $(c_1, \ldots, c_n)$ of elements as defined above are precisely the outputs of length n of the finite-state automaton $\mathcal{A}$. If we divide the sequence $(c_1, \ldots, c_n)$ into chunks of 4 elements such that the last element of one chunk matches the first element of the next chunk, we see that constraints (65) and (66) regarding T_3 are satisfied.

4th Step. Constraints (64) regarding T_1 are satisfied as well: by definition of x_i, z_i, y_i, c_i we have $0 \leq x_i \leq s_i - 1$, $z_i \in \{0,1\}$, $y_i = S_i(x_i)$ and $c_i \in \{0,1,2\}$, respectively. This means, the domains of $x_i, i \cdot z_i, y_i, c_i$ agree with the general conditions in T_1. Not all combinations are allowed, however. The following arguments show that indeed all possible 4-ary chunks $(x_i, i \cdot z_i, y_i, c_i)$ satisfy the constraints in T_1. As in the *3rd Step*, for $x = p - 1$ we define $c_i := 0$ and thus have $(x_i, i \cdot z_i, y_i, c_i) = (v_i, i, v_i, 0)$ for $1 \leq i \leq n$. Hence, for $x = p - 1$ the corresponding constraints (64) in Table T_1 are satisfied. Therefore let $x < p - 1$ and let again j be the minimal index with $x_i < v_i$.

- For $0 \leq x_i < p'$, we have $z_i = 0$, $i \cdot z_i = 0$, $y_i = S(x_i) = f(x_i)$ and $c_i = 1$ (since $x_i < p' \leq v_i$) by construction of x_i, z_i, y_i and c_i, respectively. Thus the first p' constraints in T_1 are satisfied.

- For $p' \leq x_i = v_i$ two cases can happen: if $i < j$, then $c_i = 0$; if $i > j$, then $c_i = 2$. In both cases the corresponding 4-ary chunk $x_i, i \cdot z_i = i, y_i = x_i, c_i \in \{0,2\}$ is contained in T_1. We note, the case $x_i = v_i$ and $i = j$ cannot happen due to the definition of j.

- For $p' \leq x_i < v_i$, we have $z_i = 1$, $i \cdot z_i = i$, $y_i = S(x_i) = x_i$ and $c_i = 1$ (since $x_i < v_i$). Thus the corresponding $v_i - p'$ constraints in T_1 are satisfied.

- For $v_i + 1 \leq x_i \leq s_i - 1$ it holds $z_i = 1$, $i \cdot z_i = i$, $y_i = S(x) = x_i$ and $c_i = 2$, which shows that the corresponding $s_i - v_i - 1$ constraints in T_1 are fulfilled.

Specifically, for $i = 1$ there is no entry $(x_1, i \cdot z_1, y_1, 2)$ in T_1, therefore we have to argue that this case cannot happen; this is clear, however, since we have already shown that automaton $\mathcal{B}$, which represents all valid sequences $(c_1, \ldots, c_n)$, guarantees $c_1 \in \{0,1\}$. □

Lemma 11. *The set of constraints (64)–(70) is sound, i.e., for any $x, y \in \mathbb{F}_p$ and any $\{x_i, y_i, z_i, c_i \in \mathbb{N} : 1 \leq i \leq n\}$ that satisfy them all it holds $y = \texttt{Bar}(x)$.*

Proof. We work with the standard representation of $\mathbb{F}_p$. For $\mathcal{R} := \mathbb{Z}_{s_1} \times \ldots \times \mathbb{Z}_{s_n}$ let

$$\mathcal{R}_{<p} := \{(z_1, \ldots, z_n) \in \mathcal{R} : \sum_{i=1}^{n} z_i b_i < p\}.$$

Our proof consists of the following parts:

1. Show that $(x_1, \ldots, x_n)$ is a valid decomposition of x, i.e., $(x_1, \ldots, x_n) = \texttt{Decomp}(x)$.
2. Show that for all $1 \leq i \leq n$ we have $y_i = S_i(x_i)$ according to (61) and deduce $(y_1, \ldots, y_n) = (\texttt{SBox} \circ \texttt{Decomp})(x)$.
3. Use the above two facts and deduce $y = \texttt{Bar}(x)$.

1st Step. Let $(x'_1, \ldots, x'_n) := \texttt{Decomp}(x)$ and $\hat{x} := \sum_{i=1}^{n} x_i b_i$. Suppose $\hat{x} < p$, or in other words $(x_1, \ldots, x_n) \in \mathcal{R}_{<p}$. Then by (69) we have $\hat{x} = \hat{x} \bmod p = \sum_{i=1}^{n} x_i b_i \bmod p = x < p$, and thus

$$\begin{aligned}\texttt{Decomp}(x) &= \texttt{Decomp}\left(\sum_{i=1}^{n} x_i b_i \bmod p\right) \\ &= (\texttt{Decomp} \circ \texttt{Comp})(x_1, \ldots, x_n) = (x_1, \ldots, x_n).\end{aligned}$$

The last equality uses the fact, that Decomp and Comp are inverse to each other on $\mathcal{R}_{<p}$ and $\mathbb{F}_p$; we proved this in more detail in Lemma 12.

We show that the case $\hat{x} \geq p$ leads to a contradiction. For this, suppose $\hat{x} \geq p$. This implies that there exists $1 \leq k \leq n$ with

$$x_i = v_i \text{ for all } 1 \leq i < k \text{ and } x_k > v_k.$$

Note that $k > 1$ as $x_1 \leq v_1$ by Table T_1 (constraint (64)). Also, by constraint (64) it holds $c_i \in \{0, 2\}$ for all $1 \leq i < k$ and in particular $c_1 = 0$. Therefore, constraints (65) and (66) regarding Table T_3 ensure that actually all $c_i = 0$ for $1 \leq i < k$ since there is no sequence with $(\ldots, 0, 2, \ldots)$ in T_3. Therefore, again by constraints (65) and (66), we have that $c_k \in \{0, 1\}$. By constraint (64) this is only possible if $x_k \leq v_k$. A contradiction.

2nd Step. Let $1 \leq i \leq n$. We show $y_i = S(x_i)$. By constraints (67) and (68) it holds $z_i \in \{0, 1\}$. If $z_i = 0$ then $i \cdot z_i = 0$ and by constraint (64) we have $x_i < p'$ and $y_i = f(x_i)$. If $z_i = 1$, we have $i \cdot z_i = i > 1$, and again by constraint (64) it holds $x_i \geq p'$ and $y_i = x_i$. Altogether we get that $y_i = S_i(x_i)$ and thus

$$\begin{aligned}(y_1, \ldots, y_n) &= \texttt{SBox}(x_1, \ldots, x_n) \\ &\overset{\text{Part1}}{=} \texttt{SBox}(\texttt{Decomp}(x)) = (\texttt{SBox} \circ \texttt{Decomp})(x).\end{aligned} \tag{71}$$

3rd Step. For the last part we use the definition of Bar, Part 2, the definition of Comp and constraint (70), which yields

$$\begin{aligned}\texttt{Bar}(x) &\overset{(58)}{=} (\texttt{Comp} \circ \texttt{SBox} \circ \texttt{Decomp})(x) \\ &= \texttt{Comp}(\texttt{SBox} \circ \texttt{Decomp}(x)) \\ &\overset{\text{Part 2}}{=} \texttt{Comp}(y_1, \ldots, y_n) \\ &\overset{(60)}{=} \sum_{i=1}^{n} y_i b_i \bmod p \overset{(70)}{=} y.\end{aligned}$$

□

9.7 Concrete Instances

The values of $\alpha_1, \alpha_2, \beta_1, \beta_2$ are given by

- $p = p_{\text{BLS381}}$: (1,3,2,4).
- $p = p_{\text{BN254}}$: (1,3,2,4)

- $p = p_{ST}$: (1,2,3,4).

For the `Bar` function we choose a decomposition into $n = 27$ small S-boxes for p being the order of BLS12-381 or BN254 curves.

BLS12-381. The prime p is given by

$$p_{\text{BLS381}} = \texttt{0x73eda753299d7d483339d80809a1d80}$$
$$\texttt{553bda402fffe5bfeffffffff00000001}.$$

The bucket sizes

$$\begin{matrix} s_{27}, & s_{26}, & \ldots, & s_{19}, \\ s_{18}, & s_{17}, & \ldots, & s_{10}, \\ s_9, & s_8, & \ldots, & s_1, \end{matrix}$$

for the `Bar` layer are given by

$$\begin{matrix} 693, & 696, & 694, & 668, & 679, & 695, & 691, & 693, & 700, \\ 688, & 700, & 694, & 701, & 694, & 699, & 701, & 701, & 701, \\ 695, & 698, & 697, & 703, & 702, & 691, & 688, & 703, & 679. \end{matrix}$$

If $(v_1, \ldots, v_{27})$ denotes the decomposition of $p - 1$, the largest prime p' smaller than or equal to $v = \min_{1 \le i \le 27} v_i$ is $p' = 659$. The values s_i were found by a variant of branch-and-bound process where we recursively determine from s_{27} to s_1 under the constraint that $s_i - v_i$ is not too large for any i.

BN254. The prime p is given by

$$p_{\text{BN254}} = \texttt{0x30644e72e131a029b85045b68181585}$$
$$\texttt{d2833e84879b9709143e1f593f0000001}.$$

The bucket sizes for the `Bar` layer are given by

$$\begin{matrix} 651, & 658, & 656, & 666, & 663, & 654, & 668, & 677, & 681, \\ 683, & 669, & 681, & 680, & 677, & 675, & 668, & 675, & 683, \\ 681, & 683, & 683, & 655, & 680, & 683, & 667, & 678, & 673. \end{matrix}$$

If $(v_1, \ldots, v_{27})$ denotes the decomposition of $p - 1$, the largest prime p' smaller than or equal to $v = \min_{1 \le i \le 27} v_i$ is $p' = 641$. Decomposition was found in the same way.

Special prime. We have crafted a special prime for the proof systems that are not elliptic curve based, so that the decomposition and modular reduction are extremely fast. Concretely, we found out that a 250-bit prime

$$p_{ST} = 0\text{x}3fa000 \ldots 001$$

admits the following representation:

$$p_{ST} = 2^{250} - 3 \cdot 2^{241} + 1 = \sum_{i=0}^{24} (2^{10} - 6) 2^{10i} + 1,$$

i.e.,

$$s_2 = s_3 = \cdots = s_{24} = 1024, \tag{72}$$
$$s_{25} = 1023, v_1 = v_2 = \cdots = v_{25} = 1018. \tag{73}$$

For this decomposition we first selected s_i to be almost all powers of two, prepared constraints that $(p - 1)$ is divisible by 2^{30} for Discrete Fourier Transform, and then tried a few values for v_i until we find a prime.

9.8 Performance

In this section we consider performance of plain and zero knowledge (circuit) implementations of RC. As the application, we consider a single call to permutation RC, which corresponds to hashing of two $\mathbb{F}$ elements, or computing one node of a Merkle tree.

9.8.1 Proof System Performance

Circuit metrics

So far many circuit implementations of hash functions are tailored to the proof system implementation they will be used, so it is extremely difficult to compare apples to apples by just measuring prover time. This is more complicated for proof systems that support lookups as only reference implementations are available[3].

Thus we turned to different metrics. First one just count gates and assumes that there are two types of gates: an arithmetic gate and a lookup gate, with the former implementing a quadratic constraint of form

$$a_1x_1x_2 + a_3x_3 + a_4x_4 + a_5x_5 = a_6$$

with x_i being witness variables and a_i being values of selector polynomials. It can handle a 2-ary addition. A lookup gate has form

$$(x_1, x_2, x_3, x_4) \in T$$

where T is the lookup table. These two gates are the ones defined in the Plonk and Plookup papers [GWC19; PFM+22] and thus we call it *regular gates* metric.

The second metric applies to custom gates, which implement arbitrary polynomial lookup constraints, and attempts to estimate the prover cost by assuming it is approximated as

$$C_{prover} \sim (\text{number of gates}) \times (\text{max degree of a gate constraint}) \times (\text{gate arity})$$

We call it *area-degree* product. The maximum degree of a regular gate constraint is 3, the arity is 5, so each gate contributes with cost 15.

Measuring hash functions

RC. The regular gates count for the BLS/BN primes.

- `Bricks`: 8 gates per round (7 for p_{ST} with $d = 3$);
- `Concrete`: 2 gates per element, 6 per round.
- `Bars`: 94 gates per element, 282 per round
  - decomposition: 26 gates
  - composition: 26 gates
  - table: 42 gates.

Total: $8 \cdot 6 + 6 \cdot 8 + 282 = 378$ gates to process two $\mathbb{F}_p$ elements of data. The p_{ST} case uses only 25 s_i so the total number of gates is 360. The area-degree product is thus $378 \cdot 15 = 5670$.

[3] E.g. Plonkup https://github.com/dusk-network/plonkup

Poseidon. Poseidon-128 [GKK+19] with 2 inputs, which needs 633 gates for the same setting: each full round needs 9 quadratic gates and 6 addition gates, whereas each partial round needs 3 quadratic and 6 addition gates. Total count is $15 \cdot 8 + 57 \cdot 9 = 633$.

Rescue. RESCUE with 2 inputs requires 16 full founds, which together utilize 288 quadratic gates. In addition, each (out of 16) round carries two matrix multiplications, i.e. $2 \cdot 6$ addition gates per round. The total regular gate count is then 480. RESCUE-Prime, a new variant of Rescue, requires only 14 rounds and, thus, is 12% cheaper.

Sinsemilla. SINSEMILLA is parameterized by k that determines the lookup table length 2^k and the same number of EC generators $P_0, P_1, \ldots, P_{2^k-1}$. A hash of tk-bit $M = (M_1, M_2, \ldots, M_t), t < 254$ is defined as

$$H(M) = (Q + \sum_{i \leq t} [2^{t-i}] P_{M_i})_x,$$

where Q is some EC point, $+$ is EC addition, $[a]$ is the EC scalar multiplication by a, $()_x$ is the x-coordinate of the curve.

To make a regular gate measurement, we take their system [BH21] of $5t$ quadratic equations and a single t-ary addition of message decomposition. Measuring in regular gates, we obtain that SINSEMILLA needs $9t$ arithmetic gates, and t lookup gates. For $k = 10$ and $t = 51$ we obtain 510-bit message input, for which the total gate count is about 510 regular gates.

The authors also provide an optimized version with 51 custom gates of degree 6 and arity 5. This yields the area-degree product of $51 \cdot 6 \cdot 5 = 1530$.

9.8.2 Plain Implementation Performance

We implemented `RC` in pure Rust using the ff_ce library[4] for field operations. Further, we re-implemented POSEIDON, RESCUE, and GRIFFIN with a statesize of 3 words, NEPTUNE using a statesize of 4 words, and Feistel-MiMC using ff_ce to compare them to `RC` in a fair setting. We further compare `RC` to pure Rust implementations of traditional hash algorithms[5], and compare it to SINSEMILLA using an implementation found in the Zcash/Orchard repository on Github[6], and to a Pedersen Hash implementation from librustzcash[7]. We benchmark input sizes of at least 512-bit (i.e., two field elements in `RC`). We, thus, benchmark one permutation call for all symmetric hash functions, except for Feistel-MiMC for which we require two. All benchmarks were obtained on a Linux Desktop PC with an Intel i7-4790 CPU (3.6 GHz) and 16 GB RAM using stable Rust version 1.58 and the `target-cpu=native` flag. The resulting benchmarks can be found in Section 9.8.2, code to reproduce them is publicly available at [Has].

As Section 9.8.2 shows, the plain performance of `RC` highly depends on the choice of the prime field, more specifically, how elements can be decomposed. The `Bars`-layer for p_{ST} can be evaluated by using only one big-integer division[8], whereas a generic decomposition, i.e., for p_{BN254} and p_{BLS12}, requires significantly more. The result is a runtime difference by a

[4] https://docs.rs/ff_ce/0.13.1/ff_ce/
[5] https://github.com/RustCrypto/hashes
[6] https://github.com/zcash/orchard, uses lookup tables to speed up performance.
[7] https://github.com/zcash/librustzcash
[8] We implemented divisions using precomputed reciprocals for all prime fields.

Hashing algorithm		BN	BLS	ST
	ns	*ns*	*ns*	*ns*
RC	-	3 419	3 538	1 087
Concrete Layer	-	39.1	39.5	34.2
Bricks Layer	-	172.4	188.0	101.67
Bars Layer	-	2 063	2 062	204.9
POSEIDON	-	19 944	20.423	18 185
RESCUE	-	470 030	498 210	388 430
RESCUE-Prime	-	408 780	431 130	314 660
Feistel-MiMC	-	37 980	39 883	31 894
GRIFFIN	-	113 670	120 450	90 455
NEPTUNE	-	20 265	20 453	18 825
SINSEMILLA	137 600	-	-	-
Pedersen Hash	54 027	-	-	-
SHA-256	319.1	-	-	-
Blake2b	189.6	-	-	-
Blake2s	213.3	-	-	-
SHA3-256	419.2	-	-	-

Table 10: Plain performance comparison in nano-seconds (ns) of different hash functions over prime fields with primes $p_{\text{BN}_{254}}, p_{\text{BLS}_{381}}, p_{\text{ST}}$. Implemented in Rust.

factor of 3 for the total hashing time. Compared to the previous state-of-the-art one can observe that RC is significantly faster. More concretely, RC is faster than the previously fastest hash function over finite fields (i.e., POSEIDON) by a factor of 5 for p_{BN254} and p_{BLS12}, and by a factor 16 for the p_{ST} prime field. The SINSEMILLA hash algorithm, which also leverages lookup tables for a faster plain evaluation, is thereby slower than RC by a factor of up to 125, while the traditional Pedersen Hash is only slower by a factor of 49. Compared to fast binary hash function, RC is only slower by a factor of 5 than Blake2, the fastest benchmarked hashing algorithm. Blake2 in turn however requires 7 times more Plookup gates than RC.

To further highlight the requirement for fast plain performance of ZK-friendly hash functions, we compare the runtime to accumulate a Merkle tree with 2^{20} elements in Section 9.8.2. One can observer, that using traditional hash function, accumulating the Merkle tree already requires $3\,s$, the runtime is significantly worse when using ZK-friendly hash functions, such as POSEIDON and RESCUE. RC with its fast plain performance, however, is only insignificantly slower then traditional hash functions, making it the optimal choice for use case which require fast plain performance, as well as fast ZK-proof generation.

Acknowledgements. We express our special thanks to Ferdinand Sauer for valuable discussion on Gröbner basis computations and for his assistance in conducting the practical experiments. We thank Alex Vlasov (Matter Labs) for his modular math optimizations and comments on the earlier version of the paper.

Lorenzo Grassi is supported by the European Research Council under the ERC advanced grant agreement under grant ERC-2017-ADG Nr. 788980 ESCADA. Roman Walch is supported by the "DDAI" COMET Module within the COMET – Competence Centers for Excellent Technologies Programme, funded by the Austrian Federal Ministry for Transport, Innovation and

Hashing algorithm		BN	BLS	ST
	s	s	s	s
RC	-	3.91	3.97	1.36
POSEIDON	-	22.6	23.8	22.3
RESCUE	-	497.2	520.6	396.8
RESCUE-Prime	-	436.3	458.4	324.3
Feistel-MiMC	-	42.2	44.3	34.1
GRIFFIN	-	122.7	129.6	95.0
NEPTUNE	-	24.4	26.1	24.1
SINSEMILLA	144.9	-	-	-
Pedersen Hash	60.1	-	-	-
SHA-256	0.624	-	-	-
Blake2b	0.225	-	-	-
Blake2s	0.222	-	-	-
SHA3-256	0.439	-	-	-

Table 11: Performance comparison in seconds (s) of different hash functions over prime fields with primes $p_{\text{BN}_{254}}, p_{\text{BLS}_{381}}, p_{\text{ST}}$ for computing a Merkle tree with 2^{20} elements. Implemented in Rust.

Technology (bmvit), the Austrian Federal Ministry for Digital and Economic Affairs (bmdw), the Austrian Research Promotion Agency (FFG), the province of Styria (SFG) and partners from industry and academia. The COMET Programme is managed by FFG.

9.a Bijectivity of Bar

Lemma 12. *The function* Bar *permutes* $\mathbb{F}_p$.

Proof. We work with the standard representations of $\mathbb{F}_p$ and $\mathbb{Z}_{s_1}, \ldots, \mathbb{Z}_{s_n}$. For $\mathcal{R} := \mathbb{Z}_{s_1} \times \ldots \times \mathbb{Z}_{s_n}$ let

$$\mathcal{R}_{<p} := \{(z_1, \ldots, z_n) \in \mathcal{R} : \sum_{i=1}^{n} z_i b_i < p\}.$$

The idea of the proof reads as follows: we show that

1. Decomp is injective and $\texttt{Decomp}(\mathbb{F}_p) \subseteq \mathcal{R}_{<p}$;
2. $\texttt{SBox}(\mathcal{R}_{<p}) \subseteq \mathcal{R}_{<p}$ and deduce that SBox permutes $\mathcal{R}_{<p}$;
3. Comp is injective on $\mathcal{R}_{<p}$.

With these statements, it follows at once that the function $\texttt{Bar} : \mathbb{F}_p \to \mathbb{F}_p$ given by

$$\texttt{Bar} = \texttt{Comp} \circ \texttt{SBox} \circ \texttt{Decomp}$$

is injective and hence surjective as well. In particular, we see that Decomp and Comp are indeed inverse functions over $\mathcal{R}_{<p}$ and $\mathbb{F}_p$.

Ad (1), (3): the statement $\texttt{Decomp}(\mathbb{F}_p) \subseteq \mathcal{R}_{<p}$ is a direct consequence of the definition of Decomp. For the injectivity of Decomp we show that it has a

left inverse function on $\mathcal{R}_{<p}$ which is precisely given by Comp restricted to $\mathcal{R}_{<p}$. Indeed, for $x \in \mathbb{F}_p$ it holds

$$\begin{aligned}(\texttt{Comp} \circ \texttt{Decomp})(x) &= \texttt{Comp}(x_1, \ldots, x_n) \\ &= \sum_{i=1}^{n} x_i b_i \bmod p = \sum_{i=1}^{n} x_i b_i = x.\end{aligned}$$

The second equality is just the definition of Comp, the third equality uses the fact that $\texttt{Decomp}(\mathbb{F}_p) \subseteq \mathcal{R}_{<p}$, and the fourth equality is true by definition of Decomp. Similarly, we obtain for $(z_1, \ldots, z_n) \in \mathcal{R}_{<p}$

$$\begin{aligned}&(\texttt{Decomp} \circ \texttt{Comp})(z_1, \ldots, z_n) \\ &= \texttt{Decomp}(\sum_{i=1}^{n} z_i b_i \bmod p) \\ &= \texttt{Decomp}(\sum_{i=1}^{n} z_i b_i) = (z_1, \ldots, z_n)\end{aligned}$$

and hence that Comp restricted to $\mathcal{R}_{<p}$ has the left inverse Decomp.

Ad (2): Since SBox is the parallel application of n smaller bijections it is clearly injective. The only assertion to prove is hence the inclusion $\texttt{SBox}(\mathcal{R}_{<p}) \subseteq \mathcal{R}_{<p}$. Let $(x_1, x_2, \ldots, x_n) \in \mathcal{R}_{<p}$ and let

$$(y_1, \ldots, y_n) = (S(x_1), \ldots, S(x_n))$$

denote the image under SBox. Now recall that $v = \min_i v_i$ where

$$(v_1, v_2, \ldots, v_n) = \texttt{Decomp}(p-1),$$

and let m be the smallest index such that $x_m < v$. If there is no such m, then all S-boxes S are identity functions and the assertion holds. If such an m exists, then for all $i < m$ we have $y_i = S(x_i) = x_i$ by the definition of the S_i. Moreover, for $i = m$ we have $y_m = S(x_m) < v \leq v_m$. For the remaining part we highlight the following property of our decomposition (which has an analogous counterpart in ordinary base-b expansion): for every $1 \leq k \leq n$ it holds

$$\begin{aligned}&\sum_{i=k+1}^{n} (s_i - 1) b_i = \sum_{i=k+1}^{n} (s_i - 1) \prod_{l>i} s_l \\ &= \sum_{i=k+1}^{n} \left(\prod_{l>i-1} s_l - \prod_{l>i} s_l \right) = \prod_{l>k} s_l - 1 = b_k - 1.\end{aligned}$$

Informally speaking, this translates to the statement "the sum of the maximal values of the first $l = n - k$ *least* significant positions equals the value of the next greater significant position minus 1". We use this fact and deduce

$$\begin{aligned}&\sum_{i=1}^{n} y_i b_i = \sum_{i=1}^{m-1} y_i b_i + y_m b_m + \underbrace{\sum_{i=m+1}^{n} y_i b_i}_{<b_m} \\ &< \sum_{i=1}^{m-1} x_i b_i + (y_m + 1) b_m \leq \sum_{i=1}^{m-1} x_i b_i + v_m b_m \\ &\leq \sum_{i=1}^{m-1} v_i b_i + v_m b_m \leq p - 1.\end{aligned}$$

Hence, $\texttt{SBox}(x_1, \ldots, x_n) \in \mathcal{R}_{<p}$ which implies that SBox permutes $\mathcal{R}_{<p}$. The second last inequality uses the property that for $u \in \mathbb{F}_p$ with $u \leq p-1$, the decompositions $(u_1, \ldots, u_n)$ and $(v_1, \ldots, v_n) = \texttt{Decomp}(p-1) \in \mathcal{R}$ satisfy for any $1 \leq k \leq n$ the inequality

$$\sum_{i=1}^{k} u_i b_i \leq \sum_{i=1}^{k} v_i b_i.$$

In other words, "if u is smaller than or equal to v, the sum of the values of any first k *most* significant digits of u is smaller than or equal to the corresponding sum for v." For $u = v$, the statement is obvious. For $u \neq v$, there is at least one index $1 \leq i \leq n$ with $u_i < v_i$; let t denote the minimal index with this property. If $k < t$, then $\sum_{i=1}^{k} u_i b_i = \sum_{i=1}^{k} v_i b_i$ by definition of t. If $k \geq t$ then

$$\begin{aligned}
\sum_{i=1}^{k} u_i b_i &= \sum_{i=1}^{t-1} u_i b_i + u_t b_t + \sum_{i=t+1}^{k} u_i b_i \\
&< \sum_{i=1}^{t-1} u_i b_i + (u_t + 1) b_t \leq \sum_{i=1}^{t-1} v_i b_i + v_t b_t \\
&\leq \sum_{i=1}^{k} v_i b_i. \qquad \square
\end{aligned}$$

10

A SIGNATURE-BASED GRÖBNER BASIS ALGORITHM WITH TAIL-REDUCED REDUCTORS (M5GB)

Based on the publication

[HLL+22] Manuel Hauke, Lukas Lamster, Reinhard Lüftenegger, and Christian Rechberger. "A Signature-Based Gröbner Basis Algorithm with Tail-Reduced Reductors (M5GB)." in: *IACR Cryptology ePrint Archive* (2022), p. 987. URL: https://eprint.iacr.org/2022/987

Abstract Gröbner bases are an important tool in computational algebra and, especially in cryptography, often serve as a boilerplate for solving systems of polynomial equations. Research regarding (efficient) algorithms for computing Gröbner bases spans a large body of dedicated work that stretches over the last six decades. The pioneering work of Bruno Buchberger in 1965 can be considered as the blueprint for all subsequent Gröbner basis algorithms to date. Among the most efficient algorithms in this line of work are signature-based Gröbner basis algorithms, with the first of its kind published in the late 1990s by Jean-Charles Faugère under the name `F5`. In addition to signature-based approaches, Rusydi Makarim and Marc Stevens investigated a different direction to efficiently compute Gröbner bases, which they published in 2017 with their algorithm `M4GB`. The ideas behind `M4GB` and signature-based approaches are conceptually orthogonal to each other because each approach addresses a different source of inefficiency in Buchberger's initial algorithm by different means.

We amalgamate those orthogonal ideas and devise a new Gröbner basis algorithm, called `M5GB`, that combines the concepts of both worlds. In that capacity, `M5GB` merges strong signature-criteria to eliminate redundant S-pairs with concepts for fast polynomial reductions borrowed from `M4GB`. We provide proofs of termination and correctness and a proof-of-concept implementation in `C++` by means of the Mathic library. The comparison with a state-of-the-art signature-based Gröbner basis algorithm (implemented via the same library) validates our expectations of an overall faster runtime for quadratic overdefined polynomial systems that have been used in comparisons before in the literature and are also part of cryptanalytic challenges.

Keywords Gröbner basis, Signature-based, M4GB, Tail-reduction

10.1 Introduction

Gröbner bases are an essential tool in commutative algebra and algebraic geometry. Several important applications in these areas are *(a)* the *Ideal Equality Problem*, characterizing the equality of two ideals through the reduced Gröbner bases of their sets of generators, *(b)* the *Ideal Membership Problem*, characterizing whether a polynomial belongs to a given ideal via the division remainder modulo the respective (reduced) Gröbner basis, and *(c)* the *Elimination Problem*, eliminating variables from a system of polynomial equations through, e.g., lexicographic Gröbner bases. There are many

more applications of Gröbner bases in signal and image processing, robotics, automated geometric theorem proving, and solving systems of polynomial equations [BW98], [Abl10].

Especially in cryptography, both public-key and symmetric cryptography, the problem of solving systems of polynomial equations arises in different contexts, ranging from block cipher and hash function analysis [ACG+19; BPW06] to the analysis of asymmetric encryption and signature schemes [TPD21; FJ03]. The applicability of Gröbner bases to cryptanalysis has been one of the driving factors for research on efficient algorithms for computing them. In general, the pioneering `Buchberger` algorithm for computing Gröbner bases devised by Bruno Buchberger and published in 1965 [Buc65; Buc76] can be considered highly inefficient, mainly due to the excessive amount of redundant computations that do not provide any new information for the eventual Gröbner basis. In more detail, the `Buchberger` algorithm repeatedly reduces so-called *S-pairs* [CLO15, p.85], adds all non-zero remainders to the current basis and repeats this process until all S-pairs reduce to zero with respect to the current basis. During the process of repeatedly reducing S-pairs, often many of those S-pairs reduce to zero and thus they do not provide any new information. To tackle this inefficiency, further criteria have been developed to streamline the `Buchberger` algorithm by detecting and discarding S-pairs that would otherwise reduce to zero. The work by Gebauer and Möller [GM88] implements these criteria and presents a more efficient instantiation of the `Buchberger` algorithm.

A different approach, which was initially investigated by Buchberger [Buc83a; Buc83b] and Lazard [Laz79; Laz83; Laz01] and further developed by Faugère in 1999 [Fau99], relates the problem of reducing S-pairs to the problem of reducing matrices. The basic underlying idea is that for a degree bound large enough and all term multiples of the initial polynomials up to this degree, the matrix containing the corresponding coefficients of the term multiples yields, after Gaussian row reduction, a Gröbner basis of the ideal generated by the initial polynomials. Rather than choosing a degree bound large enough and constructing one large matrix, Faugère's `F4` algorithm in [Fau99] constructs matrices for smaller degrees, row-reduces the corresponding smaller matrices and continues in this fashion until a Gröbner basis is found. Compared to the `Buchberger` algorithm, the advantage of `F4` is that S-pairs are reduced in parallel rather than sequentially. This advantage is the main source of the particular efficiency of `F4` and some of the fastest Gröbner basis implementations to date rely on this approach, as, e.g., the implementation in the computer algebra system Magma.

Signature-based Gröbner basis algorithms are another line of work regarding more efficient instantiations of the `Buchberger` algorithm. In the `F5` algorithm introduced by Faugère in 2002 [Fau02], so-called *signatures* help to keep track from which initial polynomials some S-pair has been calculated. The information from the signatures allows to detect whether a S-pair reduces to zero without having to carry out the reduction. Thus, the main idea of signature-based criteria is reducing the amount of redundant reductions. For a particular class of polynomial systems, called regular sequences, `F5` does not carry out any redundant reduction at all. The `F5` algorithm and other signature-based Gröbner basis algorithms have later been incorporated into the *rewrite* framework published by Christian Eder and Bjarke Roune [ER13]. The rewrite framework generalizes many different (signature-based) approaches for computing Gröbner bases and unifies them under the umbrella of a single comprehensive framework.

Compared to the approaches discussed above, Rusydi Makarim and Marc Stevens present an orthogonal concept for computing Gröbner bases [MS17]: their M4GB algorithm is based on the Gebauer-Möller version of the Buchberger algorithm, with the difference that reductions of S-pairs are carried out with tail-reduced reductors. In addition, these tail-reduced reductors are stored for potential later reuse. The main advantage of this approach is that no new reducible terms are introduced into the reduction remainder, which allows reducing S-pairs in a term-wise and recursive manner. This results in a fast polynomial reduction routine. The downside, however, is that any reductor has to be tail-reduced first, and the overall advantage of this approach depends on how often the algorithm is able to reuse already constructed and stored tail-reduced reductors.

10.1.1 Our Contribution

We present a new algorithm for computing Gröbner bases, called M5GB, which combines the strengths of M4GB [MS17] and signature-based Gröbner basis algorithms like F5 [Fau02]. We provide proofs of termination and correctness for M5GB. In particular, we show how one can adapt the fast reduction routine used in M4GB to work with the signature-based criteria from F5-like algorithms. This creates a generic optimization that can be used for any signature-based Gröbner basis algorithm that does not use a matrix approach for polynomial reduction. The question of merging the fast reduction routine from M4GB with signature-based criteria arises naturally but resolving it is a non-trivial task that requires technical care, especially when it comes to algorithmic efficiency. To date, and to the best of our knowledge, no algorithm in this line of work has been published yet.

For a proof-of-concept implementation, we concentrate on the signature-based algorithm SB from Stillman and Roune [RS12], also called SigGB in the reference implementation [Rou13b], and adapt this algorithm to be compatible with an M4GB-like reduction routine. We show that using the same library for implementing SB and M5GB, we obtain a significant, scalable speed-up for dense, quadratic, overdefined systems. These systems are used for benchmark purposes in the original article about M4GB by Makarim and Stevens [MS17] and are posed as a problem instance in the MQ Challenge [Tak15].

10.1.2 Related Work

Compared to our approach for computing Gröbner bases in M5GB, there exist related but different approaches in the literature. Here, we briefly discuss the main conceptual differences. One difference applies to all discussed algorithms below: we state our M5GB algorithm in the rewrite framework [ER13], while the below algorithms adopt the basic structure of F5. The rewrite framework comprises the original F5 algorithm as a special instantiation.

F4/5 Albrecht and Perry [AP10] describe an algorithm that combines F4-style reduction with F5-like signature criteria. This means, [AP10] integrates - like M5GB - a fast reduction routine with signature-based criteria to discard S-pairs. The difference to M5GB is that their algorithm F4/5 uses the same linear algebra approach for reducing S-pairs as F4, and thus conceptionally resembles Matrix-F5 [BFS15] rather than M5GB. Hence, it is the reduction of S-pairs that distinguishes F4/5 and M5GB: the former algorithm uses F4-style

reduction, while the latter one uses M4GB-style reduction. For a more detailed differentiation between the respective reduction routines in F4 and M4GB we refer to [MS17, Chapter 4.2].

F5C and F5R Eder and Perry [EP10] present a variant of Faugère's original F5 algorithm which works with reduced intermediate Gröbner bases rather than non-reduced ones. This results in fewer S-pairs to consider for checking the signature criteria and, eventually, fewer polynomial reductions. Eder and Perry differentiate their approach called F5C, "F5 Computing by reduced Gröbner bases", from the approach devised by Stegers [Sho09] for which they use the denomination F5R, "F5 Reducing by reduced Gröbner bases". In [Sho09], Stegers' F5R algorithm uses reduced intermediate bases only for polynomial reductions, however, it still uses unreduced intermediate bases for computing new S-pairs. In contrast, F5C uses reduced intermediate bases for, both, polynomial reductions and generating new S-pairs. To summarize, the advantage of F5R over F5 is faster polynomial reductions, while the advantage of F5C over F5R is a lower number of S-pairs to compute. The main conceptual difference between F5C and M5GB is, again, the reduction routine: F5C uses ordinary polynomial reduction while M5GB uses M4GB-style reduction.

10.2 Preliminaries

Any work treating the theory behind Gröbner bases and, in particular, describing different algorithms to compute Gröbner bases is faced with the challenge of having to introduce a significant amount of definitions and denominations. On top of that, there are often considerable notational differences between different authors. This being said, in Section 10.2.1 we pay attention to stay close to commonly shared denominations and to find a balance between a rigorous and compact nomenclature.

Furthermore, in Section 10.2.2 and Section 10.2.3 we give brief accounts of M4GB and signature-based algorithms for computing Gröbner bases, respectively. We describe these algorithms only to the extent that we are able to sketch their core ideas needed for our presentation of M5GB in Section 10.3. We assume some familiarity with these algorithms from the reader, although the core ideas should become apparent without any deeper prior knowledge.

10.2.1 Preliminary Definitions

We work with polynomials in the variables $X_1, \ldots, X_n$ over a finite field $\mathbb{F}$, i.e., with the polynomial ring $P := \mathbb{F}[X_1, \ldots, X_n]$. A term is a power product of variables, while a monomial is a product of coefficient and term. By T we denote the set of all terms in P. For a polynomial $f \in P$, the set $T(f)$ shall denote the set of all terms of f. For a term $t \in T(f)$, the corresponding coefficient is denoted as $C_t(f)$. We define the free P-module P^m with generators $\boldsymbol{e_1} := (1, 0, \ldots, 0), \ldots, \boldsymbol{e_m} := (0, \ldots, 0, 1)$. As for polynomials, a *module term* is an element in P^m of the form $t\boldsymbol{e_i}$, while a *module monomial* is an element of the form $c \cdot t\boldsymbol{e_i}$, for $c \in \mathbb{F}$, $t \in T$ and $1 \leq i \leq m$. The set of all module terms in P^m is denoted by T_m.

Throughout this article, we write module elements $\boldsymbol{f}, \boldsymbol{g}, \ldots$ in P^m in boldface, whereas polynomials $f, g, \ldots$ in P are written in normal style. We

denote a term order on T and a compatible order extension[1] to module terms in T_m by the same sign $\leq$. We believe, this ambiguity is justified by an easier notation and causes no deeper confusion because the context clarifies whether $\leq$ relates polynomials or module elements. For a given term order $\leq$, the *leading term* of a polynomial $f \in P$, denoted by $LT(f)$, is defined as the $\leq$-maximum term in $T(f)$ and the *leading coefficient* as the associated coefficient of $\mathrm{LT}(f)$. In a similar fashion, the *module leading term* $\mathrm{MLT}(\boldsymbol{f})$ and *module leading monomial* $\mathrm{MLM}(\boldsymbol{f})$ are defined for a module element $\boldsymbol{f} \in P^m$ and a compatible order extension $\leq$. The polynomial $\mathrm{Tail}(f) \coloneqq f - \mathrm{LC}(f) \cdot \mathrm{LT}(f)$ is called the *tail of* f.

Given a finite set of non-zero polynomials $F \coloneqq \{f_1, \ldots, f_m\} \subseteq P \setminus \{0\}$, the module homomorphism $\varphi_F : P^m \to P$ given by $(p_1, \ldots, p_m) \mapsto \sum_i p_i f_i$ connects the module and polynomial perspective. Usually, the underlying set F is clear, therefore we often omit the subscript and just write φ instead of φ_F. Using the canonical generators of P^m, we can also write $\varphi : \sum_i p_i \boldsymbol{e_i} \mapsto \sum_i p_i f_i$. Any module element $\boldsymbol{h} \in P^m$ with $\varphi(\boldsymbol{h}) = 0$ is called a *syzygy*. The *signature* of a module element $\boldsymbol{f} \in P^m$ is given by $\mathrm{sig}(\boldsymbol{f}) \coloneqq \mathrm{MLT}(\boldsymbol{f}) \in T_m$; of course, always relative to some compatible order extension $\leq$.

For a finite set of polynomials $G \subseteq P \setminus \{0\}$, a non-zero polynomial f is said to be *reducible with respect to* G, if there exist a term $t \in T(f)$ and an element $g \in G$ such that $\mathrm{LT}(g) \mid t$. If $u \coloneqq t/\mathrm{LT}(g)$ and $c \coloneqq C_t(f)$, we denote the reduction itself by $f \longrightarrow_G f - c \cdot ug$.

The element $c \cdot ug$ is called a *reductor of* f. If $t = \mathrm{LT}(f)$, the reduction step is also called a *top*-reduction, otherwise a *tail-reduction* and the corresponding reductors are called *top-reductor* and *tail-reductor*, respectively. If a polynomial is not reducible (or tail-reducible) with respect to G, it is called *irreducible* (or *tail-irreducible*) *with respect to* G. For the sake of notational convenience, any non-zero scalar multiple $d \cdot ug$, $d \in \mathbb{F} \setminus \{0\}$, is also called a reductor of f. This is why we often drop the scalar coefficient and just call ug a reductor of f. If f reduces to $h \in P$ in finitely many reduction steps with respect to G, we denote this by $f \longrightarrow_{G,*} h$. This also includes the case in which no reduction steps are done at all, hence $f \longrightarrow_G f$ is trivially valid. We call a polynomial $f' \in P$ to be a *normal form of* f *with respect to* G if $f \longrightarrow_{G,*} f'$ and f' is irreducible with respect to G. We use the denomination

$$f \bmod G \coloneqq \{f' \in P : f' \text{ a normal form of } f \text{ w.r.t. } G\}$$

to write down the set of all normal forms of a polynomial f.

Remark 1. Usually, we omit the specification *with respect to* G and presume it to be clear from the context; whenever necessary, we explicitly mention the underlying set G. The same applies for *Sig*-reductions defined below. Furthermore, we often do not mention nor incorporate the underlying (module) term order in our definitions and terminology. Again, the aim is having a lighter notation.

For a finite set of module elements $\boldsymbol{G} \subseteq P^m \setminus \{\boldsymbol{0}\}$, a non-zero module element $\boldsymbol{f} \in P^m \setminus \{\boldsymbol{0}\}$ is said to be *Sig-reducible with respect to* $\boldsymbol{G}$ if there exist a term $t \in T(\varphi(\boldsymbol{f}))$ and an element $\boldsymbol{g} \in \boldsymbol{G}$ such that the following two properties hold: *(i)* $\mathrm{LT}(\varphi(\boldsymbol{g})) \mid t$, in which case we set $u \coloneqq t/LT(\varphi(\boldsymbol{g}))$; *(ii)* $\mathrm{sig}(\boldsymbol{f}) \geq \mathrm{sig}(u\boldsymbol{g})$. If these properties are fulfilled, we define $\boldsymbol{f} - c \cdot u\boldsymbol{g}$ as the outcome of the *Sig*-reduction, where $c \coloneqq \mathrm{C}_t(\varphi(\boldsymbol{f}))/\mathrm{LC}(\varphi(\boldsymbol{g}))$, and denote the *Sig*-reduction itself by $\boldsymbol{f} \longrightarrow_{\boldsymbol{G}} \boldsymbol{f} - c \cdot u\boldsymbol{g}$. In particular, the element

[1] An order extension is called compatible, if $\forall u, v \in T\ \forall 1 \leq i \leq m : u \leq v \Rightarrow u\boldsymbol{e_i} \leq v\boldsymbol{e_i}$.

$c \cdot u\boldsymbol{g}$ is called a *Sig-reductor of* $\boldsymbol{f}$. If $\operatorname{sig}(\boldsymbol{f}) > \operatorname{sig}(u\boldsymbol{g})$, we call it a *regular Sig-reduction*, otherwise a *singular Sig-reduction*. We denote a regular *Sig*-reduction by $\boldsymbol{f} \longrightarrow_{\boldsymbol{G},reg} \boldsymbol{h}$ and, analogously, any finite number of regular *Sig*-reductions on $\boldsymbol{f}$ to a module element $\boldsymbol{h}$ by $\boldsymbol{f} \longrightarrow_{\boldsymbol{G},reg,*} \boldsymbol{h}$. We say $\boldsymbol{f} \longrightarrow_{\boldsymbol{G},*} 0$ if $\boldsymbol{f} \longrightarrow_{\boldsymbol{G},*} \boldsymbol{h}$ for a syzygy $\boldsymbol{h}$. This notation is justified by $\varphi(\boldsymbol{h}) = 0$. We believe, the definitions of *Sig-top-reduction*, *Sig-tail-reduction*, *Sig-irreducible*, *Sig-tail-irreducible*, *regularly Sig-irreducible*, *regularly Sig-tail-irreducible*, $\boldsymbol{f} \longrightarrow_{\boldsymbol{G},*} \boldsymbol{h}$ are clear without any further explication. In some cases it is convenient to speak of ordinarily reducing a module element $\boldsymbol{f} \in P^m$ (i.e., without above constraint *(ii)* regarding the signatures), when, in fact, we mean reducing the corresponding polynomial $\varphi(\boldsymbol{f}) \in P$.

We say $\boldsymbol{f'} \in P^m$ is a (regular) *Sig*-normal form of $\boldsymbol{f}$ if $\boldsymbol{f} \longrightarrow_{\boldsymbol{G},reg,*} \boldsymbol{f'}$ and $\boldsymbol{f'}$ is (regularly) *Sig*-irreducible. We denote by $\boldsymbol{f} \bmod \boldsymbol{G}$ the set of all *Sig*-normal forms of $\boldsymbol{f}$ with respect to $\boldsymbol{G}$, and by $\boldsymbol{f} \bmod_{\text{reg}} \boldsymbol{G}$ the set of all regular *Sig*-normal forms with respect to $\boldsymbol{G}$. For a pair of module elements $\boldsymbol{f}, \boldsymbol{g} \in P^m$ we define the *S*-pair of $\boldsymbol{f}$ and $\boldsymbol{g}$ as

$$\operatorname{Spair}(\boldsymbol{f}, \boldsymbol{g}) := \left(\frac{\ell}{LM(\varphi(\boldsymbol{f}))} \boldsymbol{f}, \frac{\ell}{LM(\varphi(\boldsymbol{g}))} \boldsymbol{g} \right) := (u\boldsymbol{f}, v\boldsymbol{g})$$

where $l := \operatorname{lcm}(LT(\varphi(\boldsymbol{f})), LT(\varphi(\boldsymbol{g})))$. We call $\operatorname{Spair}(\boldsymbol{f}, \boldsymbol{g})$ *regular* if $\operatorname{sig}(u\boldsymbol{f}) \neq \operatorname{sig}(v\boldsymbol{g})$ and *singular* otherwise.

Let $F := \{f_1, ..., f_m\} \subseteq P \setminus \{0\}$ be a set of polynomials, $I := \langle F \rangle$ the ideal generated by F and $\boldsymbol{s} \in T_m$ a module term. A set of module elements $\boldsymbol{G} \subseteq P^m \setminus \{\boldsymbol{0}\}$ is defined to be a *Sig-Gröbner basis of I up to signature* $\boldsymbol{s}$ if

$$\forall \boldsymbol{f} \in P^m : \operatorname{sig}(\boldsymbol{f}) < \boldsymbol{s} \Longrightarrow \boldsymbol{f} \longrightarrow_{\boldsymbol{G},*} 0.$$

The set $\boldsymbol{G}$ is called a *Sig-Gröbner basis of I* if $\boldsymbol{G}$ is a *Sig*-Gröbner basis up to every $\boldsymbol{s} \in P^m$ (i.e., for all possible signatures $\boldsymbol{s}$). The dependence on the set F (and thus the ideal I) is implicitly contained in the condition $\boldsymbol{f} \longrightarrow_{\boldsymbol{G},*} 0$, since for $\varphi = \varphi_F$ this implies $\varphi(\boldsymbol{f}) \longrightarrow_{\varphi(\boldsymbol{G}),*} 0$.[2]

A total order $\preceq$ on $\boldsymbol{G}$ with $\operatorname{sig}(\boldsymbol{f}) \mid \operatorname{sig}(\boldsymbol{g}) \Longrightarrow \boldsymbol{f} \preceq \boldsymbol{g}$, for all $\boldsymbol{f}, \boldsymbol{g} \in \boldsymbol{G}$, is called a *rewrite order*. We assume that all elements in $\boldsymbol{G}$ have distinct signatures, hence, the notion of a rewrite order is well-defined. For $\boldsymbol{s} \in T_m$, $\boldsymbol{f} \in P^m$ and $u \in T$, the element $u\boldsymbol{f} \in P^m$ is called *the canonical rewriter of signature* $\boldsymbol{s}$ *with respect to* $\boldsymbol{G}$ if $\boldsymbol{G} = \emptyset$ or if $\operatorname{sig}(u\boldsymbol{f}) = \boldsymbol{s}$ and $\boldsymbol{f} = \max_{\preceq}\{\boldsymbol{g} \in \boldsymbol{G} : \operatorname{sig}(\boldsymbol{g}) \mid \boldsymbol{s}\}$. Instead of this bulky denomination, we often just say "the canonical rewriter of $\boldsymbol{s}$", because the set $\boldsymbol{G}$ will be clear from the context.

10.2.2 M4GB Algorithm

In 2017 Rusydi Makarim and Marc Stevens published a new algorithm for computing Gröbner bases called M4GB. The main innovation of M4GB is a fast polynomial reduction routine that only uses tail-reduced reductors in each reduction step. In addition, M4GB maintains a set of already used (tail-reduced) reductors and thus allows to reuse reductors. We describe a variant of M4GB which is sketched in the performance section of [MS17, Sec. 4.1]. This variant outputs the same result as the original M4GB algorithm, albeit it is considered more performant due to time savings in the update process of the set of reductors. The authors of M4GB call this variant a *lazy* variant, whereas we simply refer to this variant as M4GB. Here, we only describe the

[2] The notion of *Sig*-Gröbner bases is motivated by the fact that if $\boldsymbol{G}$ is a *Sig*-Gröbner basis, then $\varphi(\boldsymbol{G})$ is a Gröbner basis.

core ideas and those parts of M4GB that are relevant for our new Gröbner basis algorithm M5GB in Section 10.3. In particular, we focus on the reduction of polynomials in M4GB. For a more detailed description of M4GB we refer the reader to the original article [MS17].

The M4GB algorithm essentially follows the basic outline of the textbook Buchberger algorithm [Buc76], which is "Select, Reduce, Update": selecting an S-pair, reducing it, and adding the reduced S-pair to the current basis in case it is nonzero. Whenever a nonzero reduced S-pair is added to the current basis, the set of S-pairs is updated. In M4GB, updating the set of S-pairs is achieved via the Gebauer-Möller criteria [GM88]. This process is repeated until all S-pairs have been processed. In addition to the basic "Select, Reduce, Update" triad, M4GB is characterised by the following two distinct properties: (a) it performs reductions only with tail-reduced reductors and, (b) it maintains a list of already used (tail-reduced) reductors for future use. The benefit of these two properties are faster reductions because (b) allows to reuse an already constructed (tail-reduced) reductor instead of re-constructing it again, while (a) ensures that during a reduction no new reducible terms are introduced into the resulting polynomial.

More formally, let G denote the current basis and $T_G(f)$ the set of reducible terms of $f \in P$ with respect to G. Assume M4GB reduces a term t in a polynomial p by an appropriate reductor m and m is *not* tail-irreducible with respect to G. Then, for further reducing the result of the reduction $p - m$, all terms in

$$T_G(p - m) = (T_G(p) \cup T_G(m)) \setminus \{t\}$$

would have to be reduced modulo G. However, if m is tail-irreducible we have by definition $T_G(m) \subseteq \{\mathrm{LT}(m)\} = \{t\}$, hence

$$T_G(p - m) = T_G(p) \setminus \{t\},$$

and only terms in $T(p) \setminus \{t\}$ need to be reduced modulo G. This is the main conceptual advantage of M4GB and its fast reduction routine.

Throughout all computations, M4GB maintains a set of reductors $M \supseteq G$, i.e., a set of monomial multiples of the current basis elements. All elements in M have unique leading terms, which is why the current basis G can be referenced only by its leading terms L. Nevertheless, we refer to L as the intermediate (or current) basis. The original formulation of M4GB in [MS17] proactively updates the whole set M in advance whenever a new basis element is generated. In contrast, the variant of M4GB that we describe (and that the authors of [MS17] implement) updates the elements in M only on-demand.[3] This means, only when an element $m \in M$ is reused, the algorithm checks if it needs to be tail-reduced with respect to the elements referenced by L. This leads to a *lazy* implementation of the update process of M. Although not explicitly stated in [MS17], for this lazy variant of M4GB the authors implicitly use the concept of *generations*: the generation of a reductor $m \in M$ is the cardinality $|L|$ of the intermediate basis L when m was added to M. Keeping track of the generation has the following purpose: whenever a reductor $m \in M$ is reused during the execution of M4GB and the generation of m is equal to the current generation, then we know m is tail-irreducible with respect to the current basis L and it can be used reused without any further considerations. If the generation of m is strictly smaller than the current generation, m needs to updated.

[3] When we speak of updating the set of reductors M, this is conceptionally different from updating the set of S-pairs. The former one is specific to M4GB, while the latter one is an essential feature of all Gröbner basis algorithms.

10.2.3 Signature-Based Algorithms

In the textbook version of the `Buchberger` algorithm, many of the S-pairs will be reduced to zero, which means they do not contribute any new information to the eventual Gröbner basis. Hence, a reduction to zero is redundant work, and it would be nice to have an oracle detecting whether or not an S-pair will be reduced to zero *without* having to carry out the actual reduction. There are criteria known to improve the textbook Buchberger algorithm in this regard (i.e., Buchberger's Product and Chain Criterion, realized in the Gebauer-Möller instantiation [GM88] of the `Buchberger` algorithm), but still many redundant reductions to zero might occur. In the following, a change of perspective helps to establish even stronger criteria for detecting redundant reductions to zero. Let f be a polynomial in the ideal generated by the polynomials $f_1, \ldots, f_m \in P$, i.e., $f \in \langle f_1, ..., f_m \rangle$. Then f can be written as $f = \sum_{i=1}^{m} p_i f_i$, for some polynomials $p_1, \ldots, p_m \in P$ (which are not necessarily unique). This notation of f motivates a new perspective: f cannot only be considered as polynomial but also as module element $(p_1, \ldots, p_m) \in P^m$. Adopting the module's perspective, it is possible to introduce a new concept called *signatures* for detecting unnecessary S-pair reductions. The main idea behind signatures is, roughly speaking, to keep track of how the polynomials generated during a Gröbner basis computation depend on the original input polynomials. More concretely, this means a signature-based algorithm not only processes information coming from a polynomial f itself but also from the vector $(p_1, \ldots, p_m)$ constituting the relation $f = \sum_i p_i f_i$, where the f_i would be the original input polynomials. On the one hand, this idea aims at exploiting zero-relations between the input polynomials (i.e., syzygies from the module perspective) to detect redundant reductions; on the other hand, it uses the (more subtle) fact that different polynomial combinations of the input polynomials (i.e., different module elements from the module perspective) can have the same reduction remainder. Thus only one of these reductions need to be performed. The former observation is the basis for the so-called *syzygy criterion*, while the latter observation leads to the *rewrite-criterion* (see Line 8 and 4, respectively, in Algorithm 15).

With above motivation of signatures at hand, we state the signature equivalent of Buchberger's S-pair criterion. The fundamental theorem underlying all signature-based algorithms is the following result.

Theorem 18 ([ER13], Theorem 3)**.** *Let $\boldsymbol{s} \in T_m$ be a module term and $\boldsymbol{G} \subseteq P^m$ be a finite set of module elements. If for all $\boldsymbol{p} \in P^m$ with $\boldsymbol{p}$ a regular S-pair of elements in $\boldsymbol{G}$ or $\boldsymbol{p}$ a canonical basis vector $\boldsymbol{e_i}$ (and $\operatorname{sig}(\boldsymbol{p}) < \boldsymbol{s}$, resp.) it holds that $\boldsymbol{p} \bmod_{\text{reg}} \boldsymbol{G}$ contains a syzygy or a singularly Sig-top-reducible element, then $\boldsymbol{G}$ is a signature Gröbner basis (up to $\boldsymbol{s}$, resp.).*

The following two observations explicate how signatures help to detect unnecessary reductions to zero in advance: assume we have a Gröbner basis $\boldsymbol{G} \subseteq P^m$ up to signature $\boldsymbol{s} \in T_m$. First, one can show that for any two regularly *Sig*-irreducible module elements $\boldsymbol{f}, \boldsymbol{g} \in P^m$ it holds

$$\operatorname{sig}(\boldsymbol{f}) = \operatorname{sig}(\boldsymbol{g}) = \boldsymbol{s} \Longrightarrow \varphi(\boldsymbol{f}) = c \cdot \varphi(\boldsymbol{g})$$

for some $c \in \mathbb{F} \setminus \{0\}$. Second, if there exists a syzygy $\boldsymbol{h} \in P^m$ with $\operatorname{sig}(\boldsymbol{h}) \mid \boldsymbol{s}$, then

$$\forall \boldsymbol{f} \in P^m \text{ with } \operatorname{sig}(\boldsymbol{f}) = \boldsymbol{s} : \boldsymbol{f} \longrightarrow_{\boldsymbol{G},*} 0.$$

The salient points are: *(a)* we only need to *Sig*-reduce one element with a given signature (we will choose the one which is 'easier' to handle). Hence, in a signature-based algorithm, instead of an S-pair with a given signature, we are free to choose any module element with the same signature and reduce this element to check whether the current signature provides new information for our eventual Gröbner basis. This approach is called *rewriting* and Gröbner basis algorithms based on this approach are called *rewrite* algorithms [ER13]; *(b)* if we know that the signature of the element to be reduced is a multiple of the signature of a syzygy, we can skip the computation of the reduction at all. This is why a signature-based algorithm always keeps track of syzygy signatures and stores them separately.

This is all we intend to say about the ideas behind signature-based and rewrite Gröbner basis algorithms and, in particular, we do not state a pseudo code for them. The basic ideas we adopt from the signature and rewriting approach for our M5GB algorithm are evident from Algorithm 15. For a more in-depth motivation and treatment of signature-based and rewrite Gröbner basis algorithms we refer to the comprehensive survey article [EF17].

10.3 M5GB Algorithm

In this section, we present our new Gröbner basis algorithm M5GB that amalgamates the core ideas of (signature-based) rewrite algorithms with the main ideas of M4GB. For this amalgamation to be viable, we introduce a new concept called *signature flags*. On a high level, signature flags play a similar role as generations in M4GB and allow to *efficiently* fuse the ideas behind signature-based algorithms and M4GB, respectively. As such, M5GB is an algorithm which aims to combine the strengths of both worlds: *(a)* fast reduction of polynomials due to the M4GB-like reduction routine; *(b)* strong criteria for discarding redundant S-pairs adopted from signature-based algorithms.

10.3.1 New Definitions

Since M5GB works with *Sig*-tail-irreducible reductors up to some signature $\boldsymbol{s}$, we explicate this concept in a formal definition. In the following let $\boldsymbol{G} \subseteq P^m \setminus \{\mathbf{0}\}$ be a non-empty and finite set of non-zero module elements.

We call a term $t \in T$ *Sig-reducible with respect to* $\boldsymbol{G}$ *and up to* $\boldsymbol{s}$, if there exist $u \in T$, $\boldsymbol{g} \in \boldsymbol{G}$ such that $\mathrm{LT}(\varphi(u\boldsymbol{g})) = t$ and $\mathrm{sig}(u\boldsymbol{g}) < \boldsymbol{s}$. A module element $\boldsymbol{f} \in P^m$ is called *Sig-reducible with respect to* $\boldsymbol{G}$ *and up to* $\boldsymbol{s}$, if there exists a term $t \in T(\varphi(\boldsymbol{f}))$ that is *Sig*-reducible with respect to $\boldsymbol{G}$ and up to $\boldsymbol{s}$. We denote such a reduction step by $\boldsymbol{f} \longrightarrow_{\boldsymbol{G},\boldsymbol{s}} \boldsymbol{f} - c \cdot u\boldsymbol{g}$, for an appropriate scalar $c \in \mathbb{F} \setminus \{0\}$. For a given set of terms $D \subseteq T(\varphi(\boldsymbol{f}))$, we call $\boldsymbol{f}$ *Sig-reducible with respect to* $\boldsymbol{G}$, D *and up to* $\boldsymbol{s}$ if there exists a reductor $u\boldsymbol{g}$ of $\boldsymbol{f}$ such that $\mathrm{LT}(\varphi(u\boldsymbol{g})) = d$ for some $d \in D$ and $\mathrm{sig}(u\boldsymbol{g}) < \boldsymbol{s}$. Such a reduction step is denoted by $\boldsymbol{f} \longrightarrow_{\boldsymbol{G},\boldsymbol{s},D} \boldsymbol{f} - c \cdot u\boldsymbol{g}$.

Remark 2. In particular, for $\boldsymbol{s} = \mathrm{sig}(\boldsymbol{f})$ and $D = T(\varphi(\boldsymbol{f}))$, the reduction $\boldsymbol{f} \longrightarrow_{\boldsymbol{G},\boldsymbol{s},D} \boldsymbol{f} - c \cdot u\boldsymbol{g}$ describes a regular *Sig*-reduction $\boldsymbol{f} \longrightarrow_{\boldsymbol{G},reg} \boldsymbol{f} - c \cdot u\boldsymbol{g}$. This means, our new view $\longrightarrow_{\boldsymbol{G},\boldsymbol{s},D}$ on signature-based reductions contains regular *Sig*-reductions as special case. Moreover, if we choose $D = T(\mathrm{Tail}(\varphi(\boldsymbol{f})))$, we allow all regular *Sig*-reductions except for a top-reduction. These two special cases are the instantations of D we are most

interested in, although the statements below, e.g., Lemma 13, can be applied for arbitrary $D \subseteq T(\varphi(\boldsymbol{f}))$.

As highlighted in Section 10.2.1, we often do not explicitly mention the set $\boldsymbol{G}$. In the same manner, we define $\boldsymbol{f} \longrightarrow_{\boldsymbol{G},\boldsymbol{s},*} \boldsymbol{h}$, $\boldsymbol{f} \longrightarrow_{\boldsymbol{G},\boldsymbol{s},D,*} \boldsymbol{h}$, *Sig-irreducible up to* $\boldsymbol{s}$, *Sig-irreducible with respect to* D *and up to* $\boldsymbol{s}$, *Sig-tail-irreducible up to* $\boldsymbol{s}$. A *normal form of* $\boldsymbol{f}$ *with respect to* $\boldsymbol{G}$ *and up to* $\boldsymbol{s}$ is an element $\boldsymbol{f'} \in P^m$ that is *Sig*-irreducible with respect to $\boldsymbol{G}$ and up to $\boldsymbol{s}$ and for which it holds $\boldsymbol{f} \longrightarrow_{\boldsymbol{G},\boldsymbol{s},*} \boldsymbol{f'}$. We denote the set of all normal forms of $\boldsymbol{f}$ with respect to $\boldsymbol{G}$ and up to $\boldsymbol{s}$ by $\boldsymbol{f} \bmod_{\boldsymbol{s}} \boldsymbol{G}$. For a set of terms $D \subseteq T(\varphi(\boldsymbol{f}))$, a *normal form of* $\boldsymbol{f}$ *with respect to* $\boldsymbol{G}$, D *and up to* $\boldsymbol{s}$ is an element $\boldsymbol{f'} \in P^m$ that is *Sig*-irreducible with respect to $\boldsymbol{G}$, D and up to $\boldsymbol{s}$ such that $\boldsymbol{f} \longrightarrow_{\boldsymbol{G},\boldsymbol{s},D,*} \boldsymbol{f'}$. We denote the set of all normal forms of $\boldsymbol{f}$ with respect to $\boldsymbol{G}$ and up to $\boldsymbol{s}$ by $\boldsymbol{f} \bmod_{\boldsymbol{s},D} \boldsymbol{G}$.

As in M4GB, the *generation* $\operatorname{gen}(\boldsymbol{m})$ of a reductor $\boldsymbol{m} \in \boldsymbol{M}$ is defined as the cardinality of the set $\boldsymbol{G}$ at the time $\boldsymbol{m}$ is constructed. [4] We denote the instance of $\boldsymbol{G}$ at this time with $\boldsymbol{G}_{\operatorname{gen}(\boldsymbol{m})}$. For a module element $\boldsymbol{f} \in P^m$ the *signature flag with respect to* $\boldsymbol{G}$ is defined as

$$\operatorname{Flag}(\boldsymbol{f}) \coloneqq \min\{\operatorname{sig}(v\boldsymbol{g}) : \boldsymbol{g} \in \boldsymbol{G}, v \in T, v\boldsymbol{g} \text{ a tail-reductor of } \boldsymbol{f}\},$$

or $\operatorname{Flag}(\boldsymbol{f}) \coloneqq \boldsymbol{\infty}$ if $\boldsymbol{f}$ is tail-irreducible. The symbol $\boldsymbol{\infty}$ can be understood as a formal symbol added to T_m with the simple property that

$$\forall \boldsymbol{s} \in T_m : \boldsymbol{s} < \boldsymbol{\infty}.$$

10.3.2 Description of M5GB

The overall structure of M5GB is depicted in Algorithm 15 and resembles the basic structure of a rewrite Gröbner basis algorithm (as outlined in [ER13]) with signature-based criteria to discard redundant S-pairs (see Line 4, 5, 6) and the fundamental "Select, Reduce, Update" triad from the Buchberger algorithm [Buc76]. In particular, M5GB processes S-pairs in strictly increasing signature and keeps track of syzygy signatures in a separate set $\boldsymbol{H}$ (Line 8). If a new basis element is found (Line 10), the Update routine (Algorithm 16) for the current basis $\boldsymbol{G}$ and the current set of S-pairs $\boldsymbol{P}$ is triggered. The steps in Update are governed by the same principles as in any other signature-based algorithm, with the difference, that Update detects whether a basis element $\boldsymbol{e_i}$ has been processed and thus extends the set of syzygy signatures $\boldsymbol{H}$ accordingly (Algorithm 16, Line 3). The main innovations of M5GB are incorporated into the reduction routine Reduce described in Algorithm 17. In the following, we discuss the novel features as well as the intricacies of Reduce more comprehensively.

[4] Here, the term "constructed" also encompasses the case when $\boldsymbol{m}$ is updated, or in other words, "re-constructed".

Algorithm 15: M5GB

Input: Non-zero input polynomials $F = \{f_1, ..., f_m\}$, a rewrite order, a term order on T and a compatible order extension on T_m
Output: A Gröbner basis G of the ideal generated by F

$\boldsymbol{G} := \emptyset$; $\boldsymbol{M} := \emptyset$; $\boldsymbol{H} := \emptyset$
$\boldsymbol{P} := \{\boldsymbol{e_i} : i \in \{1, ..., m\}\}$
while $\boldsymbol{P} \neq \emptyset$
  Select $\boldsymbol{f} \in \boldsymbol{P}$ with minimal signature $\boldsymbol{s} = \text{sig}(\boldsymbol{f})$ and $\boldsymbol{ug}$ the canonical rewriter of $\boldsymbol{s}$ w.r.t $\boldsymbol{G}$.
  $\boldsymbol{P} := \boldsymbol{P} \setminus \{\boldsymbol{p} \in \boldsymbol{P} : \text{sig}(\boldsymbol{p}) = \boldsymbol{s}\}$
  if $\boldsymbol{s}$ *is not divisible by some* $\boldsymbol{h} \in \boldsymbol{H}$ **then**
    $(\boldsymbol{M}, \boldsymbol{f'}) := \texttt{Reduce}(\boldsymbol{f}, \varphi(\boldsymbol{f}), \boldsymbol{s}, \boldsymbol{M}, \boldsymbol{G})$
    if $\boldsymbol{f'} = 0$ **then**
      $\boldsymbol{H} := \boldsymbol{H} \cup \{\boldsymbol{s}\}$
    else
      $(\boldsymbol{G}, \boldsymbol{P}, \boldsymbol{H}) := \texttt{Update}(\boldsymbol{f'}, \boldsymbol{G}, \boldsymbol{P}, \boldsymbol{H})$
return $\varphi(\boldsymbol{G})$

Algorithm 16: Update

Input: Current basis $\boldsymbol{G}$, set of S-pairs $\boldsymbol{P}$ and set of syzygy signatures $\boldsymbol{H}$, new basis element $\boldsymbol{f}$
Output: Updated $\boldsymbol{G}$, $\boldsymbol{P}$ and $\boldsymbol{H}$

$\boldsymbol{P} := \boldsymbol{P} \cup \{\text{Spair}(\boldsymbol{f}, \boldsymbol{g}) : \boldsymbol{g} \in \boldsymbol{G}, \text{Spair}(\boldsymbol{f}, \boldsymbol{g}) \text{ regular}\}$
$\boldsymbol{G} := \boldsymbol{G} \cup \{\boldsymbol{f}\}$
if $\text{sig}(\boldsymbol{f}) = \boldsymbol{e_i}$ **then** // $\boldsymbol{f}$ comes from a basis element $\boldsymbol{e_i}$
  $\boldsymbol{H} := \boldsymbol{H} \cup \{\text{sig}(\varphi(\boldsymbol{g})\boldsymbol{e_i} - \varphi(\boldsymbol{e_i})\boldsymbol{g}) : \boldsymbol{g} \in \boldsymbol{G}\}$
return $(\boldsymbol{G}, \boldsymbol{P}, \boldsymbol{H})$

As in M4GB, the Reduce routine keeps track of previously used reductors and stores them in a set $\boldsymbol{M}$. The key feature of M4GB, namely, working with tail-reduced reductors, is implemented in Reduce as well. The difference to M4GB and a crucial point is that whenever a reductor $\boldsymbol{m} \in P^m$ is added to $\boldsymbol{M}$, it need not be fully tail-irreducible with respect to $\boldsymbol{G}$ but only *Sig*-tail-irreducible up to the current signature $\boldsymbol{s}$. This property is an important part of our efficient amalgamation of signature-based algorithms with M4GB: by Theorem 18, signature-based algorithms work with *regular Sig-reductions* and hence, only those terms in $T(\text{Tail}(\boldsymbol{m}))$ need be reduced that have a reducer with signature smaller than $\boldsymbol{s}$. We formalized this particular property in the definitions in Section 10.3.1.

Again, as in M4GB, elements in $\boldsymbol{M}$ are updated in a lazy manner, meaning only on-demand when they are reused and not proactively whenever a new basis element is added to $\boldsymbol{G}$.

Remark 3. In the context of M5GB, updating an element of $\boldsymbol{M}$ alludes to the process of restoring its *Sig*-tail-irreducibility with respect to the current basis $\boldsymbol{G}$ and up to the current signature $\boldsymbol{s}$.

Below, we consider the two scenarios when some element $\boldsymbol{m} \in \boldsymbol{M}$ stops being *Sig*-tail-irreducible and thus needed to be updated in case it was reused:

1. A new element is added to $\boldsymbol{G}$ which regularly *Sig*-tail-reduces $\boldsymbol{m}$.
2. An existing tail-reductor of $\boldsymbol{m}$ becomes a valid *Sig*-tail-reductor in light of the current signature $\boldsymbol{s}$.

Algorithm 17: `Reduce`

Input: $f \in P^m$, polynomial $p \in P$ with $T(p) \subseteq T(\varphi(f))$, signature s, current basis G, current set of reductors $M \subseteq P^m$
Output: Possibly extended set M, Sig-normal form $f' \in f \bmod_{s,T(p)} G$ with respect to $T(p)$ and up to s

1 $f' := f$
2 **for** $t \in T(p)$ **do**
3 **if** $\exists m \in M : LT(\varphi(m)) = t$ **then**
4 Select such m
5 $m' := m$
6 **if** $\text{gen}(m) < |G|$ **then**
7 $M := M \setminus \{m\}$
8 $(M, m') := \texttt{Reduce}(m, \text{Tail}(\varphi(m)), s, G \setminus G_{\text{gen}(m)}, M)$
9 **if** $Flag(m) < s$ **then**
10 $(M, m') := \texttt{Reduce}(m', \text{Tail}(\varphi(m')), s, G_{\text{gen}(m)}, M)$
11 $(M, m') := \texttt{UpdateM}(M, m', G)$
12 **else if** $Flag(m) < s$ **then**
13 $M := M \setminus \{m\}$
14 $(M, m') := \texttt{Reduce}(m', \text{Tail}(\varphi(m')), s, G, M)$
15 $(M, m') := \texttt{UpdateM}(M, m', G)$
16 $f' := f' - C_t(\varphi(f')) \cdot m'$
17 **else if** $\exists g \in G : LT(\varphi(g)) \mid t, \text{sig}(ug) < s, u := t/LT(\varphi(g))$ **then**
18 Select such g
19 $(M, m') := \texttt{Reduce}(ug, \text{Tail}(\varphi(ug)), s, G, M)$
20 $(M, m') := \texttt{UpdateM}(M, m', G)$
21 $f' := f' - C_t(\varphi(f')) \cdot m'$
22 **return** (M, f');

The aspect in 2 needs some clarification. Assume, at the time m was added to M it was regularly Sig-tail-irreducible up to some signature r but not ordinarily tail-irreducible (i.e., $\varphi(m)$ is not tail-irreducible with respect to $\varphi(G)$). This means, at the time m was added to M there was some basis element multiple ug which tail-reduced m but the reduction was not a valid regular Sig-tail-reduction up to r, because $r \leq \text{sig}(ug)$. If the current signature s fulfills $s > \text{sig}(ug)$, the reductor ug becomes a valid reductor for a regular Sig-tail-reduction.

To resolve *(1)*, we use the concept of 'generations' (adopted from `M4GB`). All reductors added to M are equipped with a generation (the cardinality of G at the time m is created). Everytime a new basis element is added to G, the generation increases and thus, any reductor in M being reused and having a strictly smaller generation than the current one needs to be updated (Algorithm 17, Line 6). To resolve *(2)*, we use the new concept of 'signature flags'. All reductors added to M are equipped with a signature flag. The idea of a signature flag is to define it as the minimal signature for which *(2)* occurs. Consequently, if the signature flag of a reductor being reused is smaller than the current signature, the reductor needs to be updated (Algorithm 17, Line 9 and 12).

Algorithm 18: `UpdateM`

Input: Current set of reductors M, reductor m' to be normalized and equipped with generation and signature flag, current basis G
Output: Updated set M and updated m'

1 $m' := LC(\varphi(m'))^{-1} \cdot m'$
2 $Flag(m') := \min\{Flag(t) : t \in T(\mathrm{Tail}(\varphi(m')))\}$
3 $\mathrm{gen}(m') := |G|$
4 $M := M \cup \{m'\}$
5 **return** (M, m')

10.3.3 Termination and Correctness

Before we prove termination and correctness, we want to shed more light on the particular update process of reductors in `Reduce`. For this, we come back to the two situations in Section 10.3.2 when a reductor $m \in M$ stops being Sig-tail-irreducible with respect to the current basis G and up to the current signature s. Here, we state them more formally and by means of our new definitions from Section 10.3.1. Case *(1)* in Section 10.3.2 corresponds to

$$\exists t \in T(\mathrm{Tail}(m)), g \in G \setminus G_{\mathrm{gen}(m)}, v \in T : \mathrm{sig}(vg) < s \wedge \mathrm{LT}(\varphi(vg)) = t,$$

whereas case *(2)* is characterised by

$$\exists t \in T(\mathrm{Tail}(m)), g \in G_{\mathrm{gen}(m)}, v \in T : \mathrm{sig}(vg) < s \wedge \mathrm{LT}(\varphi(vg)) = t.$$

This is the reason why `Reduce` only needs to Sig-reduce with respect to $G \setminus G_{\mathrm{gen}(m)}$ in Line 8 whenever an update due to an older generation is necessary and the same reasoning applies to Line 10 and $G_{\mathrm{gen}(m)}$.

The outline of `M5GB` follows the same outline as a rewrite basis algorithm, with only the reduction routine `Reduce` being different. Since `M5GB` always calls `Reduce` with the arguments $(f, \varphi(f), s, M, G)$ and it holds $s = \mathrm{sig}(f)$, we only need to prove correctness and termination of `Reduce` to argue correctness and termination for `M5GB`. We begin with an important lemma. In essence, Lemma 13 explains why `Reduce` correctly computes a Sig-normal form with respect to a given set of terms D and up to signature s. We emphasize that the usage of Sig-tail-irreducible reductors is crucial here, without it, the statement would be wrong.

Lemma 13. *Let $f \in P^m$, $s \in T_m \cup \{\infty\}$ and $G \subseteq P^m$. Let $T_{G,s,D}(f)$ denote the set of all terms in $D \subseteq T(\varphi(f))$ that are Sig-reducible with respect to G and up to signature s. For each $t \in T_{G,s,D}(f)$, let m_t denote a reductor of t with $\mathrm{sig}(m_t) < s$ which is Sig-tail-irreducible with respect to G and up to s. Then*

$$f' := f - \sum_{t \in T_{G,s,D}(f)} c_t(f) \cdot m_t \in f \bmod_{s,D} G$$

where the coefficients $c_t(f)$ are defined by $\varphi(f) = \sum_{t \in T(f)} c_t(f) t$.

Proof. Because all m_t are Sig-tail-irreducible up to s, we have

$$T_{G,s,D}(f') \subseteq \Big(T_{G,s,D}(f) \cup \bigcup_{t \in T_{G,s,D}(f)} T_{G,s,D}(m_t)\Big) \setminus T_{G,s,D}(f) \quad = \emptyset,$$

so it follows that f' is *Sig*-irreducible with respect to D and up to s. We are left to show that $f \longrightarrow_{G,s,D,*} f'$. To do so, we proceed inductively: assume by hypothesis that for a fixed $n \in \mathbb{N}$, we have that

$$f \longrightarrow_{G,s,D,*} f_n := f - \sum_{t \in S} c_t(f) \cdot m_t$$

holds for arbitrary $S \subseteq T_{G,s,D}(f)$ with $|S| = n$. If $S = T_{G,s,D}(f)$, then $f' = f_n$ and the claim holds trivially. Otherwise, let $t_0 \in T_{G,s,D}(f) \setminus S$. We need to show that $f_n \longrightarrow_{G,s,D,*} f_n - c_{t_0}(f) m_{t_0}$. As $\operatorname{sig}(f) = \operatorname{sig}(f_n)$, it suffices to show that $t_0 \in T_{G,s,D}(f_n)$. As m_t is Sig-tail-irreducible by assumption for every $t \in S$, we have $t_0 \notin \bigcup_{t \in S} T(m_t)$ and in particular, $t_0 \in T_{G,s,D}(f_n)$ follows. This concludes the proof. □

Theorem 19. `Reduce` *terminates and correctly computes a Sig-normal form* $f' \in f \bmod_{s,T(p)} G$.

Proof. For termination, we note that whenever `Reduce` is processing a term t in recursion level $n \in \mathbb{N}_0$ and calls itself, all terms being processed in the following recursion level $n+1$ regarding t are strictly smaller than t. This is because whenever `Reduce` calls itself in level n while processing a term t, it calls itself on $\operatorname{Tail}(v)$ of some polynomial v with $\operatorname{LT}(v) = t$ and thus for any subsequent term u in level $n+1$ regarding t it holds $u < t$. Hence, the recursion depth of `Reduce` must be finite. Since at a given recursion level only finitely many terms are being processed, we conclude that `Reduce` eventually terminates.

To argue correctness, in view of Lemma 13, it suffices to prove that for every $t \in T(p)$, a potential reductor m'_t is *Sig*-tail-irreducible up to s and fulfills $\operatorname{sig}(m'_t) < s$.

It is clear that `Reduce` reaches the end of a recursive path if and only if it processes a reductor where it does not call itself anymore. Looking at Algorithm 17, this is the case if and only if `Reduce` is being called with $(f, T(p), s, G, M)$ such that every $t \in T(p)$ is either *(i) Sig*-irreducible with respect to G and up to s or *(ii)* there already exists a reductor $m \in M$ with $\operatorname{gen}(m) = |G|$ and $\operatorname{Flag}(m) \geq s$. By the definitions of generation and signature flag and by the construction of elements in M, *(ii)* is equivalent with $\operatorname{sig}(m) < s$ and m being *Sig*-tail-irreducible with respect to G and up to s. Using Lemma 13, we deduce that the reduction remainder $f' \in f \bmod_{s,T(p)} G$. If f' serves as a reductor m'_t in a recursion level above, note that $\operatorname{sig}(f) < s$ and hence, also $\operatorname{sig}(f') < s$ follows. □

Corollary 6. *The algorithm M5GB terminates and is correct.*

Proof. The routines of Algorithm 15 and Algorithm 16 are essentially identical to the algorithm RB in [ER13], where a short proof of termination and correctness is provided. For a more detailed treatment we refer the interested reader to [Hau20]. Thus we are left to show that the reduction routine in `Reduce` terminates and outputs a *Sig*-normal form $f' \in f \bmod_{s,T(p)} G$, which is done in Theorem 19. □

10.4 Implementation & Performance

In this section, we discuss some implementation details and the performance of our `M5GB` algorithm. We base our implementation on the Mathic `C++`-library developed by Roune [Rou13a]. In Mathic, we integrate our algorithm

as a new module into the MathicGB Gröbner basis module. Using the Mathic framework allows us to directly compare the performance of M5GB against the signature-based algorithm SB presented by Roune and Stillman [RS12]. Keeping the same naming convention as in [RS12], we refer to their signature-based Gröbner basis algorithm as SB. As we optimize our implementation of the reduction routine outlined in Algorithm 17, there are minor differences to the pseudocode. These differences have no impact on the overall behaviour or correctness of the algorithm. Instead, they aim to leverage the language-specific advantages of C++ to create a competitive proof-of-concept implementation. The source code of our implementation of M5GB is available under https://extgit.iaik.tugraz.at/krypto/m5gb.git.

We show that using the same library for implementing SB and M5GB, we obtain a significant, scalable speed-up for dense, quadratic, overdefined polynomial systems. These systems are used for benchmark purposes in the original article about M4GB by Makarim and Stevens [MS17] and are posed as a problem instance in the MQ Challenge [Tak15].

We also performed informal tests for other systems, e.g., some canonical test systems in the literature like katsura, eco or cyclic. Most of the results indicated that the performance of M5GB falls behind that of SB. We conjecture several reasons behind these results. First, creating tail-reduced reductors is time-consuming and, depending on the structure of the polynomial system, may not yield an overall advantage compared to using ordinary reductors. Second, due to the recursive nature of the M4GB-style reductions, we cannot use the efficient data structures that Mathic uses to increase the performance of their implementation. Lastly, and connected to the previous point, since M5GB uses M4GB-style reduction, our algorithm also inherits the disadvantages of M4GB. This is further evidenced by the outcomes of informal comparisons between M4GB and M5GB. Although these two algorithms are implemented in a substantially different way, we found that whenever M5GB performed poorly this also was the case for M4GB. However, to provide a more reliable conclusion in this regard, further and more systematic experiments are needed. This, as well, includes implementing M4GB and M5GB in a more comparable manner. We leave this open for future work.

10.4.1 Implementation Details

The original signature-based algorithm SB of Roune and Stillman [RS12; Rou13b] does not use signature flags and generations. Thus, we extend the underlying data structures such that generations and signature flags are supported. Both generations and signature flags are implemented on term and polynomial granularity. Each polynomial stores its generation as an integer value. An unordered map $\boldsymbol{I}$, that maps term hashes to generations, stores the generations of irreducible terms. We do not need to store additional information for reducible terms, as they always cause a reducer lookup in the current basis or a lookup in the current set of tail-reduced reductors $\boldsymbol{M}$.

Contrary to the pseudocode, we do not explicitly calculate and store signature flags when terms and module elements are stored in $\boldsymbol{I}$ or $\boldsymbol{M}$. Instead, we only store the information on whether a term has a dividing leading term in the base. Only those terms may have a signature flag that is not ∞. We encode this information using a single bit in the generation integer. If a term or polynomial has a finite flag, we calculate and store the actual signature flag on the first subsequent access. This approach allows us to calculate signature flags only for elements where the flag is actually

needed by the algorithm. Thus, we avoid unnecessary flag computations and also reduce the memory overhead. For systems where most signature flags are infinite, this optimization allows us to skip most of the flag logic, which leads to a further increase in the performance of our implementation.

10.4.2 Performance Metrics

We evaluate our implementation of `M5GB` by computing the Gröbner basis for overdefined dense quadratic systems with an increasing number of variables N and $M = 2N$ polynomials over $\mathbb{F}_{101}$, unless stated otherwise. The variable count ranges from $N = 5$ to $N = 21$. For each N, we generate 10 distinct equation systems that are certain to have a solution. The performance metrics for each N are computed as the arithmetic mean of the metric over all 10 system instances. In our evaluation, we consider the following three metrics.

Time per Basis Element The time spent per basis element is the primary indicator of the performance of our algorithm. A lower amount of time per basis element indicates a faster implementation. The resulting time per basis element is computed as the overall runtime divided by the number of elements that reside in the final Gröbner basis.

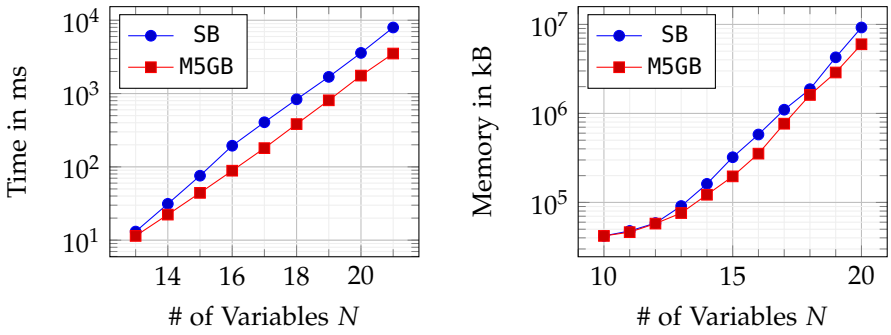
Peak Memory Usage We monitor the memory consumption of the implementations using the `time`-program on Linux. While we could track all memory allocations in the program through instrumentation-based monitoring, we chose to measure the overall memory footprint instead, as it can become a limiting factor when calculating large bases.

Number of Reductions As a third metric, we keep track of the number of actual reductions. A *reduction* (or *reduction step*) in this context is a single step in the process of reducing a polynomial (with respect to some set of divisors). We extend the existing `SB` implementation such that each reduction step is counted. Likewise, we keep track of reduction steps in `M5GB` as well. Then, for a fixed polynomial system, we compare the respective number of reductions in `SB` and `M5GB`.

10.4.3 Evaluation and Discussion

Figure 17a illustrates the arithmetic mean of the measured timing results. The obtained results show that our implementation outperforms `SB` for dense quadratic systems in all tested systems. For both implementations, the time per basis element approximately doubles with each variable. Nevertheless, the runtime of `M5GB` consistently stays below the runtime of `SB` in any of the tested systems. Our evaluation shows that the runtime ration between `SB` and `M5GB` fluctuates over different values for N. Figure 19a depicts the ratio between the arithmetic means of runtimes depending on the variable count N. After a slight drop between $N = 17$ and $N = 20$, for larger systems with $N = 21$ the performance advantage of `M5GB` starts to increase again.

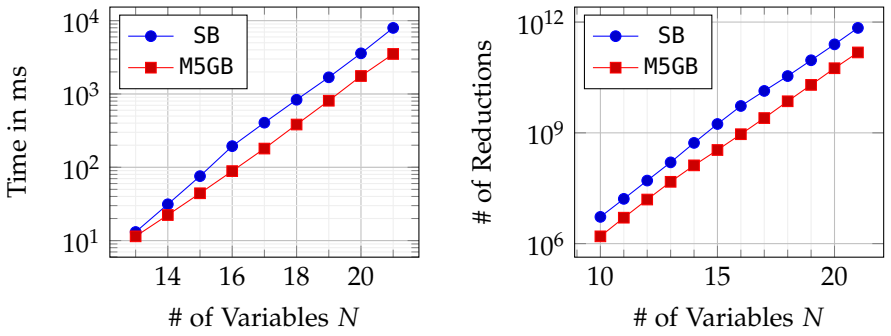
As we are particularly interested in dense quadratic systems, we also evaluate the performance depending on the number of polynomials M for a fixed number of variables N. We find that decreasing the number of polynomials negatively influences the runtime and the performance gain of `M5GB` compared to `SB`. For all evaluated systems, increasing the equation count reduces the runtime for both implementations. The runtime ratio is



(a) The mean time spent per basis element calculated over 10 dense quadratic systems per variable count.

(b) The mean memory consumption depending on the number of variables, averaged over 10 systems.

Figure 17: The mean runtime and peak memory consumption for overdefined quadratic polynomial systems with increasing variable count N and $M = 2N$ polynomials over $\mathbb{F}_{101}$.



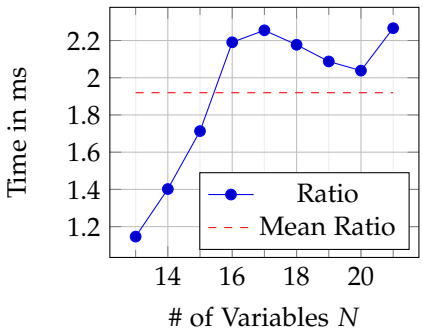
(a) The mean time spent per basis element calculated over 10 dense quadratic systems per variable count.

(b) The mean number of reductions steps for `SB` and `M5GB`.

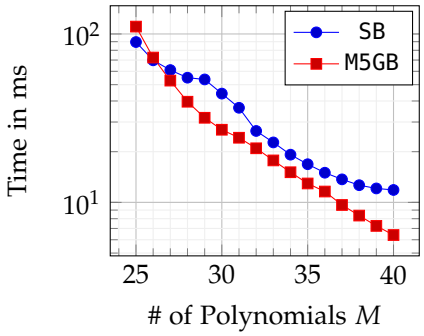
Figure 18: Comparison between mean runtime spent per basis element and the mean number of reduction steps. The average is the arithmetic mean over 10 runs for dense, overdefined, quadratic polynomial systems over $\mathbb{F}_{101}$.

not strongly influenced by increasing the number of polynomials M. Figure 19b illustrates the runtime changes depending on the number of provided polynomials M. Our baseline system has $N = 15$ variables, and we vary the equation count. Our evaluation demonstrates that once $M \geq 1.8 \cdot N$ holds, our implementation outperforms the classic implementation for the tested systems. From these results, we conjecture that `M5GB` will continue to outperform the classic algorithm for even larger systems, as long as they are sufficiently overdefined.

Figure 17b shows the memory consumption of both implementations on a logarithmic scale. As depicted, `M5GB` tends to use less memory than the `SB` implementation. We can see that memory consumption increases exponentially. This is unsurprising, given that the number of possible S-pairs also grows exponentially. We could further reduce the memory footprint by design choices in the implementation. This, however, would not improve the memory growth behaviour of the algorithm itself. For the peak memory consumption metric, we were not able to test systems with $N = 21$, as the memory profiling imposes an additional runtime overhead.



(a) Ratio of runtimes between SB and M5GB. The mean ratio is approximately 1.9.



(b) Runtime per basis element for an overdefined system over $\mathbb{F}_{101}$ with $N = 15$ variables and a varying number of M polynomials.

Figure 19: Ratio of runtimes between SB and M5GB for a varying variable count and runtimes of SB and M5GB for a fixed number of variables N and a varying number of polynomials M over $\mathbb{F}_{101}$.

In Figure 18b we see that the number of actually performed reductions is significantly lower in M5GB than in SB. A comparison of Figure 18a and Figure 18b indicates that the number of reductions is a good indicator of the time cost. Note that the ratio between the reduction counts is larger than the actual ratio of runtimes in Figure 19a. As our implementation is meant as a proof of concept, this observation leads to the assumption that a well-optimized implementation might lead to even higher performance gains.

10.5 Future Work

As future work we plan to implement our M5GB algorithm via the dedicated and optimized implementation of M4GB. It is the optimized implementation of M4GB that holds some of the top rankings in the MQ challenge [Tak15].

BIBLIOGRAPHY

[AAB+20] Abdelrahaman Aly, Tomer Ashur, Eli Ben-Sasson, Siemen Dhooghe, and Alan Szepieniec. "Design of Symmetric-Key Primitives for Advanced Cryptographic Protocols." In: *IACR Transactions on Symmetric Cryptology* 2020.3 (2020), pp. 1–45. DOI: 10.13154/tosc.v2020.i3.1-45.

[Abl10] Rafal Ablamowicz. "Some Applications of Gröbner Bases in Robotics and Engineering." In: *Geometric Algebra Computing - in Engineering and Computer Science*. Ed. by Eduardo Bayro-Corrochano and Gerik Scheuermann. Springer, 2010, pp. 495–517. DOI: 10.1007/978-1-84996-108-0_23.

[ABM23] Tomer Ashur, Thomas Buschman, and Mohammad Mahzoun. "Algebraic Cryptanalysis of HADES Design Strategy: Application to Poseidon and Poseidon2." In: *IACR Cryptology ePrint Archive* (2023), p. 537. URL: https://eprint.iacr.org/2023/537.

[AC09] Martin R. Albrecht and Carlos Cid. "Algebraic Techniques in Differential Cryptanalysis." In: *Fast Software Encryption - FSE 2009*. Ed. by Orr Dunkelman. Vol. 5665. Lecture Notes in Computer Science. Springer, 2009, pp. 193–208. DOI: 10.1007/978-3-642-03317-9_12.

[ACF+15] Martin R. Albrecht, Carlos Cid, Jean-Charles Faugère, Robert Fitzpatrick, and Ludovic Perret. "Algebraic Algorithms for LWE Problems." In: *ACM Communications in Computer Algebra* 49.2 (2015), p. 62. DOI: 10.1145/2815111.2815158.

[ACG+19] Martin R. Albrecht, Carlos Cid, Lorenzo Grassi, Dmitry Khovratovich, Reinhard Lüftenegger, Christian Rechberger, and Markus Schofnegger. "Algebraic Cryptanalysis of STARK-Friendly Designs: Application to MARVELlous and MiMC." In: *Advances in Cryptology - ASIACRYPT 2019*. Ed. by Steven D. Galbraith and Shiho Moriai. Vol. 11923. Lecture Notes in Computer Science. Springer, 2019, pp. 371–397. DOI: 10.1007/978-3-030-34618-8_13.

[AD18] Tomer Ashur and Siemen Dhooghe. "MARVELlous: A STARK-Friendly Family of Cryptographic Primitives." In: *IACR Cryptology ePrint Archive* (2018), p. 1098. URL: https://eprint.iacr.org/2018/1098.

[AD20] Tomer Ashur and Siemen Dhooghe. "Prelude to Marvellous (With the Designers' Commentary, Two Bonus Tracks, and a Foretold Prophecy)." In: *IACR Cryptology ePrint Archive* (2020), p. 568. URL: https://eprint.iacr.org/2020/568.

[AFI+04] Gwénolé Ars, Jean-Charles Faugère, Hideki Imai, Mitsuru Kawazoe, and Makoto Sugita. "Comparison Between XL and Gröbner Basis Algorithms." In: *Advances in Cryptology - ASIACRYPT 2004*. Ed. by Pil Joong Lee. Vol. 3329. Lecture Notes in Computer Science. Springer, 2004, pp. 338–353. DOI: 10.1007/978-3-540-30539-2_24.

[AG11] Sanjeev Arora and Rong Ge. "New Algorithms for Learning in Presence of Errors." In: *Internation Colloquium Automata, Languages and Programming - ICALP 2011*. Ed. by Luca Aceto, Monika Henzinger, and Jirí Sgall. Vol. 6755. Lecture Notes in Computer Science. Springer, 2011, pp. 403–415. DOI: 10.1007/978-3-642-22006-7_34.

[AGP+19] Martin R. Albrecht, Lorenzo Grassi, Léo Perrin, Sebastian Ramacher, Christian Rechberger, Dragos Rotaru, Arnab Roy, and Markus Schofnegger. "Feistel Structures for MPC, and More." In: *European Symposium on Research in Computer Security - ESORICS 2019*. Ed. by Kazue Sako, Steve A. Schneider, and Peter Y. A. Ryan. Vol. 11736. Lecture Notes in Computer Science. Springer, 2019, pp. 151–171. DOI: 10.1007/978-3-030-29962-0_8.

[AGR+16] Martin R. Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. "MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity." In: *Advances in Cryptology - ASIACRYPT 2016*. Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Vol. 10031. Lecture Notes in Computer Science. 2016, pp. 191–219. DOI: 10.1007/978-3-662-53887-6_7.

[AHU74] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

[AKM+22] Tomer Ashur, Al Kindi, Willi Meier, Alan Szepieniec, and Bobbin Threadbare. "Rescue-Prime Optimized." In: *IACR Cryptology ePrint Archive* (2022), p. 1577. URL: https://eprint.iacr.org/2022/1577.

[AKM23] Tomer Ashur, Al Kindi, and Mohammad Mahzoun. "XHash8 and XHash12: Efficient STARK-friendly Hash Functions." In: *IACR Cryptology ePrint Archive* (2023), p. 1045. URL: https://eprint.iacr.org/2023/1045.

[AM09] Jean-Philippe Aumasson and Willi Meier. "Zero-Sum Distinguishers for Reduced Keccak-f and for the Core Functions of Luffa and Hamsi." In: *Presented at the rump session of Cryptographic Hardware and Embedded Systems - CHES 2009* (2009).

[AP10] Martin Albrecht and John Perry. *F4/5*. 2010. arXiv: 1006.4933 [math.AC].

[ARS+15a] Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. "Ciphers for MPC and FHE." In: *Advances in Cryptology - EUROCRYPT 2015*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9056. Lecture Notes in Computer Science. Springer, 2015, pp. 430–454. DOI: 10.1007/978-3-662-46800-5_17.

[ARS+15b] Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. "Ciphers for MPC and FHE." In: *Advances in Cryptology - EUROCRYPT 2015*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9056. Lecture Notes in Computer Science. Springer, 2015, pp. 430–454. DOI: 10.1007/978-3-662-46800-5_17.

[Ash19] Tomer Ashur. *Private Communication*. 2019.

[Ava16] Roberto Avanzi. "A Salad of Block Ciphers." In: *IACR Cryptology ePrint Archive* (2016), p. 1171. URL: http://eprint.iacr.org/2016/1171.

[Bak19] Valentin Bakoev. "Distribution of the Boolean Functions of n Variables according to their Algebraic Degrees." In: *Serdica Journal of Computing* 13.1-2 (2019), pp. 17–26. DOI: 10.55630/sjc.2019.13.17-26.

[Bar04] Magali Bardet. "Étude des Systèmes Algébriques Surdéterminés. Applications aux Codes Correcteurs et à la Cryptographie." PhD thesis. Pierre and Marie Curie University, Paris, France, 2004. URL: https://tel.archives-ouvertes.fr/tel-00449609.

[BBB+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Gregory Maxwell. "Bulletproofs: Short Proofs for Confidential Transactions and More." In: *Symposium on Security and Privacy - SP 2018*. IEEE Computer Society, 2018, pp. 315–334. DOI: 10.1109/SP.2018.00020.

[BBC+23] Clémence Bouvier, Pierre Briaud, Pyrros Chaidos, Léo Perrin, Robin Salen, Vesselin Velichkov, and Danny Willems. "New Design Techniques for Efficient Arithmetization-Oriented Hash Functions: Anemoi Permutations and Jive Compression Mode." In: *Advances in Cryptology - CRYPTO 2023*. Ed. by Helena Handschuh and Anna Lysyanskaya. Vol. 14083. Lecture Notes in Computer Science. Springer, 2023, pp. 507–539. DOI: 10.1007/978-3-031-38548-3_17.

[BBH+18a] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. "Fast Reed-Solomon Interactive Oracle Proofs of Proximity." In: *International Colloquium on Automata, Languages, and Programming - ICALP 2018*. Ed. by Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella. Vol. 107. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018, 14:1–14:17. DOI: 10.4230/LIPIcs.ICALP.2018.14.

[BBH+18b] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. "Scalable, Transparent, and Post-Quantum Secure Computational Integrity." In: *IACR Cryptology ePrint Archive* (2018), p. 46. URL: https://eprint.iacr.org/2018/046.

[BBL+22] Augustin Bariant, Clémence Bouvier, Gaëtan Leurent, and Léo Perrin. "Algebraic Attacks against Some Arithmetization-Oriented Primitives." In: *IACR Transactions Symmetric Cryptology* 2022.3 (2022), pp. 73–101. DOI: 10.46586/tosc.v2022.i3.73-101.

[BBS99] Eli Biham, Alex Biryukov, and Adi Shamir. "Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials." In: *Advances in Cryptology - EUROCRYPT 1999*. Ed. by Jacques Stern. Vol. 1592. Lecture Notes in Computer Science. Springer, 1999, pp. 12–23. DOI: 10.1007/3-540-48910-X_2.

[BC13] Christina Boura and Anne Canteaut. "On the Influence of the Algebraic Degree of F^{-1} on the Algebraic Degree of $G \circ F$." In: *IEEE Transactions on Information Theory* 59.1 (2013), pp. 691–702. DOI: 10.1109/TIT.2012.2214203.

[BCB20] Morgan Barbier, Hayat Cheballah, and Jean-Marie Le Bars. "On the computation of the Möbius transform." In: *Theoretical Computer Science* 809 (2020), pp. 171–188. DOI: 10.1016/j.tcs.2019.12.005.

[BCC11] Christina Boura, Anne Canteaut, and Christophe De Cannière. "Higher-Order Differential Properties of Keccak and Luffa." In: *Fast Software Encryption - FSE 2011*. Ed. by Antoine Joux. Vol. 6733. Lecture Notes in Computer Science. Springer, 2011, pp. 252–269. DOI: 10.1007/978-3-642-21702-9_15.

[BCD+20a] Tim Beyne, Anne Canteaut, Itai Dinur, Maria Eichlseder, Gregor Leander, Gaëtan Leurent, María Naya-Plasencia, Léo Perrin, Yu Sasaki, Yosuke Todo, and Friedrich Wiemer. "Out of Oddity - New Cryptanalytic Techniques Against Symmetric Primitives Optimized for Integrity Proof Systems." In: *Advances in Cryptology - CRYPTO 2020*. Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12172. Lecture Notes in Computer Science. Springer, 2020, pp. 299–328. DOI: 10.1007/978-3-030-56877-1_11.

[BCD+20b] Tim Beyne, Anne Canteaut, Itai Dinur, Maria Eichlseder, Gaëtan Leurent, Gregor Leander, María Naya-Plasencia, Léo Perrin, Yu Sasaki, Yosuke Todo, and Friedrich Wiemer. *Report on the Security of STARK-Friendly Hash Functions (Version 2.0)*. 2020. URL: https://eips.ethereum.org/assets/eip-5988/papers/report_security_stark_friendly_hash.pdf.

[BCG+14a] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. "Zerocash: Decentralized Anonymous Payments from Bitcoin." In: *IACR Cryptology ePrint Archive* (2014), p. 349. URL: http://eprint.iacr.org/2014/349.

[BCG+14b] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. "Zerocash: Decentralized Anonymous Payments from Bitcoin." In: *Symposium on Security and Privacy - SP 2014*. IEEE Computer Society, 2014, pp. 459–474. DOI: 10.1109/SP.2014.36.

[BCG+18] Jonathan Bootle, Andrea Cerulli, Jens Groth, Sune K. Jakobsen, and Mary Maller. "Arya: Nearly Linear-Time Zero-Knowledge Proofs for Correct Program Execution." In: *Advances in Cryptology - ASIACRYPT 2018*. Ed. by Thomas Peyrin and Steven D. Galbraith. Vol. 11272. Lecture Notes in Computer Science. Springer, 2018, pp. 595–626. DOI: 10.1007/978-3-030-03326-2_20.

[BCI+13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. "Succinct Non-interactive Arguments via Linear Interactive Proofs." In: *Theory of Cryptography Conference - TCC 2013*. Ed. by Amit Sahai. Vol. 7785. Lecture Notes in Computer Science. Springer, 2013, pp. 315–333. DOI: 10.1007/978-3-642-36594-2_18.

[BCL+20] Tim Beyne, Anne Canteaut, Gregor Leander, María Naya Plasencia, Léo Perrin, and Friedrich Wiemer. "On the Security of the Rescue Hash Function." In: *IACR Cryptology ePrint Archive* (2020), p. 820. URL: https://eprint.iacr.org/2020/820.

[BCL+21] Benedikt Bünz, Alessandro Chiesa, William Lin, Pratyush Mishra, and Nicholas Spooner. "Proof-Carrying Data Without Succinct Arguments." In: *Advances in Cryptology - CRYPTO 2021*. Ed. by Tal Malkin and Chris Peikert. Vol. 12825. Lecture Notes in Computer Science. Springer, 2021, pp. 681–710. DOI: 10.1007/978-3-030-84242-0_24.

[BCM+20] Benedikt Bünz, Alessandro Chiesa, Pratyush Mishra, and Nicholas Spooner. "Recursive Proof Composition from Accumulation Schemes." In: *Conference on the Theory of Cryptography - TCC 2020*. Ed. by Rafael Pass and Krzysztof Pietrzak. Vol. 12551. Springer, 2020, pp. 1–18. DOI: 10.1007/978-3-030-64378-2_1.

[BCP23] Clémence Bouvier, Anne Canteaut, and Léo Perrin. "On the Algebraic Degree of Iterated Power Functions." In: *Designs, Codes, Cryptography* 91.3 (2023), pp. 997–1033. DOI: 10.1007/s10623-022-01136-x.

[BCR+19] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. "Aurora: Transparent Succinct Arguments for R1CS." In: *Advances in Cryptology - EUROCRYPT 2019*. Ed. by Yuval Ishai and Vincent Rijmen. Vol. 11476. Lecture Notes in Computer Science. Springer, 2019, pp. 103–128. DOI: 10.1007/978-3-030-17653-2_4.

[BDF+21] Dan Boneh, Justin Drake, Ben Fisch, and Ariel Gabizon. "Halo Infinite: Proof-Carrying Data from Additive Polynomial Commitments." In: *Advances in Cryptology - CRYPTO 2021*. Ed. by Tal Malkin and Chris Peikert. Vol. 12825. Lecture Notes in Computer Science. Springer, 2021, pp. 649–680. DOI: 10.1007/978-3-030-84242-0_23.

[BDK+21] Nicolas Bordes, Joan Daemen, Daniël Kuijsters, and Gilles Van Assche. "Thinking Outside the Superbox." In: *Advances in Cryptology - CRYPTO 2021*. Ed. by Tal Malkin and Chris Peikert. Vol. 12827. Lecture Notes in Computer Science. Springer, 2021, pp. 337–367. DOI: 10.1007/978-3-030-84252-9_12.

[BDP+08] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "On the Indifferentiability of the Sponge Construction." In: *Advances in Cryptology - EUROCRYPT 2008*. Ed. by Nigel P. Smart. Vol. 4965. Lecture Notes in Computer Science. Springer, 2008, pp. 181–197. DOI: 10.1007/978-3-540-78967-3_11.

[BDP+11] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. *The Keccak Reference*. Tech. rep. 2011. URL: https://keccak.team/files/Keccak-reference-3.0.pdf.

[BDP+13] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "Keccak." In: *Advances in Cryptology - EUROCRYPT 2013*. Ed. by Thomas Johansson and Phong Q. Nguyen. Vol. 7881. Lecture Notes in Computer Science. Springer, 2013, pp. 313–314. DOI: 10.1007/978-3-642-38348-9_19.

[Ber71] Elwyn R. Berlekamp. "Factoring Polynomials over Large Finite Fields." In: *Symposium on Symbolic and Algebraic Manipulation - SYMSAC 1971*. Ed. by Stanley R. Petrick, Jean E. Sammet, Robert G. Tobey, and Joel Moses. ACM, 1971, p. 223. DOI: 10.1145/800204.806290.

[BFP09] Luk Bettale, Jean-Charles Faugère, and Ludovic Perret. "Hybrid Approach for Solving Multivariate Systems over Finite Fields." In: *Journal of Mathematical Cryptology* 3.3 (2009), pp. 177–197. DOI: 10.1515/JMC.2009.009.

[BFP12] Luk Bettale, Jean-Charles Faugère, and Ludovic Perret. "Solving Polynomial Systems over Finite Fields: Improved Analysis of the Hybrid Approach." In: *International Symposium on Symbolic and Algebraic Computation - ISSAC 2012*. Ed. by Joris van der Hoeven and Mark van Hoeij. ACM, 2012, pp. 67–74. DOI: 10.1145/2442829.2442843.

[BFS04] Magali Bardet, Jean-Charles Faugere, and Bruno Salvy. "On the Complexity of Gröbner Basis Computation of Semi-Regular Overdetermined Algebraic Equations." In: *International Conference on Polynomial System Solving - ICPSS 2004*. 2004, pp. 71–74. URL: http://magali.bardet.free.fr/Publis/ltx43BF.pdf.

[BFS+05] Magali Bardet, Jean-Charles Faugère, Bruno Salvy, and B.-Y. Yang. "Asymptotic Behaviour of the Degree of Regularity of Semi-Regular Polynomial Systems." In: *Effective Methods in Algebraic Geometry - MEGA 2005*. 2005, pp. 1–16. URL: https://www-polsys.lip6.fr/~jcf/Papers/BFS05b.pdf.

[BFS15] Magali Bardet, Jean-Charles Faugère, and Bruno Salvy. "On the Complexity of the F5 Gröbner Basis Algorithm." In: *Journal of Symbolic Computation* 70 (2015), pp. 49–70.

[BGL20] Eli Ben-Sasson, Lior Goldberg, and David Levit. "STARK Friendly Hash - Survey and Recommendation." In: *IACR Cryptology ePrint Archive* (2020), p. 948. URL: https://eprint.iacr.org/2020/948.

[BH21] Sean Bowe and Daira Hopwood. *Zcash Orchard: Sinsemilla Gadget*. 2021. URL: https://zcash.github.io/orchard/design/circuit/gadgets/sinsemilla.html.

[BHH+19] Sean Bowe, Daira Hopwood, Taylor Hornby, and Nathan Wilcox. *Zcash Protocol Specification: Version 2019.0-beta-37*. 2019. URL: https://github.com/zcash/zips/blob/master/protocol/protocol.pdf.

[BHH+22] Sean Bowe, Daira Hopwood, Taylor Hornby, and Nathan Wilcox. *Zcash Protocol Specification: Version 2022.3.8*. Tech. rep. 2022. URL: https://github.com/zcash/zips/blob/main/protocol/protocol.pdf.

[BJK+16] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. "The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS." In: *Advances in Cryptology - CRYPTO 2016*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9815. Lecture Notes in Computer Science. Springer, 2016, pp. 123–153. DOI: 10.1007/978-3-662-53008-5_5.

[BKL+07] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. "PRESENT: An Ultra-Lightweight Block Cipher." In: *Cryptographic Hardware and Embedded Systems - CHES 2007*. Ed. by Pascal Paillier and Ingrid Verbauwhede. Vol. 4727.

Lecture Notes in Computer Science. Springer, 2007, pp. 450–466. DOI: 10.1007/978-3-540-74735-2_31.

[BKP16] Alex Biryukov, Dmitry Khovratovich, and Léo Perrin. "Multiset-Algebraic Cryptanalysis of Reduced Kuznyechik, Khazad, and secret SPNs." In: *IACR Transactions on Symmetric Cryptology* 2016.2 (2016), pp. 226–247. DOI: 10.13154/tosc.v2016.i2.226-247.

[BKR00] Mihir Bellare, Joe Kilian, and Phillip Rogaway. "The Security of the Cipher Block Chaining Message Authentication Code." In: *Journal of Computer and System Sciences* 61.3 (2000), pp. 362–399. DOI: 10.1006/jcss.1999.1694.

[BKR94] Mihir Bellare, Joe Kilian, and Phillip Rogaway. "The Security of Cipher Block Chaining." In: *Advances in Cryptology - CRYPTO 1994*. Ed. by Yvo Desmedt. Vol. 839. Lecture Notes in Computer Science. Springer, 1994, pp. 341–358. DOI: 10.1007/3-540-48658-5_32.

[BKR98] Mihir Bellare, Ted Krovetz, and Phillip Rogaway. "Luby-Rackoff Backwards: Increasing Security by Making Block Ciphers Non-invertible." In: *Advances in Cryptology - EUROCRYPT 1998*. Ed. by Kaisa Nyberg. Vol. 1403. Lecture Notes in Computer Science. Springer, 1998, pp. 266–280. DOI: 10.1007/BFb0054132.

[BND+20] M. Bigdeli, E. De Negri, M. M. Dizdarevic, Elisa Gorla, R. Minko, and S. Tsakou. "Semi-Regular Sequences and Other Random Systems of Equations." In: *CoRR* abs/2011.01032 (2020). arXiv: 2011.01032.

[Bon19] Xavier Bonnetain. "Collisions on Feistel-MiMC and univariate GMiMC." In: *IACR Cryptology ePrint Archive* (2019), p. 951. URL: https://eprint.iacr.org/2019/951.

[BPV98] Johan Borst, Bart Preneel, and Joos Vandewalle. "On the Time-Memory Tradeoff between Exhaustive Key Search and Table Precomputation." In: *Symposium on Information Theory in the Benelux*. Ed. by P.H. de With and M. van der Schaar-Mitrea. Werkgemeenschap voor Informatieen Communicatietheorie. 1998, pp. 111–118. URL: https://www.esat.kuleuven.be/cosic/publications/article-332.pdf.

[BPW06] Johannes Buchmann, Andrei Pyshkin, and Ralf-Philipp Weinmann. "A Zero-Dimensional Gröbner Basis for AES-128." In: *Fast Software Encryption - FSE 2006*. Ed. by Matthew J. B. Robshaw. Vol. 4047. Lecture Notes in Computer Science. Springer, 2006, pp. 78–88. DOI: 10.1007/11799313_6.

[BR14] Andrey Bogdanov and Vincent Rijmen. "Linear hulls with correlation zero and linear cryptanalysis of block ciphers." In: *Designs, Codes, Cryptography* 70.3 (2014), pp. 369–383. DOI: 10.1007/s10623-012-9697-z.

[BS18] Eli Ben-Sasson. *State of the STARK*. 2018. URL: https://drive.google.com/file/d/1Osa0MXu-04dfwn1YOSgN6CXOgWnsp-Tu/view.

[BS91] Eli Biham and Adi Shamir. "Differential Cryptanalysis of DES-like Cryptosystems." In: *Journal of Cryptology* 4.1 (1991), pp. 3–72. DOI: 10.1007/BF00630563.

[BS93] Eli Biham and Adi Shamir. *Differential Cryptanalysis of the Data Encryption Standard*. Springer, 1993. DOI: 10.1007/978-1-4613-9314-6.

[Buc06] Bruno Buchberger. "Bruno Buchberger's PhD Thesis 1965: An Algorithm for Finding the Basis Elements of the Residue Class Ring of a Zero Dimensional Polynomial Ideal." In: *Journal of Symbolic Computation* 41.3-4 (2006), pp. 475–511. DOI: 10.1016/j.jsc.2005.09.007.

[Buc18] Bruno Buchberger. "Gröbner Bases Computation by Triangularizing Macaulay Matrices." In: *The 50th Anniversary of Gröbner Bases*. Ed. by Takayuki Hibi. Vol. 77. Advanced Studies in Pure Mathematics. Mathematical Society of Japan, 2018, pp. 25–33.

[Buc65] Bruno Buchberger. "Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal." PhD thesis. University of Innsbruck, 1965. URL: https://www3.risc.jku.at/publications/download/risc_2769/1965-00-00-A.pdf.

[Buc76] Bruno Buchberger. "A Theoretical Basis for the Reduction of Polynomials to Canonical Forms." In: *SIGSAM Bulletin* 10.3 (1976), pp. 19–29. DOI: 10.1145/1088216.1088219.

[Buc83a] Bruno Buchberger. "Gröbner Bases, Gaussian Elimination and Euclidean Algorithm." In: *Invited Colloquium Talk at University of Grenoble, IMAG Institute*. 1983.

[Buc83b] Bruno Buchberger. *Miscellaneous Results on Gröbner Bases for Polynomial Ideals II. Technical Report 83/1*. Tech. rep. University of Delaware, Department of Computer and Information Sciences. 1983.

[BW93] Thomas Becker and Volker Weispfenning. *Gröbner bases. A Computational Approach to Commutative Algebra*. Vol. 141. Graduate Texts in Mathematics. Springer, 1993. DOI: 10.1007/978-1-4612-0913-3.

[BW98] Bruno Buchberger and Franz Winkler, eds. *Gröbner Bases and Applications*. Vol. 251. London Mathematical Society Lecture Note Series. Cambridge University Press, 1998.

[Car10] Claude Carlet. "Boolean Functions for Cryptography and Error-Correcting Codes." In: *Boolean Models and Methods in Mathematics, Computer Science, and Engineering*. Ed. by Yves Crama and Peter L. Hammer. Cambridge University Press, 2010, pp. 257–397. DOI: 10.1017/cbo9780511780448.011.

[Car20] Claude Carlet. "Graph Indicators of Vectorial Functions and Bounds on the Algebraic Degree of Composite Functions." In: *IEEE Transactions on Information Theory* 66.12 (2020), pp. 7702–7716. DOI: 10.1109/TIT.2020.3017494.

[CB07] Nicolas T. Courtois and Gregory V. Bard. "Algebraic Cryptanalysis of the Data Encryption Standard." In: *International Conference on Coding and Cryptography - ICCC 2007*. Ed. by Steven D. Galbraith. Vol. 4887. Lecture Notes in Computer Science. Springer, 2007, pp. 152–169. DOI: 10.1007/978-3-540-77272-9_10.

[CCH+19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. "Fiat-Shamir: From practice to Theory." In: *SIGACT Symposium on Theory of Computing - STOC 2019*. Ed. by Moses Charikar and Edith Cohen. ACM, 2019, pp. 1082–1090. DOI: 10.1145/3313276.3316380.

[CCZ98] Claude Carlet, Pascale Charpin, and Victor A. Zinoviev. "Codes, Bent Functions and Permutations Suitable For DES-like Cryptosystems." In: *Designs, Codes, Cryptography* 15.2 (1998), pp. 125–156. DOI: 10.1023/A:1008344232130.

[CG20] Alessio Caminata and Elisa Gorla. "Solving Multivariate Polynomial Systems and an Invariant from Commutative Algebra." In: *Arithmetic of Finite Fields - 8th International Workshop, WAIFI 2020, Rennes, France, July 6-8, 2020, Revised Selected and Invited Papers*. Ed. by Jean-Claude Bajard and Alev Topuzoglu. Vol. 12542. Lecture Notes in Computer Science. Springer, 2020, pp. 3–36. DOI: 10.1007/978-3-030-68869-1_1.

[CGG+20] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. "TFHE: Fast Fully Homomorphic Encryption Over the Torus." In: *Journal of Cryptology* 33.1 (2020), pp. 34–91. DOI: 10.1007/s00145-019-09319-x.

[CGG+22] Carlos Cid, Lorenzo Grassi, Aldo Gunsing, Reinhard Lüftenegger, Christian Rechberger, and Markus Schofnegger. "Influence of the Linear Layer on the Algebraic Degree in SP-Networks." In: *IACR Transactions on Symmetric Cryptology* 2022.1 (2022), pp. 110–137. DOI: 10.46586/tosc.v2022.i1.110-137.

[CHW+22] Jiamin Cui, Kai Hu, Meiqin Wang, and Puwen Wei. "On the Field-Based Division Property: Applications to MiMC, Feistel MiMC and GMiMC." In: *Advances in Cryptology - ASIACRYPT 2022*. Ed. by Shweta Agrawal and Dongdai Lin. Vol. 13793. Lecture Notes in Computer Science. Springer, 2022, pp. 241–270. DOI: 10.1007/978-3-031-22969-5_9.

[CL05] Carlos Cid and Gaëtan Leurent. "An Analysis of the XSL Algorithm." In: *Advances in Cryptology - ASIACRYPT 2005*. Ed. by Bimal K. Roy. Vol. 3788. Lecture Notes in Computer Science. Springer, 2005, pp. 333–352. DOI: 10.1007/11593447_18.

[CLO05] David A. Cox, John Little, and Donal O'Shea. *Using Algebraic Geometry*. 2nd ed. Vol. 185. Lecture Notes in Computer Science. Springer, 2005. DOI: 10.1007/b138611.

[CLO15] David Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms: an Introduction to Computational Algebraic Geometry and Commutative Algebra*. 4th ed. Springer, 2015. DOI: 10.1007/978-3-319-16721-3.

[CMR06] Carlos Cid, Sean Murphy, and Matthew J. B. Robshaw. *Algebraic Aspects of the Advanced Encryption Standard*. Springer, 2006. DOI: 10.1007/978-0-387-36842-9.

[COS20] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. "Fractal: Post-quantum and Transparent Recursive Proofs from Holography." In: *Advances in Cryptology - EUROCRYPT 2020*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12105. Lecture Notes in Computer Science. Springer, 2020, pp. 769–793. DOI: 10.1007/978-3-030-45721-1_27.

[Cou02] Nicolas T. Courtois. "Higher Order Correlation Attacks, XL Algorithm and Cryptanalysis of Toyocrypt." In: *International Conference on Information Security and Cryptology - ICISC 2002*. Ed. by Pil Joong Lee and Chae Hoon Lim. Vol. 2587. Lecture Notes in Computer Science. Springer, 2002, pp. 182–199. DOI: 10.1007/3-540-36552-4_13.

[Cou03] Nicolas Courtois. "Generic Attacks and the Security of Quartz." In: *Public Key Cryptography - PKC 2003*. Ed. by Yvo Desmedt. Vol. 2567. Lecture Notes in Computer Science. Springer, 2003, pp. 351–364. DOI: 10.1007/3-540-36288-6_26.

[CP02] Nicolas T. Courtois and Josef Pieprzyk. "Cryptanalysis of Block Ciphers with Overdefined Systems of Equations." In: *Advances in Cryptology - ASIACRYPT 2002*. Ed. by Yuliang Zheng. Vol. 2501. Lecture Notes in Computer Science. Springer, 2002, pp. 267–287. DOI: 10.1007/3-540-36178-2_17.

[CV02] Anne Canteaut and Marion Videau. "Degree of Composition of Highly Nonlinear Functions and Applications to Higher Order Differential Cryptanalysis." In: *Advances in Cryptology - EUROCRYPT 2002*. Ed. by Lars R. Knudsen. Vol. 2332. Lecture Notes in Computer Science. Springer, 2002, pp. 518–533. DOI: 10.1007/3-540-46035-7_34.

[CW09] Carlos Cid and Ralf-Philipp Weinmann. "Block Ciphers: Algebraic Cryptanalysis and Gröbner Bases." In: *Gröbner Bases, Coding, and Cryptography*. Ed. by Massimiliano Sala, Shojiro Sakata, Teo Mora, Carlo Traverso, and Ludovic Perret. Springer, 2009, pp. 307–327. DOI: 10.1007/978-3-540-93806-4_17.

[CXZ+21] Siwei Chen, Zejun Xiang, Xiangyong Zeng, and Shasha Zhang. "On the Relationships between Different Methods for Degree Evaluation." In: *IACR Transactions on Symmetric Cryptology* 2021.1 (2021), pp. 411–442. DOI: 10.46586/tosc.v2021.i1.411-442.

[CZ81] David Cantor and Hans Zassenhaus. "A New Algorithm for Factoring Polynomials over Finite Fields." In: *Mathematics of Computation* 36.154 (1981), pp. 587–592.

[DGG+21] Christoph Dobraunig, Lorenzo Grassi, Anna Guinet, and Daniël Kuijsters. "Ciminion: Symmetric Encryption Based on Toffoli-Gates over Large Finite Fields." In: *Advances in Cryptology - EUROCRYPT 2021*. Ed. by Anne Canteaut and François-Xavier Standaert. Vol. 12697. Lecture Notes in Computer Science. Springer, 2021, pp. 3–34. DOI: 10.1007/978-3-030-77886-6_1.

[DKR97] Joan Daemen, Lars R. Knudsen, and Vincent Rijmen. "The Block Cipher Square." In: *Fast Software Encryption - FSE 1997*. Ed. by Eli Biham. Vol. 1267. Lecture Notes in Computer Science. Springer, 1997, pp. 149–165. DOI: 10.1007/BFb0052343.

[DLM+15] Itai Dinur, Yunwen Liu, Willi Meier, and Qingju Wang. "Optimized Interpolation Attacks on LowMC." In: *Advances in Cryptology - ASIACRYPT 2015*. Ed. by Tetsu Iwata and Jung Hee Cheon. Vol. 9453. Lecture Notes in Computer Science. Springer, 2015, pp. 535–560. DOI: 10.1007/978-3-662-48800-3_22.

[DR01] Joan Daemen and Vincent Rijmen. "The Wide Trail Design Strategy." In: *IMA Cryptography and Coding - IMACC 2001*. Ed. by Bahram Honary. Vol. 2260. Lecture Notes in Computer Science. Springer, 2001, pp. 222–238. DOI: 10.1007/3-540-45325-3_20.

[DR02a] Joan Daemen and Vincent Rijmen. "Security of a Wide Trail Design." In: *Progress in Cryptology - INDOCRYPT 2002*. Ed. by Alfred Menezes and Palash Sarkar. Vol. 2551. Lecture Notes in Computer Science. Springer, 2002, pp. 1–11. DOI: 10.1007/3-540-36231-2_1.

[DR02b] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002. DOI: 10.1007/978-3-662-04722-4.

[DRS20] Christoph Dobraunig, Yann Rotella, and Jan Schoone. "Algebraic and Higher-Order Differential Cryptanalysis of Pyjamask-96." In: *IACR Transactions on Symmetric Cryptology* 2020.1 (2020), pp. 289–312. DOI: 10.13154/tosc.v2020.i1.289-312.

[DS09] Itai Dinur and Adi Shamir. "Cube Attacks on Tweakable Black Box Polynomials." In: *Advances in Cryptology - EUROCRYPT 2009*. Ed. by Antoine Joux. Vol. 5479. Lecture Notes in Computer Science. Springer, 2009, pp. 278–299. DOI: 10.1007/978-3-642-01001-9_16.

[Dwo15] Morris Dworkin. *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. 2015. DOI: https://doi.org/10.6028/NIST.FIPS.202.

[EF17] Christian Eder and Jean-Charles Faugère. "A Survey on Signature Based Algorithms for Computing Gröbner Bases." In: *Journal of Symbolic Computation* 80 (2017), pp. 719–784. DOI: 10.1016/j.jsc.2016.07.031.

[EGL+20] Maria Eichlseder, Lorenzo Grassi, Reinhard Lüftenegger, Morten Øygarden, Christian Rechberger, Markus Schofnegger, and Qingju Wang. "An Algebraic Attack on Ciphers with Low-Degree Round Functions: Application to Full MiMC." In: *Advances in Cryptology - ASIACRYPT 2020*. Ed. by Shiho Moriai and Huaxiong Wang. Vol. 12491. Lecture Notes in Computer Science. Springer, 2020, pp. 477–506. DOI: 10.1007/978-3-030-64837-4_16.

[EMJ17] Nadia El Mrabet and Marc Joye. *Guide to Pairing-Based Cryptography*. Chapman and Hall/CRC, 2017.

[EP10] Christian Eder and John Edward Perry. "F5C: A variant of Faugère's F5 algorithm with reduced Gröbner bases." In: *Journal of Symbolic Computation* 45.12 (2010), pp. 1442–1458. DOI: 10.1016/j.jsc.2010.06.019.

[ER13] Christian Eder and Bjarke Hammersholt Roune. "Signature Rewriting in Gröbner Basis Computation." In: *International Symposium on Symbolic and Algebraic Computation - ISSAC 2013*. Ed. by Manuel Kauers. ACM, 2013, pp. 331–338. DOI: 10.1145/2465506.2465522.

[Fau02] Jean-Charles Faugère. "A New Efficient Algorithm For Computing Gröbner Bases without Reduction to Zero (F5)." In: *International Symposium on Symbolic and Algebraic Computation - ISSAC 2002*. Ed. by Teo Mora. ACM, 2002, pp. 75–83. DOI: 10.1145/780506.780516.

[Fau99] Jean-Charles Faugère. "A New Efficient Algorithm for Computing Gröbner Bases (F4)." In: *Journal of Pure and Applied Algebra* 139 (1999), pp. 61–88. DOI: 10.1016/S0022-4049(99)00005-5.

[FGH+14] Jean-Charles Faugère, Pierrick Gaudry, Louise Huot, and Guénaël Renault. "Sub-Cubic Change of Ordering for Gröbner Basis: A Probabilistic Approach." In: *International Symposium on Symbolic and Algebraic Computation - ISSAC 2014*. Ed. by Katsusuke Nabeshima, Kosaku Nagasaka, Franz Winkler, and Ágnes Szántó. ACM, 2014, pp. 170–177. DOI: 10.1145/2608628.2608669.

[FGL+93] Jean-Charles Faugère, Patrizia Gianni, Daniel Lazard, and Teo Mora. "Efficient Computation of Zero-Dimensional Gröbner Bases by Change of Ordering." In: *Journal of Symbolic Computation* 16.4 (1993), pp. 329–344. DOI: 10.1006/jsco.1993.1051.

[FGO+13] Jean-Charles Faugère, Valérie Gauthier-Umaña, Ayoub Otmani, Ludovic Perret, and Jean-Pierre Tillich. "A Distinguisher for High-Rate McEliece Cryptosystems." In: *IEEE Transactions on Information Theory* 59.10 (2013), pp. 6830–6844. DOI: 10.1109/TIT.2013.2272036.

[FGP+15] Jean-Charles Faugère, Danilo Gligoroski, Ludovic Perret, Simona Samardjiska, and Enrico Thomae. "A Polynomial-Time Key-Recovery Attack on MQQ Cryptosystems." In: *Public-Key Cryptography - PKC 2015*. Ed. by Jonathan Katz. Vol. 9020. Lecture Notes in Computer Science. Springer, 2015, pp. 150–174. DOI: 10.1007/978-3-662-46447-2_7.

[FJ03] Jean-Charles Faugère and Antoine Joux. "Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems Using Gröbner Bases." In: *Advances in Cryptology - CRYPTO 2003*. Ed. by Dan Boneh. Vol. 2729. Lecture Notes in Computer Science. Springer, 2003, pp. 44–60. DOI: 10.1007/978-3-540-45146-4_3.

[FM11a] Jean-Charles Faugère and Chenqi Mou. "Fast Algorithm for Change of Ordering of Zero-Dimensional Gröbner Bases with Sparse Multiplication Matrices." In: *International Symposium on Symbolic and Algebraic Computation - ISSAC 2011*. Ed. by Éric Schost and Ioannis Z. Emiris. ACM, 2011, pp. 115–122. DOI: 10.1145/1993886.1993908.

[FM11b] Jean-Charles Faugère and Chenqi Mou. "Fast algorithm for Change of Ordering of Zero-Dimensional Gröbner Bases with Sparse Multiplication Matrices." In: *International Symposium on Symbolic and Algebraic Computation - ISSAC 2011*. Ed. by Éric

Schost and Ioannis Z. Emiris. ACM, 2011, pp. 115–122. DOI: 10.1145/1993886.1993908.

[FM17] Jean-Charles Faugère and Chenqi Mou. "Sparse FGLM Algorithms." In: *Journal of Symbolic Computation* 80 (2017), pp. 538–569. ISSN: 0747-7171. DOI: 10.1016/j.jsc.2016.07.025.

[FPP14] Jean-Charles Faugère, Ludovic Perret, and Frédéric de Portzamparc. "Algebraic Attack against Variants of McEliece with Goppa Polynomial of a Special Form." In: *Advances in Cryptology - ASIACRYPT 2014*. Ed. by Palash Sarkar and Tetsu Iwata. Vol. 8873. Lecture Notes in Computer Science. Springer, 2014, pp. 21–41. DOI: 10.1007/978-3-662-45611-8_2.

[Frö85] Ralf Fröberg. "An Inequality for Hilbert Series of Graded Algebras." In: *Mathematica Scandinavica* 56 (1985), pp. 117–144. DOI: 10.7146/math.scand.a-12092.

[Frö98] Ralf Fröberg. *An Introduction to Gröbner Bases*. Pure and Applied Mathematics. Wiley, 1998.

[FS86] Amos Fiat and Adi Shamir. "How to Prove Yourself: Practical Solutions to Identification and Signature Problems." In: *Advances in Cryptology - CRYPTO 1986*. Ed. by Andrew M. Odlyzko. Vol. 263. Lecture Notes in Computer Science. Springer, 1986, pp. 186–194. DOI: 10.1007/3-540-47721-7_12.

[FTI+17] Yuki Funabiki, Yosuke Todo, Takanori Isobe, and Masakatu Morii. "Improved Integral Attack on HIGHT." In: *Australasian Conference on Information Security and Privacy - ACISP 2017*. Ed. by Josef Pieprzyk and Suriadi Suriadi. Vol. 10342. Lecture Notes in Computer Science. Springer, 2017, pp. 363–383. DOI: 10.1007/978-3-319-60055-0_19.

[GCL92] Keith Geddes, Stephen Czapor, and George Labahn. *Algorithms for Computer Algebra*. Springer, 1992. DOI: 10.1007/b102438.

[GCR+21] Aaron Geary, Marco Calderini, Constanza Riera, and Pantelimon Stanica. "Higher Order c-Differentials." In: *CoRR* abs/2111.04661 (2021). arXiv: 2111.04661.

[Geno7] Giulio Genovese. "Improving the algorithms of Berlekamp and Niederreiter for Factoring Polynomials over Finite Fields." In: *Journal of Symbolic Computatation* 42.1-2 (2007), pp. 159–177. DOI: 10.1016/j.jsc.2006.02.007.

[GGP+13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. "Quadratic Span Programs and Succinct NIZKs without PCPs." In: *Advances in Cryptology - EUROCRYPT 2013*. Ed. by Thomas Johansson and Phong Q. Nguyen. Vol. 7881. Lecture Notes in Computer Science. Springer, 2013, pp. 626–645. DOI: 10.1007/978-3-642-38348-9_37.

[GHR+23] Lorenzo Grassi, Yonglin Hao, Christian Rechberger, Markus Schofnegger, Roman Walch, and Qingju Wang. "Horst Meets Fluid-SPN: Griffin for Zero-Knowledge Applications." In: *Advances in Cryptology - CRYPTO 2023*. Ed. by Helena Handschuh and Anna Lysyanskaya. Vol. 14083. Lecture Notes in Computer Science. Springer, 2023, pp. 573–606. DOI: 10.1007/978-3-031-38548-3_19.

[Giu84] Marc Giusti. "Some Effectivity Problems in Polynomial Ideal Theory." In: *International Symposion on Symbolic and Algebraic Manipulation - EUROSAM 1984*. Ed. by John P. Fitch. Vol. 174. Lecture Notes in Computer Science. Springer, 1984, pp. 159–171. DOI: 10.1007/BFb0032839.

[Giu85] Marc Giusti. "A Note on the Complexity of Constructing Standard Bases." In: *European Conference on Computer Algebra - EUROCAL 1985*. Ed. by B. F. Caviness. Vol. 204. Lecture Notes in Computer Science. Springer, 1985, pp. 411–412. DOI: 10.1007/3-540-15984-3_300.

[GJM+11] Bertoni Guido, Daemen Joan, P Michaël, and VA Gilles. *Cryptographic Sponge Functions*. Tech. rep. 2011. URL: https://keccak.team/files/CSF-0.1.pdf.

[GKK+19] Lorenzo Grassi, Daniel Kales, Dmitry Khovratovich, Arnab Roy, Christian Rechberger, and Markus Schofnegger. "Starkad and Poseidon: New Hash Functions for Zero Knowledge Proof Systems." In: *IACR Cryptology ePrint Archive* (2019), p. 458. URL: https://eprint.iacr.org/2019/458.

[GKL+22] Lorenzo Grassi, Dmitry Khovratovich, Reinhard Lüftenegger, Christian Rechberger, Markus Schofnegger, and Roman Walch. "Reinforced Concrete: A Fast Hash Function for Verifiable Computation." In: *SIGSAC Computer and Communications Security - CCS 2022*. Ed. by Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi. ACM, 2022, pp. 1323–1335. DOI: 10.1145/3548606.3560686.

[GKL+23] Lorenzo Grassi, Dmitry Khovratovich, Reinhard Lüftenegger, Christian Rechberger, Markus Schofnegger, and Roman Walch. "Monolith: Circuit-Friendly Hash Functions with New Nonlinear Layers for Fast and Constant-Time Implementations." In: *IACR Cryptology ePrint Archive* (2023), p. 1025. URL: https://eprint.iacr.org/2023/1025.

[GKR+21] Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. "Poseidon: A New Hash Function for Zero-Knowledge Proof Systems." In: *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*. Ed. by Michael Bailey and Rachel Greenstadt. USENIX Association, 2021, pp. 519–535. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/grassi.

[GKR+22] Lorenzo Grassi, Dmitry Khovratovich, Sondre Rønjom, and Markus Schofnegger. "The Legendre Symbol and the Modulo-2 Operator in Symmetric Schemes over $GF(p)^n$: Preimage Attack on Full Grendel." In: *IACR Transactions on Symmetric Cryptolology* 2022.1 (2022), pp. 5–37. DOI: 10.46586/tosc.v2022.i1.5-37.

[GKS23] Lorenzo Grassi, Dmitry Khovratovich, and Markus Schofnegger. "Poseidon2: A Faster Version of the Poseidon Hash Function." In: *Progress in Cryptology - AFRICACRYPT 2023*. Ed. by Nadia El Mrabet, Luca De Feo, and Sylvain Duquesne. Vol. 14064. Lecture Notes in Computer Science. Springer, 2023, pp. 177–203. DOI: 10.1007/978-3-031-37679-5_8.

[GLR+20] Lorenzo Grassi, Reinhard Lüftenegger, Christian Rechberger, Dragos Rotaru, and Markus Schofnegger. "On a Generalization of Substitution Permutation Networks: The HADES Design Strategy." In: *Advances in Cryptology - EUROCRYPT 2020*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12106. Lecture Notes in Computer Science. Springer, 2020, pp. 674–704. DOI: 10.1007/978-3-030-45724-2_23.

[GM86] Rüdiger Gebauer and Hans Michael Möller. "Buchberger's Algorithm and Staggered Linear Bases." In: *International Symposium on Symbolic and Algebraic Computation - ISSAC 1986*. ACM, 1986, pp. 218–221. DOI: 10.1145/32439.32482.

[GM88] Rüdiger Gebauer and Hans Michael Möller. "On an Installation of Buchberger's Algorithm." In: *Journal of Symbolic Computation* 6 (1988), pp. 275–286. DOI: 10.1016/S0747-7171(88)80048-8.

[Gol01] Oded Goldreich. *The Foundations of Cryptography - Volume 1: Basic Techniques*. Cambridge University Press, 2001. DOI: 10.1017/CBO9780511546891.

[Gol08] Oded Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008. DOI: 10.1017/CBO9780511804106.

[GOP+22] Lorenzo Grassi, Silvia Onofri, Marco Pedicini, and Luca Sozzi. "Invertible Quadratic Non-Linear Layers for MPC-/FHE-/ZK-Friendly Schemes over Fnp Application to Poseidon." In: *IACR Transactions on Symmetric Cryptology* 2022.3 (2022), pp. 20–72. DOI: 10.46586/tosc.v2022.i3.20-72.

[GPR21] Lior Goldberg, Shahar Papini, and Michael Riabzev. "Cairo - a Turing-complete STARK-friendly CPU architecture." In: *IACR Cryptology ePrint Archive* (2021), p. 1063. URL: https://eprint.iacr.org/2021/1063.

[Gro16] Jens Groth. "On the Size of Pairing-Based Non-interactive Arguments." In: *Advances in Cryptology - EUROCRYPT 2016*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9666. Lecture Notes in Computer Science. Springer, 2016, pp. 305–326. DOI: 10.1007/978-3-662-49896-5_11.

[GRR+16] Lorenzo Grassi, Christian Rechberger, Dragos Rotaru, Peter Scholl, and Nigel P. Smart. "MPC-Friendly Symmetric Key Primitives." In: *SIGSAC Computer and Communications Security - CCS 2016*. Ed. by Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi. ACM, 2016, pp. 430–443. DOI: 10.1145/2976749.2978332.

[GRS21] Lorenzo Grassi, Christian Rechberger, and Markus Schofnegger. "Proving Resistance Against Infinitely Long Subspace Trails: How to Choose the Linear Layer." In: *IACR Transactions on Symmetric Cryptology* 2021.2 (2021), pp. 314–352. DOI: 10.46586/tosc.v2021.i2.314-352.

[GW20] Ariel Gabizon and Zachary J. Williamson. "Plookup: A Simplified Polynomial Protocol for Lookup Tables." In: *IACR Cryptology ePrint Archive* (2020), p. 315. URL: https://eprint.iacr.org/2020/315.

[GWC19] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. "PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive Arguments of Knowledge." In: *IACR Cryptology ePrint Archive* (2019), p. 953. URL: https://eprint.iacr.org/2019/953.

[Has] *Hash Functions for Zero-Knowledge Applications Zoo*. IAIK, Graz University of Technology. Aug. 2021. URL: https://extgit.iaik.tugraz.at/krypto/zkfriendlyhashzoo.

[Hau20] Manuel Hauke. "Signature Gröbner bases - A comprehensive survey and a new algorithmic approach." In: *Master's Thesis, Graz University of Technology* (2020). URL: https://diglib.tugraz.at/download.php?id=608275af23dc4&location=browse.

[Hel80] Martin Hellman. "A Cryptanalytic Time-Memory Trade-Off." In: *IEEE Transactions on Information Theory* 26.4 (1980), pp. 401–406. DOI: 10.1109/TIT.1980.1056220.

[HL14] Jialin Huang and Xuejia Lai. "What is the Effective Key Length for a Block Cipher: An Attack on Every Practical Block Cipher." In: *Science China Information Sciences* 57.7 (2014), pp. 1–11. DOI: 10.1007/s11432-014-5096-6.

[HLL+22] Manuel Hauke, Lukas Lamster, Reinhard Lüftenegger, and Christian Rechberger. "A Signature-Based Gröbner Basis Algorithm with Tail-Reduced Reductors (M5GB)." In: *IACR Cryptology ePrint Archive* (2022), p. 987. URL: https://eprint.iacr.org/2022/987.

[Hog06] Leslie Hogben. *Handbook of Linear Algebra*. CRC Press, 2006.

[Hop19] Daira Hopwood. *Zcono Conference Notes*. 2019. URL: https://www.zfnd.org/zcon/0/workshop-notes/Zcon0%20Circuit%20Optimisation%20handout.pdf.

[Hor72] Ellis Horowitz. "A Fast Method for Interpolation Using Preconditioning." In: *Information Processing Letters* 1.4 (1972), pp. 157–163. DOI: 10.1016/0020-0190(72)90050-6.

[HSW+20] Kai Hu, Siwei Sun, Meiqin Wang, and Qingju Wang. "An Algebraic Formulation of the Division Property: Revisiting Degree Evaluations, Cube Attacks, and Key-Independent Sums." In: *Advances in Cryptology - ASIACRYPT 2020*. Ed. by Shiho Moriai and Huaxiong Wang. Vol. 12491. Lecture Notes in Computer Science. Springer, 2020, pp. 446–476. DOI: 10.1007/978-3-030-64837-4_15.

[JK97] Thomas Jakobsen and Lars R. Knudsen. "The Interpolation Attack on Block Ciphers." In: *Fast Software Encryption - FSE 1997*. Ed. by Eli Biham. Vol. 1267. Lecture Notes in Computer Science. Springer, 1997, pp. 28–40. DOI: 10.1007/BFb0052332.

[KBN09] Dmitry Khovratovich, Alex Biryukov, and Ivica Nikolic. "Speeding up Collision Search for Byte-Oriented Hash Functions." In: *Topics in Cryptology - CT-RSA 2009*. Ed. by Marc Fischlin. Vol. 5473. Lecture Notes in Computer Science. Springer, 2009, pp. 164–181. DOI: 10.1007/978-3-642-00862-7_11.

[Ker83a] Auguste Kerckhoffs. "La Cryptographie Militaire." In: *Journal des Sciences Militaires* 9 (1883), pp. 5–38.

[Ker83b] Auguste Kerckhoffs. "La Cryptographie Militaire." In: *Journal des Sciences Militaires* 9 (1883), pp. 161–191.

[KL21] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. 3rd ed. Chapman and Hall/CRC, 2021. DOI: 10.1201/9781351133036.

[Knu94a] Lars R. Knudsen. "Truncated and Higher Order Differentials." In: *Fast Software Encryption - FSE 1994*. Ed. by Bart Preneel. Vol. 1008. Lecture Notes in Computer Science. Springer, 1994, pp. 196–211. DOI: 10.1007/3-540-60590-8_16.

[Knu94b] Lars Ramkilde Knudsen. "Block Ciphers - Analysis, Design and Applications." PhD thesis. Aarhus Universitet. Department of Computer Science, 1994.

[Knu98] Lars Knudsen. *DEAL - A 128-bit Block Cipher*. Tech. rep. 1998. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=9530ed5c22f52d7889c2561dd93b41fb3e41ec93.

[KP02] Sergei Konyagin and Francesco Pappalardi. "Enumerating Permutation Polynomials over Finite Fields by Degree." In: *Finite Fields and their Application* 8 (2002), pp. 548–553. DOI: 10.1006/ffta.2002.0363.

[KR00] Martin Kreuzer and Lorenzo Robbiano. *Computational Commutative Algebra 1*. Berlin: Springer, 2000. DOI: 10.1007/978-3-540-70628-1.

[KR05] Martin Kreuzer and Lorenzo Robbiano. *Computational Commutative Algebra 2*. Berlin: Springer, 2005. DOI: 10.1007/3-540-28296-3.

[KR07] Lars R. Knudsen and Vincent Rijmen. "Known-Key Distinguishers for Some Block Ciphers." In: *Advances in Cryptology - ASIACRYPT 2007*. Ed. by Kaoru Kurosawa. Vol. 4833. Lecture Notes in Computer Science. Springer, 2007, pp. 315–324. DOI: 10.1007/978-3-540-76900-2_19.

[KR11] Lars R. Knudsen and Matthew Robshaw. *The Block Cipher Companion*. Information Security and Cryptography. Springer, 2011. DOI: 10.1007/978-3-642-17342-4.

[KR21] Nathan Keller and Asaf Rosemarin. "Mind the Middle Layer: The HADES Design Strategy Revisited." In: *Advances in Cryptology - EUROCRYPT 2021*. Ed. by Anne Canteaut and François-Xavier Standaert. Vol. 12697. Lecture Notes in Computer Science. Springer, 2021, pp. 35–63. DOI: 10.1007/978-3-030-77886-6_2.

[KS98] Erich L. Kaltofen and Victor Shoup. "Subquadratic-Time Factoring of Polynomials over Finite Fields." In: *Mathematics of Computation* 67.223 (1998), pp. 1179–1197. DOI: 10.1090/S0025-5718-98-00944-2.

[KS99] Aviad Kipnis and Adi Shamir. "Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization." In: *Advances in Cryptology - CRYPTO 1999*. Ed. by Michael J. Wiener. Vol. 1666. Lecture Notes in Computer Science. Springer, 1999, pp. 19–30. DOI: 10.1007/3-540-48405-1_2.

[KU11] Kiran S. Kedlaya and Christopher Umans. "Fast Polynomial Factorization and Modular Composition." In: *SIAM Journal on Computing* 40.6 (2011), pp. 1767–1802. DOI: 10.1137/08073408X.

[Kun73] Hsiang-Tsung Kung. *Fast Evaluation and Interpolation*. Tech. rep. Department of Computer Science, Carnegie-Mellon University, 1973. URL: http://www.eecs.harvard.edu/~htk/publication/1973-cmu-cs-technical-report-kung.pdf.

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. "Constant Size Commitments to Polynomials and Their Applications." In: *Advances in Cryptology - ASIACRYPT 2010*. Ed. by Masayuki Abe. Vol. 6477. Lecture Notes in Computer Science. Springer, 2010, pp. 177–194. DOI: 10.1007/978-3-642-17373-8_11.

[Lai94] Xuejia Lai. "Higher Order Derivatives and Differential Cryptanalysis." In: *Communications and Cryptography: Two Sides of One Tapestry* (1994), pp. 227–233. DOI: 10.1007/978-1-4615-2694-0_23.

[LAW+23] Fukang Liu, Ravi Anand, Libo Wang, Willi Meier, and Takanori Isobe. "Coefficient Grouping: Breaking Chaghri and More." In: *Advances in Cryptology - EUROCRYPT 2023*. Ed. by Carmit Hazay and Martijn Stam. Vol. 14007. Lecture Notes in Computer Science. Springer, 2023, pp. 287–317. DOI: 10.1007/978-3-031-30634-1_10.

[Laz01] Daniel Lazard. "Solving systems of algebraic equations." In: *SIGSAM Bulletin* 35.3 (2001), pp. 11–37. DOI: 10.1145/569746.569750.

[Laz79] Daniel Lazard. "Systems of Algebraic Equations." In: *International Symposium on Symbolic and Algebraic Computation - EUROSAM 1979*. Ed. by Edward W. Ng. Vol. 72. Lecture Notes in Computer Science. Springer, 1979, pp. 88–94.

[Laz83] Daniel Lazard. "Gröbner Bases, Gaussian Elimination and Resolution of Systems of Algebraic Equations." In: *European Conference on Computer Algebra - EUROCAL 1983*. Ed. by J. A. van Hulzen. Vol. 162. Lecture Notes in Computer Science. Springer, 1983, pp. 146–156.

[LLB03] Ting Li, Zhenjiang Lin, and Fengshan Bai. "Heuristic Methods for Computing the Minimal Multi-Homogeneous Bézout Number." In: *Applied Mathematics and Computation* 146.1 (2003), pp. 237–256. DOI: 10.1016/S0096-3003(02)00540-4.

[LM90] Xuejia Lai and James L. Massey. "A Proposal for a New Block Encryption Standard." In: *Advances in Cryptology - EUROCRYPT 1990*. Ed. by Ivan Damgård. Vol. 473. Lecture Notes in Computer Science. Springer, 1990, pp. 389–404. DOI: 10.1007/3-540-46877-3_35.

[LMR+09] Mario Lamberger, Florian Mendel, Christian Rechberger, Vincent Rijmen, and Martin Schläffer. "Rebound Distinguishers: Results on the Full Whirlpool Compression Function." In: *Advances in Cryptology - ASIACRYPT 2009*. Ed. by Mitsuru Matsui. Vol. 5912. Lecture Notes in Computer Science. Springer, 2009, pp. 126–143. DOI: 10.1007/978-3-642-10366-7_8.

[LN96] Rudolf Lidl and Harald Niederreiter. *Finite Fields*. 2nd ed. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1996. DOI: 10.1017/CBO9780511525926.

[LP19] Chaoyun Li and Bart Preneel. "Improved Interpolation Attacks on Cryptographic Primitives of Low Algebraic Degree." In: *Selected Areas in Cryptography - SAC 2019, Revised Selected Papers*. Ed. by Kenneth G. Paterson and Douglas Stebila. Vol. 11959. Lecture Notes in Computer Science. Springer, 2019, pp. 171–193. DOI: 10.1007/978-3-030-38471-5_8.

[LR88] Michael Luby and Charles Rackoff. "How to Construct Pseudorandom Permutations from Pseudorandom Functions." In: *SIAM Journal on Computing* 17.2 (1988), pp. 373–386. DOI: 10.1137/0217022.

[Mag] "The Magma Algebra System I: The User Language." In: *Journal of Symbolic Computation* 24.3 (1997), pp. 235–265. DOI: 10.1006/jsco.1996.0125.

[Mat93] Mitsuru Matsui. "Linear Cryptanalysis Method for DES Cipher." In: *Advances in Cryptology - EUROCRYPT 1993*. Ed. by Tor Helleseth. Vol. 765. Lecture Notes in Computer Science. Springer, 1993, pp. 386–397. DOI: 10.1007/3-540-48285-7_33.

[MBK+19] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. "Sonic: Zero-Knowledge SNARKs from Linear-Size Universal and Updatable Structured Reference Strings." In: *SIGSAC Conference on Computer and Communications Security - CCS 2019*. Ed. by Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz. ACM, 2019, pp. 2111–2128. DOI: 10.1145/3319535.3339817.

[MF21] Arno Mittelbach and Marc Fischlin. *The Theory of Hash Functions and Random Oracles: An Approach to Modern Cryptography*. Information Security and Cryptography. Springer, 2021. DOI: 10.1007/978-3-030-63287-8.

[MH20] Benyamin M.-Alizadeh and Amir Hashemi. "Deterministic Normal Position Transformation and its Applications." In: *Theoretical Computer Science* 842 (2020), pp. 50–64. DOI: 10.1016/j.tcs.2020.07.025.

[MM07] Gregorio Malajovich and Klaus Meer. "Computing Minimal Multi-Homogeneous Bezout Numbers Is Hard." In: *Theory Comput. Syst.* 40.4 (2007), pp. 553–570. DOI: 10.1007/s00224-006-1322-y.

[MR02] Sean Murphy and Matthew J. B. Robshaw. "Essential Algebraic Structure within the AES." In: *Advances in Cryptology - CRYPTO 2002*. Ed. by Moti Yung. Vol. 2442. Lecture Notes in Computer Science. Springer, 2002, pp. 1–16. DOI: 10.1007/3-540-45708-9_1.

[MRS+09] Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. "The Rebound Attack: Cryptanalysis of Reduced Whirlpool and Grøstl." In: *Fast Software Encryption - FSE 2009*. Ed. by Orr Dunkelman. Vol. 5665. Lecture Notes in Computer Science. Springer, 2009, pp. 260–276. DOI: 10.1007/978-3-642-03317-9_16.

[MS17] Rusydi H. Makarim and Marc Stevens. "M4GB: An Efficient Gröbner-Basis Algorithm." In: *International Symposium on Symbolic and Algebraic Computation - ISSAC 2017*. Ed. by Michael A. Burr, Chee K. Yap, and Mohab Safey El Din. ACM, 2017, pp. 293–300. DOI: 10.1145/3087604.3087638.

[MS87] Alexander Morgan and Andrew Sommese. "A Homotopy for Solving General Polynomial Systems that Respects m Homogeneous Structures." In: *Applied Mathematics and Computation* 24.2 (1987), pp. 101–113. DOI: 10.1016/0096-3003(87)90063-4.

[MSK98] Shiho Moriai, Takeshi Shimoyama, and Toshinobu Kaneko. "Higher Order Differential Attak of CAST Cipher." In: *Fast Software Encryption - FSE 1998*. Ed. by Serge Vaudenay. Vol. 1372. Lecture Notes in Computer Science. Springer, 1998, pp. 17–31. DOI: 10.1007/3-540-69710-1_2.

[OWW+20] Alex Ozdemir, Riad S. Wahby, Barry Whitehat, and Dan Boneh. "Scaling Verifiable Computation Using Efficient Set Accumulators." In: *USENIX Security Symposium 2020*. Ed. by Srdjan Capkun and Franziska Roesner. USENIX Association, 2020, pp. 2075–2092. URL: https://www.usenix.org/conference/usenixsecurity20/presentation/ozdemir.

[PFM+22] Luke Pearson, Joshua Brian Fitzgerald, Héctor Masip, Marta Bellés-Muñoz, and Jose Luis Muñoz-Tapia. "PlonKup: Reconciling PlonK with Plookup." In: *IACR Cryptology ePrint Archive* (2022), p. 86. URL: https://eprint.iacr.org/2022/086.

[PHG+13] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. "Pinocchio: Nearly Practical Verifiable Computation." In: *Symposium on Security and Privacy - SP 2013*. IEEE Computer Society, 2013, pp. 238–252. DOI: 10.1109/SP.2013.47.

[PSS21] Alexey Pertsev, Roman Semenov, and Roman Storm. *Tornado Cash Privacy Solution Version 1.4*. Tech. rep. 2021. URL: https://berkeley-defi.github.io/assets/material/Tornado%20Cash%20Whitepaper.pdf.

[RAS21] Arnab Roy, Elena Andreeva, and Jan Ferdinand Sauer. "Interpolation Cryptanalysis of Unbalanced Feistel Networks with Low Degree Round Functions." In: *Selected Areas in Cryptography - SAC 2021, Revised Selected Papers*. Springer. 2021, pp. 273–300. DOI: 10.1007/978-3-030-81652-0_11.

[RDP+96] Vincent Rijmen, Joan Daemen, Bart Preneel, Antoon Bosselaers, and Erik De Win. "The Cipher SHARK." In: *Fast Software Encryption - FSE 1996*. Ed. by Dieter Gollmann. Vol. 1039. Lecture Notes in Computer Science. Springer, 1996, pp. 99–111. DOI: 10.1007/3-540-60865-6_47.

[RKH10] Kamisetty Ramamohan Rao, Do Nyeon Kim, and Jae Jeong Hwang. *Fast Fourier Fransform: Algorithms and Applications*. Springer, 2010. DOI: 10.1007/978-1-4020-6629-0.

[Rob20] Borut Robic. *The Foundations of Computability Theory*. 2nd ed. Springer, 2020. DOI: 10.1007/978-3-662-62421-0.

[Rou13a] Bjarke Hammersholt Roune. "mathic." In: *GitHub repository* Commit: 66b5d74f8417459414cbf3753cfa9a0128483cbd (2013). https://github.com/broune/mathic.

[Rou13b] Bjarke Hammersholt Roune. "mathicgb." In: *GitHub repository* Commit: c72c945ba8e18e68e5650e7e4982b86e558abe6c (2013). URL: https://github.com/broune/mathicgb.

[RS12] Bjarke Hammersholt Roune and Michael Stillman. "Practical Gröbner Basis Computation." In: *International Symposium on Symbolic and Algebraic Computation - ISSAC 2012*. ACM, 2012, pp. 203–210. DOI: 10.1145/2442829.2442860.

[RST23] Arnab Roy, Matthias Johann Steiner, and Stefano Trevisani. "Arion: Arithmetization-Oriented Permutation and Hashing from Generalized Triangular Dynamical Systems." In: *CoRR* abs/2303.04639 (2023). arXiv: 2303.04639.

[SAD20] Alan Szepieniec, Tomer Ashur, and Siemen Dhooghe. "Rescue-Prime: A Standard Specification (SoK)." In: *IACR Cryptology ePrint Archive* (2020), p. 1143. URL: https://eprint.iacr.org/2020/1143.

[Sal23] Robin Salen. "Two Additional Instantiations from the Tip5 Hash Function Construction." In: (2023). URL: https://toposware.com/paper_tip5.pdf.

[Sha] *The SHA-3 Cryptographic Hash Algorithm Competition*. Nov 2007 – Oct 2012. URL: http://csrc.nist.gov/groups/ST/hash/sha-3/index.html.

[Sho09] Victor Shoup. *A computational Introduction to Number Theory and Algebra*. 2nd ed. Cambridge University Press, 2009. DOI: 10.1017/CBO9780511814549.

[SK98] Takeshi Shimoyama and Toshinobu Kaneko. "Quadratic Relation of S-box and Its Application to the Linear Attack of Full Round DES." In: *Advances in Cryptology - CRYPTO 1998*. Ed. by Hugo Krawczyk. Vol. 1462. Lecture Notes in Computer Science. Springer, 1998, pp. 200–211. DOI: 10.1007/BFb0055729.

[SLS+23] Alan Szepieniec, Alexander Lemmens, Jan Ferdinand Sauer, and Bobbin Threadbare. "The Tip5 Hash Function for Recursive STARKs." In: *IACR Cryptology ePrint Archive* (2023), p. 107. URL: https://eprint.iacr.org/2023/107.

[Spa12] Pierre-Jean Spaenlehauer. "Solving Multi-H and Determinantal Systems: Algorithms, Complexity, Applications." PhD thesis. Pierre and Marie Curie University, Paris, France, 2012. URL: https://tel.archives-ouvertes.fr/tel-01110756.

[SS21] Jan Ferdinand Sauer and Alan Szepieniec. "SoK: Gröbner Basis Algorithms for Arithmetization Oriented Ciphers." In: *IACR Cryptology ePrint Archive* (2021), p. 870. URL: https://eprint.iacr.org/2021/870.

[SSL15] Kazuo Sakiyama, Yu Sasaki, and Yang Li. *Security of Block Ciphers - From Algorithm Design to Hardware Implementation*. Wiley, 2015. URL: http://eu.wiley.com/WileyCDA/WileyTitle/productCd-1118660013.html.

[Ste+19] William Stein et al. *Sage Mathematics Software Version 8.6*. The Sage Development Team. 2019. URL: http://www.sagemath.org.

[Sze21] Alan Szepieniec. "On the Use of the Legendre Symbol in Symmetric Cipher Design." In: *IACR Cryptology ePrint Archive* (2021), p. 984. URL: https://eprint.iacr.org/2021/984.

[Tak15] Takanori Yasuda, Okayama University of Science. *Fukuoka MQ Challenge*. Accessed: 2023-08-04. 2015. URL: https://www.mqchallenge.org/.

[Tha22] Justin Thaler. "Proofs, Arguments, and Zero-Knowledge." In: *Foundations and Trends in Privacy and Security* 4.2-4 (2022), pp. 117–660. DOI: 10.1561/3300000030.

[TM16] Yosuke Todo and Masakatu Morii. "Bit-Based Division Property and Application to Simon Family." In: *Fast Software Encryption - FSE 2016*. Ed. by Thomas Peyrin. Vol. 9783. Lecture Notes in Computer Science. Springer, 2016, pp. 357–377. DOI: 10.1007/978-3-662-52993-5_18.

[Tod15] Yosuke Todo. "Structural Evaluation by Generalized Integral Property." In: *Advances in Cryptology - EUROCRYPT 2015*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9056. Lecture Notes in Computer Science. Springer, 2015, pp. 287–314. DOI: 10.1007/978-3-662-46800-5_12.

[TPD21] Chengdong Tao, Albrecht Petzoldt, and Jintai Ding. "Efficient Key Recovery for All HFE Signature Variants." In: *Advances in Cryptology - CRYPTO 2021*. Ed. by Tal Malkin and Chris Peikert. Vol. 12825. Lecture Notes in Computer Science. Springer, 2021, pp. 70–93. DOI: 10.1007/978-3-030-84242-0_4.

[Vas07] Oleg Nikolaevich Vasilenko. *Number-Theoretic Algorithms in Cryptography*. Vol. 232. Translations of Mathematical Monographs. American Mathematical Society, 2007. DOI: 10.1090/mmono/232.

[VTJ14] Henk Van Tilborg and Sushil Jajodia, eds. *Encyclopedia of Cryptography and Security*. Springer, 2014. DOI: 10.1007/978-1-4419-5906-5.

[WHT+18] Qingju Wang, Yonglin Hao, Yosuke Todo, Chaoyun Li, Takanori Isobe, and Willi Meier. "Improved Division Property Based Cube Attacks Exploiting Algebraic Properties of Superpoly." In: *Advances in Cryptology - CRYPTO 2018*. Ed. by Hovav Shacham and Alexandra Boldyreva. Vol. 10991. Lecture Notes in Computer Science. Springer, 2018, pp. 275–305. DOI: 10.1007/978-3-319-96884-1_10.

[Wil20] Zac Williamson. *zkSummit: Plookup*. 2020. URL: https://youtu.be/Vdlc1CmRYRY?t=1564.

[Woo14] Gavin Wood. *Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper. Byzantinum Version 2d0661f-2018-11-08*. 2014.

[WSM+11] Meiqin Wang, Yue Sun, Nicky Mouha, and Bart Preneel. "Algebraic Techniques in Differential Cryptanalysis Revisited." In: *Australasian Conference on Information Security and Privacy - ACISP 2011*. Ed. by Udaya Parampalli and Philip Hawkes. Vol. 6812. Lecture Notes in Computer Science. Springer, 2011, pp. 120–141. DOI: 10.1007/978-3-642-22497-3_9.

[WW11] Manuela Wiesinger-Widi. "Gröbner Bases and Generalized Sylvester Matrices." PhD thesis. Johannes Kepler University of Linz, 2011. URL: https://epub.jku.at/obvulihs/download/pdf/776913?originalFilename=true.

[YG00] Amr M. Youssef and Guang Gong. "On the Interpolation Attacks on Block Ciphers." In: *Fast Software Encryption - FSE 2000*. Ed. by Bruce Schneier. Vol. 1978. Lecture Notes in Computer Science. Springer, 2000, pp. 109–120. DOI: 10.1007/3-540-44706-7_8.

[ZZD+21] Haibo Zhou, Rui Zong, Xiaoyang Dong, Keting Jia, and Willi Meier. "Interpolation Attacks on Round-Reduced Elephant, Kravatte and Xoofff." In: *The Computer Journal* 64.4 (2021), pp. 628–638. DOI: 10.1093/comjnl/bxaa101.