Algebraic Cryptanalysis of Deterministic Symmetric Encryption

THÈSE N^{O} 6651 (2015)

PRÉSENTÉE LE 28 AOÛT 2015

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS
LABORATOIRE DE SÉCURITÉ ET DE CRYPTOGRAPHIE
PROGRAMME DOCTORAL EN INFORMATIQUE ET COMMUNICATIONS

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Petr SUŠIL

acceptée sur proposition du jury:

Prof. M. Pauly, président du jury
Prof. S. Vaudenay, directeur de thèse
Prof. J. Ding, rapporteur
Prof. N. Courtois, rapporteur
Prof. A. Lenstra, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2015

To my parents Aleš and Lenka

Abstract

Deterministic symmetric encryption is widely used in many cryptographic applications. The security of deterministic block and stream ciphers is evaluated using cryptanalysis. Cryptanalysis is divided into two main categories: statistical cryptanalysis and algebraic cryptanalysis. Statistical cryptanalysis is a powerful tool for evaluating the security but it often requires a large number of plaintext/ciphertext pairs which is not always available in real life scenario. Algebraic cryptanalysis requires a smaller number of plaintext/ciphertext pairs but the attacks are often underestimated compared to statistical methods. In algebraic cryptanalysis, we consider a polynomial system representing the cipher and a solution of this system reveals the secret key used in the encryption. The contribution of this thesis is twofold.
Firstly, we evaluate the performance of existing algebraic techniques with respect to number of plaintext/ciphertext pairs and their selection. We introduce a new strategy for selection of samples. We build this strategy based on cube attacks, which is a well-known technique in algebraic cryptanalysis. We use cube attacks as a fast heuristic to determine sets of plaintexts for which standard algebraic methods, such as Gröbner basis techniques or SAT solvers, are more efficient.
Secondly, we develop a new technique for algebraic cryptanalysis which allows us to speed-up existing Gröbner basis techniques. This is achieved by efficient finding special polynomials called mutants. Using these mutants in Gröbner basis computations and SAT solvers reduces the computational cost to solve the system.
Hence, both our methods are designed as tools for building polynomial system representing a cipher. Both tools can be combined and they lead to a significant speedup, even for very simple algebraic solvers.

keywords: algebraic cryptanalysis, symmetric encryption, KATAN32, LBlock, SIMON, cube attacks, selection of samples

Résumé

De nombreuses applications cryptographiques utilisent le chiffrement symétrique déterministe : le chiffrement par bloc et le chiffrement par flot. On évalue leur sécurité à l'aide de la cryptanalyse. Celle-ci est divisée en deux catégories principales : la cryptanalyse statistique et la cryptanalyse algébrique. La cryptanalyse statistique est un outil puissant qui permet d'évaluer la sécurité mais qui requiert souvent un grand nombre de couples de messages en clair et chiffrés ce qui n'est pas toujours faisable dans un scénario réaliste. La cryptanalyse algébrique requiert moins de couples, mais ces attaques sont souvent sous-estimées par rapport aux méthodes statistiques. En cryptanalyse algébrique, nous considérons un système d'équations polynomiales qui représente le système de chiffrement. La clé secrète utilisée pour le chiffrement est solution de ce système d'équations polynomiales.

Les contributions de cette thèse sont doubles. Tout d'abord, nous évaluons les performances des méthodes algébriques existantes par rapport à la quantité de couples nécessaires à l'attaque. Nous étudions aussi la façon de choisir ces couples et proposons une nouvelle stratégie. Nous basons cette stratégie sur les attaques dites *cube attacks* qui sont une technique très connue en cryptanalyse algébrique. Nous utilisons ces *cube attacks* afin de trouver rapidement des ensembles de textes clairs pour lesquels des méthodes algébriques standards, comme le calcul de bases de Gröbner ou les SAT-solveurs, sont plus efficaces.

Deuxièmement, nous développons une nouvelle technique en cryptanalyse algébrique qui nous permet d'accélérer le calcul de bases de Gröbner. Nous arrivons à ce résultat en trouvant de façon efficace des polynômes mutants, un type spécial de polynôme. L'utilisation de ces polynômes mutants dans le calcul de bases de Gröbner ou dans un SAT-solveur permet de réduire la complexité temps de ces algorithmes.

Nos deux méthodes sont conçues sous la forme d'outils qui permettent de construire un système d'équations polynomiales qui représente le chiffrement. Ces deux outils peuvent être combinés et permettent un gain de temps significatif, même pour des solveurs algébriques simples.

Mots-clés: cryptanalyse algébrique, chiffrement symétrique, KATAN, LBlock, SIMON, *cube attacks*, choix d'échantillons

Acknowledgment

Mainly, I would like to express my gratitude to my advisor Prof. Serge Vaudenay for giving me an opportunity to pursue Ph.D. studies in LASEC. I am greatful for his supervision as he has able to advise me in all areas of my research. I especially appreciated his patience, the freedom he gave me in my research and his guidance during my Ph.D. studies.
It is a great honor for me to have Prof. Arjen Lenstra, Prof. Jintai Ding, Prof. Nicolas Courtois and the jury president Prof. Mark Pauly in my commitee. I sincerely thank them for reading my dissertation and giving me valuable advices for improvements. I gratefully acknowledge the support of this thesis by Swiss National Science Foundation under grant number 200021_134860/1.
I would like to specifically thank to all my colleagues, ex-collegues and members of LASEC: Pouyan Sepehrdad, Alexandre Duc, Damian Vizár, Sonia Mihaela Bogos, Adeline Langlois, Handan Kilinç, Divesh Aggarwal, Aslı Bay, Jialin Huang, Rafik Chaabouni, Khaled Ouafi, Atefeh Mashatan, Miyako Ohkubo, Ioana Boureanu, Sebastian Faust, Reza Reyhanitabar, Katerina Mitrokotsa, Martin Vuagnoux and Jean-Philippe Aumasson. I also want to express my gratitude to our secretary Martine Corval, who was always supportive and helpful with many aspects of life at EPFL.
I especially thank Pouyan for the work we have done together during the time at LASEC and for bringing my attention to algebraic attacks. I thank Alexandre for endless discussions about algebraic attacks and his countless comments which helped me to shape the methods which are presented in this thesis. It was a great pleasure to share an office with Atefeh, Sebastian and Damian during my years at EPFL. During my time in LASEC, I could always get plenty of energy in form of delightful chocolate and cheerful discussions in the office of Sonia and Alexandre.
Furthermore, I would like to thank to all my friends who I met before and during my doctoral studies and all my family for their support.

Table of Contents

1 Introduction to cryptography

The word “cryptography” is derived from Greek κρύπτος [kryptós] which means “hidden, secret” and γραφειν [graphein] which means writing. Its focus is to study techniques for secure communication in hostile enviroment. Cryptography is divided into two principal categories: symmetric and asymmetric cryptography. In the symmetric setting, both the sender and the receiver share the same secret key κ. Meanwhile in asymmetric cryptography, the receiver has a private key and also publishes a public key which is the same for everyone who wishes to communicate with him.

In this thesis, we focus on symmetric setting where Alice and Bob communicate and Eve is trying to decrypt the communication as in Figure 1.1.

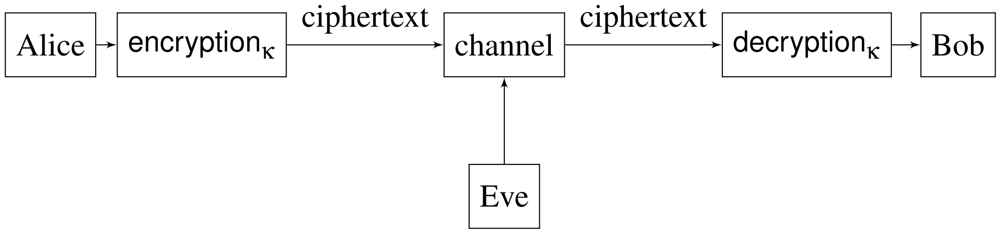


Figure 1.1: Secure communication

In 1883, Kerckhoffs stated six design principles for ciphers [Ker83]. The most famous one is: “It should not require secrecy, and it should not be a problem if it falls into enemy hands.” It was reformulated in 1945 by Shannon in [Sha49] as “one ought to design systems under the assumption that the enemy will immediately gain full familiarity with them”. Hence, the secrecy should only apply to the secret key. We follow the Kerckhoffs’ priniciple and assume that Eve knows the encryption and decryption algorithms but not κ. Moreover, we often allow her to query some $\mathsf{encryption}_\kappa$ and $\mathsf{decryption}_\kappa$ black boxes before she observes the ciphertext. Typically, she collects some plaintext/ciphertext pairs, finds key κ which is consistent with the samples and decrypts the ciphertext with κ. In what follows, we formalize the possible attack scenarios.

known plaintext attack. In this scenario, Eve gets access to plaintext-ciphertext pairs which are being transmitted. In practice, Eve can observe a predictable communication such as encrypted headers.

chosen plaintext attack. In this scenario, we assume Eve gets access to an encryption oracle. Hence, she can select plaintexts which would be encrypted. Obviously, Eve is more powerful in this scenario than in the previous case. In practice, it can be for instance a "visitor" of an unlocked unattended office. In the case of contactless wireless devices, it can be a fellow passenger of public transport who performs queries with our credit card or RFID chip which we use as an electronic door key.

chosen ciphertext attack. In this scenario, we assume Eve gets access to an decryption oracle and she can select ciphertexts for decryption. Attacks in chosen plaintext scenarios are often possible in chosen ciphertext scenario as well. In practice, we can consider a USB token which performs the encryption and decryption and Eve gains access to this device over a lunch break or a weekend.

ciphertext only attack. In this scenario, Eve gets access only to ciphertexts which are being transmitted.

In this thesis, we concentrate on deterministic symmetric encryption. The deterministic encryption refers to the fact that multiple encryptions of the same plaintext under the same secret key always leads to the same ciphertext. For instance, we consider block ciphers. A block cipher is a pair of algorithms $(\mathsf{encryption}_\kappa, \mathsf{decryption}_\kappa)$ both accepting two inputs: plaintext/ciphertext of mln bits and a key of kln bits. For example in 1977, Data Encryption Standard (DES) [Des77] was proposed as a standard for a protection of sensitive unclassified documents. In symmetric cryptography, we often use statistical methods to evaluate the security of a cipher or another primitive. The most prominent among statistical methods are linear and differential cryptanalysis. In linear cryptanalysis, we look for affine approximation of the cipher. It was first used in cryptanalysis of FEAL [MY92, OA94] and later, it was applied on DES [Mat93]. In differential cryptanalysis, we study how differences in plaintexts affect differences in ciphertexts. Then, observing the desired output difference (between two chosen or known plaintexts) suggests possible key values. The original design of DES was slightly modified in 1976 after consultation with NSA. The modification strenghten DES against differential cryptanalysis but weakened it against brute-force attacks. In 1991, Biham and Shamir showed DES can be broken with differential cryptanalysis [BS91] and in 1993, Matsui showed DES can be broken with linear cryptanalysis [Mat93]. Later in 1998, an international non-profit organization Electronic Frontier Foundation built DES cracker (Deep Crack) and showed DES can be broken in 56 hours using brute force attack. Following these results, NIST announced in 1997 a competition for a new encryption standard. In 2000, NIST announced the winner Rijndael of AES competition [DR02] and the AES standard was later published in [FIP01].

In cryptanalysis, statistical techniques are well explored and so far, they account for much greater success than algebraic cryptanalysis. However, they usually lead to a high data complexity and therefore, they are not well-suited for scenario where the attacker has limited access to our cryptographic device. Conversely, algebraic attacks can be successful even if the attacker has a limited access to an encryption/decryption device.

Algebraic cryptanalysis was considered as a tool for evaluation of security for a long time. In 1959, Shannon stated the following: “if we could show that solving a certain system requires at least as much work as solving a system of simultaneous equations in a large number of unknowns, of a complex type, then we would have a lower bound of sorts for the work characteristic”. In algebraic cryptanalysis, we model a cipher as a polynomial system with special variables corresponding to plaintext, ciphertext and key. We set plaintext-ciphertext pairs according to queries to an encryption/decryption oracle and we solve the system. This gives us the secret key. This problem can be mapped to a well-known NP-complete problem called “MQ”. The “MQ” takes as an input a multivariate polynomial system over $\mathbf{F}_2$ and the task is to decide if it has a solution/find a solution in $\mathbf{F}_2$. In 2002, advances in the XL algorithm and the XSL method led to over-optimistic assumptions about strength of algebraic cryptanalysis. It was assumed that AES is vulnerable to algebraic attacks [Sei02]. However up to now, the AES algorithm is considered secure and the initial glorification and subsequent failures of algebraic attacks inhibited the research in algebraic cryptanalysis in symmetric setting. The XL algorithm and XSL method belong to a wider family of algebraic tools which we will refer to as Gröbner basis techniques. These techniques have been well explored by Faugère in the F4 and F5 algorithms and in the subsequent sparse Gröbner basis algorithm [FSS14]. Another important tool of algebraic cryptanalysis are SAT solvers. In this case, we model a cipher as a boolean formula where we set the plaintext and ciphertext accordingly. We know that such formula is satisfiable and we use SAT solvers to find a satisfying assignment. In complexity theory, we refer to the “SAT” problem which is also NP-complete. “SAT” takes as an input a boolean formula and returns the satisfying assignment if this formula is satisfiable or empty set if it is not. To our knowledge, the SAT solvers were introduced into algebraic cryptography in [MM00]. Unlike the Gröbner basis techniques which are deterministic, SAT solvers usually rely on heuristics to find the satisfying assignment.

The algebraic cryptanalysis has already brought about several important results. Many schemes cryptanalysed by these techniques come from public key cryptography, as their algebraic structure is well suited for algebraic attacks. Several results about HFE can be found in [Cou01, Cou04b, DSW08, DH11], Broadcast NTRU in [DPD12], MQQ cryptosystem in [MDBW09, FØPG10] and other multivariate public key cryptosystems [DHN$^+$07]. In the symmetric setting, the algebraic cryptanalysis builds on the work [CM03, Cou03, Cou04a]. In later years, the stream cipher LILI was analysed in [AHDHS07] and Dragon-based cryptosystems were analysed in [BBD$^+$10]. Moreover, the cipher KeeLoq which is used in electronic door control system of cars (such as Toyota, Honda, Chrysler, Volkswagen, etc.) was analysed in [CBW08]. Furthermore, algebraic analysis of DES was given in [CB07]. Later development in algebraic techniques led to AIDA/Cube attacks [Vie07, DS09a]. The cube attack is applied against any tweakable blackbox polynomial. This blackbox polynomial represents a circuit to compute an output bit of a cipher. The tweakable polynomial means that we can select plaintexts and encryption keys. The blackbox polynomial is partially reconstructed by observing relations among inputs and outputs of the blackbox polynomial and we use this partial reconstruction in the online phase. The cube attacks are rarely successfull, as it is computationally expensive to find a good cube - set of plaintexts - which allows the attacker to find simple relations among key bits. The original cube attack was finding a cube producing a linear relation. The restriction was very significant. It speeds up the precomputation phase but it severly reduces the number of polynomials we can find. A cube attack leading to non-linear relations was explored in [ALRSS11]. An alternative approach for extension of cube attacks was considered in [DS11]. The authors tweaked a definition of a cube to reflect the behavioral of a cipher in first few rounds. They defined a so called “dynamic cube attack” which was used against the full version of Grain-128 [DS11].

About this dissertation

This dissertation consists of three chapters which follow an introduction to algebraic cryptanalysis. These chapters are based on publications related to algebraic cryptanalysis. The last chapter also contains some currently unpublished work. In our work, we focus on algebraic cryptanalysis of symmetric deterministic ciphers.

In Chapter 3, we consider a fundamental algorithm of algebraic cryptanalysis called ElimLin. This algorithm is used to simplify a polynomial system. However, the specification of the algorithm allows us to make choices which may lead to a more optimal algorithm with respect to both time and memory complexity. In this chapter, we show that results of ElimLin algorithm are invariant with respect to choices made during the algorithm.

In Chapter 4, we consider a new strategy for selection of samples for ElimLin algorithm. We give an optimized version of ElimLin algorithm (which can handle large number of samples more efficiently than previously available tools). Then, we demonstrate the strength of our selection strategy by breaking reduced round versions of selected ciphers with significantly lower complexity than what was previously achieved by a more sophisticated algebraic methods.

In Chapter 5, we develop a new technique in algebraic cryptanalysis which allows to further speed up the computation of ElimLin and more advanced algebraic tools. We suggest several new algorithms called Universal Proning, Mutant Proning and Iterative Proning. We show relations of these algorithms to standard tools in algebraic cryptanalysis. In real life cryptanalysis, we use heuristic versions of these algorithms which significantly improves the computational requirements. However, we need to verify the correctness of their results. Our Iterative Proning algorithm can be seen as a hybrid between two main techniques of algebraic cryptanalysis: Gröbner basis methods and SAT solvers.

Besides the research in algebraic cryptanalysis, my research included work on WEP, the cryptanalysis of ARX schemes and the design and analysis of cryptographic primitive ARMADILLO. These results are not part of this dissertation.

My publications

- ARMADILLO: a Multi-Purpose Cryptographic Primitive Dedicated to Hardware [BDN+10] presented at CHES'10.
- Fast Key Recovery Attack on ARMADILLO1 and Variants [SSV11] presented at CARDIS'11.
- Multipurpose Cryptographic Primitive ARMADILLO3 [SV13] presented at CARDIS'12.
- ElimLin Algorithm Revisited [CSSV12] presented at FSE'12 and part of Chapter 3 and in Chapter 4.
- Smashing WEP in A Passive Attack [SSVV13] presented at FSE'13.
- Tuple cryptanalysis of ARX with application to BLAKE and Skein [ALM+11] presented at ECRYPT II Hash Workshop.

- On Selection of Samples in Algebraic Attacks and a New Technique to Find Hidden Low Degree Equations [SSV14] presented at ACISP'14 and part of Chapter 4 and Chapter 5.

- Combined Algebraic and Truncated Differential Cryptanalysis on Reduced-round Simon [CMS+14] presented at SECRYPT2014 and part of Chapter 4.

2 Introduction to algebraic cryptanalysis

We now describe methods of algebraic cryptanalysis in more details. An algebraic attack can be divided into two steps: building a polynomial system and solving it. In the following paragraphs, we elaborate on both parts and relate them to our contributions in Chapter 3, Chapter 4 and Chapter 5.

2.1 Algebraic Representation of cryptographic primitive

Building an algebraic system. In algebraic cryptanalysis, we usually build a polynomial system representing a cipher by encoding an algorithm into a set of multivariate polynomials over a boolean ring. Each variable of such system represents a state-bit of the algorithm. Then, variables which correspond to state-bits from the initial round are set according to values of the plaintext and similarly, variables which correspond to state-bits from the last round are set according to the values of the ciphertext. Following this approach, we can build a polynomial system for a single or multiple plaintext-ciphertext pairs. This approach is described many times in existing literature, see for instance [BD03].

In Chapter 5, we give an alternative view on building the polynomial system using a new algorithm called Universal Proning. Afterwards, we derive a technique called Mutant Proning. This technique is designed to find so called "mutant" polynomials which are "interesting" polynomials when computing mXL. Finally, we extend Mutant Proning and suggest a new algorithm to build polynomial system called Iterative Proning, which is designed to mimic all step of mXL.

2.2 Tools for algebraic cryptanalysis

The polynomial system corresponds to a cipher and some plaintext/ciphertext pairs. We have two fundamentally different techniques to solve it: Gröbner basis based algorithms and SAT solvers. In the case of Gröbner basis, we perform arithmetic operations to transform polynomial system into another with the same set of solutions. I.e, these polynomials define an ideal and we work on finding a reduced representation of the the ideal. Typically, we find polynomials of form $k_i - \kappa_i$ where k_i is a variable representing a key bit and κ_i is 0 or 1, in addition to the equations defining the cipher. One example of reduced representation is a (well-chosen) Gröbner basis. The Gröbner basis can be computed using F4 [Fau99]/F5 [Jea02] algorithm and its alternatives such as XL, mXL, mXL2 [MMDB08] and mXL3 [MCD$^+$09]. The mXL3 algorithm was shown to be equivalent (but slower) to F4 in [ACFP11]. Additionaly, we consider ad-hoc tools for computation of Gröbner basis such as the XSL method [CP02], where we mimic XL algorithm but we try to reduce memory requirements. The analysis of XSL was given in [CL05, CYK09, LK07]. In our analysis, we focus on the ElimLin algorithm [BCN$^+$10] which is used by all methods above. Hence, understanding its limitations and improvements is crutial for further advances of more sophisticated algorithms. Alternative technique to solve polynomial system is based on SAT solvers. This can be seen as guessing a partial solution, and for each guess, we verify if it was consistent with the system. If it is not consistent, we try to learn new formulas from incorrect guesses, in order not to repeat the same incorrect guess. Both these strategies are discussed below.

2.2.1 Buchberger's algorithm

The Buchberger's algorithm [Buc06] is a method to transform a set of polynomials into a list of polynomials generating the same ideal, and such that it is ordered according to some monomial ordering, i.e, a Gröbner basis. The algorithm can be seen as a generalization of the Euclid algorithm and the Gauss elimination. The algorithm takes as an input a set of polynomials F over a polynomial ring R and it outputs G such that they span the same ideal. Furthermore, the list of polynomials G is ordered according to prescribed ordering.

Definition 1 (Monomial ordering)**.** *Monomial ordering on* $\mathbf{F}_2[V]$ *is a relation* $\prec$ *on* $\mathbb{Z}_+^{|V|}$ *that satisfies:*

1. *The relation* $\prec$ *is a total ordering.*
2. *If* $\alpha \prec \beta$ *and* $\beta \in \mathbb{Z}_+^{|V|}$ *then* $\alpha + \gamma \prec \beta + \gamma$
3. *The relation* $\prec$ *is a well-ordering, i.e, every nonempty subset of* $\mathbb{Z}_+^{|V|}$ *has a smallest element.*

We have a natural bijection $(a_1, \dots, a_n) \longleftrightarrow x_1^{a_1} \dots x_n^{a_n}$. Hence, $\prec$ is actually a relation on $\mathbf{F}_2^{|V|}$.
For $x \in V$, $a, b \in \mathbb{N}$, we recall some typical monomial orderings:

Lexicographic (lex): $x^a \prec x^b \Leftrightarrow \exists\, 0 \leq i \prec n : a_0 = b_0, \dots, a_{i-1} = b_{i-1}, a_i < b_i$

Degree reverse lexicographic (degrevlex): Let $deg(x^a) = a_0 + \cdots + a_{n-1}$, then

$$\begin{aligned} x^a \prec x^b \Longleftrightarrow & deg(x^a) < deg(x^b) \text{ or} \\ & deg(x^a) = deg(x^b) \text{ and } \exists i\ 0 \leq i < n \text{ such that} \\ & a_{n-1} = b_{n-1}, \dots, a_{i+1} = b_{i+1}, a_i > b_i \end{aligned}$$

Additionally, we combine this monomial ordering into a product ordering. For $x = (x_0, \dots, x_{n-1})$ and $y = (y_0, \dots, y_{m-1})$ where $\prec_1$ and $\prec_2$ are monomial orderings, we define product ordering $(\prec_1, \prec_2)$ which we now denote $\prec$. We say $x^a y^b \prec x^A y^B \Leftrightarrow x^a \prec_1 x^A$ or $x^a = x^A$ and $y^b \prec_2 y^B$.

Algorithm 1 Buchberger's algorithm [Buc06]

```
1: $G \leftarrow F$
2: repeat
3:     select $(i, j)$ such that $(f_i, f_j) \leftarrow G \times G$ is not marked.
4:     mark $(f_i, f_j)$
5:     $g_i \leftarrow$ largest term of $f_i$ with respect to a given ordering.
6:     $g_j \leftarrow$ largest term of $f_j$ with respect to a given ordering.
7:     $a_{ij} \leftarrow$ least common multiple of $g_i$ and $g_j$.
8:     $S_{ij} \leftarrow \left(\frac{a_{ij}}{g_i} f_i\right)$
9:     for all $g \in G$ do
10:        if the largest term of $g$ appears in $S_{ij}$ with a nonzero coefficient then
11:            $S_{ij} \leftarrow S_{ij} \bmod g$ {use Euclid algorithm to compute mod "reductor" $g$}
12:        end if
13:    end for
14:    if $S_{ij} \neq 0$ then
15:        $G \leftarrow G \cup \{S_{ij}\}$.
16:    end if
17: until all $G \times G$ elements are marked
18: Output $G$
```

The Buchberger algorithm can follow different strategies for selection of critical pairs and the selection of reductors (Step 3 and 9). Independently of these selections, the Buchberger algorithm gives a correct result. However, these choices are important for the total running time. The Buchberger algorithm is very inefficient as it spends 90% of time in step 11 by computing reductions to the zero. These are coming from relations $f_i f_j = f_j f_i$ and, in case of polynomial system containing field equations $v^2 = v$ for $v \in V$ (which is our case as well), from relations $f_i^2 = f_i$. Hence, a good implementation of Buchberger algorithm should avoid these polynomials as they bring no new information about the ideal. The state of the art algorithm for computing a Gröbner basis is F4/F5 which is described later in Section 2.2.3.

2.2.2 Macaulay matrix

Definition 2 (Definition 2.3 in [BDM14]). *Given a set of polynomials $F = \{f_1, \dots, f_s\} \subseteq \mathbb{B}$, each of degree d_i. We consider the set B of all monomials of degree up to d of $\mathbb{B}[V]$. Then, the Macaulay matrix of degree d, which we denote as $\mathsf{Mac}_F(d)$, is the matrix of elements from $\mathbf{F}_2$ with $|B|$ columns in which the i-th row is the list of coefficients a_{ij} of the polynomial $p_i = \sum_j a_{ij} b_j$ where b_j is j-th element of B (i.e, j-th monomial) and p_i is a product of one element of B and one element of F. We write*

$$\mathsf{Mac}_F(d) = \begin{pmatrix} f_1 \\ x_1 f_1 \\ \vdots \\ x_n^{d-d_1} f_1 \\ f_2 \\ x_1 f_2 \\ \vdots \\ x_n^{d-d_s} f_s \end{pmatrix}$$

where each polynomial f_i is multiplied with all monomials from degree 0 up to $d - d_i$ for all $i = 1, \dots, s$. In what follows, by abuse of notation we also write

$$\mathsf{Mac}_F(d) = \left\{ f_1, x_1 f_1, \dots, x_n^{d-d_1} f_1, f_2, x_1 f_2, \dots, x_n^{d-d_s} f_s \right\}.$$

2.2.3 F4/F5

The F4 algorithm [Fau99] allows to significantly decrease the number of reductions to zero using simple criteria. However, F4 still keeps many reductions to zero. In [JV11], the authors gave a variant of F4 for algebraic cryptanalysis which avoids all reductions to zero by using precomputation. The extension F5 [Jea02] computes the Gröbner basis incrementally, i.e, the Gröbner basis of ideal $\langle F_i \rangle = \langle f_1, \dots, f_i \rangle$ is computed using Gröbner basis G_{i-1} of $\langle F_{i-1} \rangle$. In [Ste06], the authors introduce a variant which uses B_{i-1} to reduce generators of $\langle F_i \rangle$. In [EP10], the author further replaces the Gröbner basis

G_{i-1} by a reduced Gröbner basis B_{i-1} to reduce the total number of reductions performed by F5. In the case when $\langle F_i \rangle = \langle f_1, \ldots, f_i \rangle$ and $(f_1, \ldots, f_i)$ is a so-called "regular sequence", the F5 algorithm was shown [Jea02] to perform no reduction to zero and hence, it is a very efficient generic method to compute a Gröbner basis. F4/F5 computes the Gröbner basis in degrevlex ordering and afterwards, we apply FGLM [FM13] algorithm to change the ordering as to the prescribed ordering.

2.2.4 XL and its mutations

The XL and mXL family can be seen as alternatives to the F4 algorithm. In what follows, we formulate the XL, mXL, mXL2 and mXL3 algorithms.

Definition 3 (level of polynomial, Definition 1 in [MMDB08])**.** *Let $g \in \langle \mathcal{S} \rangle$. We express*

$$g = \sum_{p \in \mathcal{S}} g_p p$$

where $g_p \in \mathbf{F}_2[V]$, $p \in \mathcal{S}$. The level of this representation of g is defined to be

$$\mathit{level}(g) = \max\{\deg g_p p : p \in \mathcal{S}, g = \sum_{p \in \mathcal{S}} g_p p\}$$

The level of polynomial $g \in \langle \mathcal{S} \rangle$ is minimum level among its representations and we denote it level(g) (or level$_{\mathcal{S}}(g)$ if the system $\mathcal{S}$ is not clear from the context).

Definition 4 (mutant, Definition 2 in [MMDB08])**.**
Let $\mathcal{S} \subset R$. Then $g \in \langle \mathcal{S} \rangle$ is called a mutant with respect to set $\mathcal{S}$ if its degree is smaller than its level.

Sometimes, when we find a mutant polynomial (which is also called a fall polynomial), we learn "a new information" about the ideal. Later in Chapter 5, we introduce universal and nonuniversal polynomials. The mutants which reveal new information about our ideal are nonuniversal. Finally, in Iterative Proning, we give a method to build a polynomial system from nonuniversal mutants, and the aim is to obtain a so much overdefined system that it would be easy to find a solution.

XL. The XL algorithm was introduced in [CKPS00] as a new tool for solving overdefined systems of multivariate polynomial equations. In XL, we consider $F \subset \mathbf{F}_2[V]$ and for $D \in \mathbb{N}$ the XL$_D$ algorithm builds the Macaulay matrix Mac$_F(D)$ and runs the Gauss elimination. The XL algorithm computes the Gröbner basis in a similar way as F4, but it performs additional unnecessary computations [AFI$^+$04].

mXL. The mXL_D algorithms takes as an input a polynomial system F and it returns another polynomial system F' which has the same solution. We give a formal description in Algorithm 2. Roughly speaking, it builds for some increasing $d \in [1,D]$ polynomial systems F_d such that $F_d = \mathsf{mXL}_d(F_d)$. Each F_d is initialized as polynomials in the linear span of F of polynomials of degree at most d. Then, we add to F_i all polynomials of degree i of Macaulay matrix $\mathsf{Mac}_{F_d}(d)$ and we continue with the computation of $\mathsf{Mac}_{F_i}(i)$ for i minimal such that F_i was changed. These new polynomials are called mutants (see Definition 4). In the computation of mXL, we initially need to increase the degree significantly. Then, we start to discover new mutants and hence, we will work again on systems F_d for a smaller d. If the system has a unique solution, we find it eventually in F_1. The behavioral of this degree d was analysed in [YCY13].

Comparison XL/mXL/mXL2/mXL3. The advantage of mXL over XL is very significant if we recover only a few mutants at each step of computation. The advantage of mXL over XL is reduced when the system generates large number of mutants. This leads to additional improvements mXL2 [MMDB08], mXL3 [MCD$^+$09] and Mutant Based Gröbner basis [BCDM10]. In these improvements, we limit the number of mutants which we consider in the following iterations whenever we recover too many mutants. The mXL was analysed in [TW10, MDB11]. It was used for instance in the cryptanalysis of MQQ, see [MDB08]. The mXL3 was shown to be equivalent to F4 [ACFP11] for a so called "normal selection strategy". However, it does not prevent as many reductions to zero as F4. The notion of mutant polynomials corresponds to fall polynomials in F4 which are also prefered by the "normal selection strategy". In Iterative Proning which we introduce in Chapter 5, we also recover mutant polynomials and similarly to mXL2, we consider only few mutants per iteration for efficiency.

2.2.5 ElimLin

ElimLin is a basic tool for algrebraic cryptanalysis and it is used (directly or indirectly) by all other Gröbner basis computation tools. In F4/mXL, it is hidden in the first iterations of the algorithm. It performs Gauss eliminations and substitutions by linear terms. Hence, the degree of the polynomial system never increases. However, such technique does not guarantee to find a solution. We describe ElimLin in more details in Chapter 3, and we investigate the properties of ElimLin and choices of plaintext/ciphertext pairs for building the polynomial system.

2.2.6 Gauss Elimination

Most algebraic tools rely on Gauss elimination as a tool for simplification of a polynomial system. Therefore, an efficient implementation of Gauss elimination plays vital

Algorithm 2 mXL [MDB11]

```
Input: $F \subseteq \mathbf{F}_2[V], D \in \mathbb{N}$
Output: $F_1 \subseteq \mathbf{F}_2[V]$ such that $\langle F_1 \rangle = \langle F \rangle$ and $\deg(F_1) \leq 1$.
1: for $d < D$ do
2:   $F_d \leftarrow \int \textbf{linspan}(F) \Big(^{d}$ {i.e, polynomials bounded by degree $d$, see Notation 17}
3: end for
4: $d \leftarrow 1$
5: repeat
6:   $D \leftarrow d+1$
7:   for $i = d-1$ to $1$ do
8:     $M^i \leftarrow \int \textbf{linspan}(\mathsf{Mac}_{F_d}(d)) \Big(^{i}$
9:     if $M^i \not\subseteq F_i$ then
10:      $F_i \leftarrow F_i \cup M^i$ {i.e, add mutants}
11:      $D \leftarrow i$ {i.e, continue with smallest degree where mutants were found}
12:    end if
13:  end for
14:  $d \leftarrow D$
15: until $\dim(F_1) = |V|$
```

role in algebraic cryptanalysis. The efficiency can be improved if we use additional properties of our polynomial system. For instance, in the case of algebraic cryptanalysis, we work over a finite field and in many cases, we work over $\mathbf{F}_2$. This simplifies the Gauss elimination algorithm and allows for an additional speedup. Moreover, when we choose the best strategy for Gauss elimination, we need to consider the sparsity of our system. Gauss elimination on dense systems was investigated in [AVBP11, ABH10] and a very efficient implementation can be found in [ABP11]. Details about Gauss elimination for sparse systems can be found in [Vil97, Cop93, Kal93]. The sparse Gauss elimination is beneficial when computing sparse Gröbner basis such as in [FSS14]. However, the mXL implementation usually uses M4RI library [ABP11] for dense matrices. In our implementation of ElimLin we also use M4RI, i.e, the dense representation. In Universal Proning, we compute a nullspace of a matrix that should look random and hence, a dense representation is beneficial.

2.2.7 SAT

SAT solvers have been successfull in the cryptanalysis of various schemes. In [CBW07], the authors gave a practical attack on KeeLoq. In [MZ06], the authors evaluated SAT solvers on MD4 and MD5. In [KY10], the authors considered SAT solvers to recover the secret key of AES from decayed memory image after cold restart. In [EDC09], the authors compare Gröbner basis based attacks and SAT solvers based attack on SMS4. In

[BB11], the authors were able to break 8 rounds of PRINTCipher-48 using SAT solvers and break the full PRINTCipher-48 assuming side channel leakage of Hamming weight.

2.2.8 Zero-dimensional ideals

The complexity of Gröbner basis computation can be doubly exponential. However, in algebraic cryptanalysis, the Gröbner basis computation requires “only” exponential time, as we work with so called “zero-dimensional ideal”. When we work over algebraically closed field, an ideal is zero-dimensional if the associated variety is finite. However, in the case of finite fields, we need to be more carefull as we always have a finite number of solutions. We give a definition of zero-dimensional ideal in Definition 5 and its characterization in Lemma 6.

Definition 5. *Let F be a field and V a set of variables. An ideal $I \subseteq F[V]$ is zero dimensional if and only if the F-vector space dimension of $F[V]/I$ is finite, i.e,* $\dim_F F[V]/I < \infty$.

Lemma 6 ([DF04] 26, page 705)**.** *Let $I \subseteq F[V]$ be an ideal. The following three statements are equivalent:*

1. *I is zero-dimensional.*
2. *The variety $\mathcal{V}_K(I)$ is a finite set for every field K such that $F[V] \subseteq K$.*
3. *$I \cap \mathbf{F}_2[v]$ is a finite set for every $v \in V$.*

2.3 Hybrid algorithms

The relation between algebraic methods have been extensively studied. ElimLin is a basic algorithm which is part of almost every algebraic tool. XSL is an ad-hoc method for optimization of XL which was analyzed in [CL05]. The comparison between Gröbner basis computation and SAT solver was done in [EDC09]. The complexity of SAT was further studied in [LV99]. The asymptotic estimates of the complexity of XL and Gröbner basis were given in [YCC04].

SAT + Gauss elimination. The cryptominisat [Soo10] accepts both OR clauses and XOR clauses as an input and it performs Gauss elimination on XOR clauses before the DPLL procedure takes place. In [HJ12], the authors considered Gauss-Jordan elimination. They obtained sparser XOR formulas which lead to more effective learning of new clauses, and subsequently the speedup of the SAT solver. A similar approach together with comparisons with other SAT techniques can be found in [LJN12]. A natural extension is considering a substitution after Gauss elimination. This leads to SAT + ElimLin technique.

SAT + ElimLin. ElimLin can find hidden linear relations of the polynomial system efficiently. Hence, the SAT solver avoids learning clauses equivalent to these linear relations which reduces the running time of a SAT solver. This technique is used by the publicly available tool for algebraic cryptoanalysis from Courtois [Cou10].

SAT + mXL. The technique above (SAT + ElimLin) can be further extended by replacing ElimLin with a degree bounded mXL (i.e, to SAT + mXL_D). However, this strategy results in a dense polynomial system which is usually difficult to solve for a SAT solver. We now discuss an opposite strategy, i.e, we use SAT solver to "learn" new clauses and we use these in mXL computation. This strategy is a long-standing proposal in algebraic cryptanalysis, but we are not aware of any efficient implementation of this approach. In Chapter 5, we develop a technique called Universal Proning. In Universal Proning, we learn polynomials by computation of nullspace and hence, our Universal Proning can be seen as a substitute for a SAT solver in the above scenario. Furthermore, we give an extension called Iterative Proning, which iterarively reduces the keyspace by learning new polynomials by the computation of nullspace. Hence, the technique developed in Chapter 5 can be seen as a hybrid algorithm mixing the SAT solving and Gröbner basis basis techniques.

2.4 Algrebraic Cryptanalysis

In algebraic cryptanalysis, the polynomial system has additional properties which may be used to speed-up an algebraic attack.

Non random structure. Many cryptographic algorithms are based on iteration of a simple subroutine. This reflects into the structure of the corresponding polynomial system. Hence, unlike an instance of MQ problem, an instance of a problem from algebraic cryptanalysis is not a random problem of MQ and the generic complexity of MQ problem is only an upper bound. Therefore, we should try to use the structure of the polynomial system and develop a dedicated algorithm for solving such polynomial systems. The idea of taking a structure into a consideration has already appeared in the literature, for instance [FSS14]. To use such structure efficiently, we want to select samples in such a way that the structure leads to a system which can be highly simplified. Then in Chapter 5, we look for such simplification without solving the polynomial system.

Freedom of choice. Unlike in the MQ problem, we are allowed to choose plaintext/ciphertext pairs and hence, we tweak the polynomial system used for the algebraic attack. This leads us to another algebraic attack called cube attack, which is specialized in selecting plaintexts in such a way that finding the secret key is especially easy. We show in Chapter 3 that a carefull selection of samples is beneficial even in the case of ElimLin, which is the common ground for all algebraic tools.

Kerkhoff's principle. Due to Kerkhoff's principle, we can efficiently build a polynomial system representing a cipher. Moreover, we can perform the encryption/decryption algorithm for a known key. This allows us to adapt the polynomial system by selecting plaintext/ciphertext pairs which lead to attacks with a smaller computational requirements. We use this in Chapter 4 for selection of samples, and in Chapter 5 to explore a hidden structure of our polynomial system.

2.5 Definition of a Polynomial System

Definition 7 (Boolean polynomial). *Let V be a set of variables. Let $b \in \mathbf{F}_2[V]$ be a polynomial such that*

$$b = \sum_{W \subset V} a_W \prod_{w \in W} w^{ew}$$

where $a_W \in \mathbf{F}_2$. Then b is called a boolean polynomial iff for all $w \in W$ we have $e_w \in \{0,1\}$. We denote $\mathbb{B}[V]$ the set of all boolean polynomials of ring $\mathbf{F}_2[V]$.

Definition 8. *Given a set of variables W, we denote*

$$\mathit{FieldEq}[W] = \left\langle v^2 - v : v \in W \right\rangle_{\mathbf{F}_2[W]}$$

The ideal $\mathsf{FieldEq}[V]$ is an ideal of trivial relations which exist due to computation in function field.

Notation 9. *We use kln to represent the key length. We use mln to represent the message length and the length of the state vector. We use smpn to represent the number of plaintext/ciphertext pairs (samples). We use rndn to represent the number of rounds of the cipher.*

We represent state bits and key bits by variables as in Notation 10.

Notation 10. *Each state variable $\mathsf{s}_{p,r}^j$ corresponds to a sample of index p, a round r, and an index j in the state vector. The key is represented by key variables $k_1, \ldots, k_{\mathit{kln}}$. The plaintext p is represented by $\mathsf{s}_{p,0}^j$ and the ciphertext is represented by $\mathsf{s}_{p,\mathit{rndn}}^j$ and round keys at round r are represented by $k_1^r, \ldots, k_{\mathit{kln}}^r$.*

Notation 11. *We denote the set of variables as*

$$
\begin{aligned}
V_K &= \bigcup_{t\in[1,\mathit{kln}]} \{k_t\} \\
V_S &= \bigcup_{\substack{p\in[1,\mathit{smpn}]\\ r\in[0,\mathit{rndn}]\\ j\in[1,\mathit{mln}]}} \left\{\mathsf{s}^j_{p,r}\right\} \\
\overline{K} &= \bigcup_{\substack{r\in[0,\mathit{rndn}]\\ j\in[1,\mathit{mln}]}} \{k^r_1,\dots,k^r_{\mathit{kln}}\} \\
\mathit{PT} &= \bigcup_{\substack{p\in[1,\mathit{smpn}]\\ j\in[1,\mathit{mln}]}} \left\{\mathsf{s}^j_{p,0}\right\} \\
\mathit{CT} &= \bigcup_{\substack{p\in[1,\mathit{smpn}]\\ j\in[1,\mathit{mln}]}} \left\{\mathsf{s}^j_{p,\mathit{rndn}}\right\} \\
V &= V_K \cup \overline{K} \cup V_S
\end{aligned}
$$

The round function of the cipher is represented by a set of polynomials r^j_r which take as input all state variables at round r and return the j-th state variable at round $r+1$, i.e, $\mathsf{s}^j_{p,r+1}$ is given by polynomial $\mathsf{r}^j_r(\mathsf{s}^1_{p,r},\dots,\mathsf{s}^{\mathsf{mln}}_{p,r},k^r_1,\dots,k^r_{\mathsf{kln}})$. We denote the corresponding equation [1]

$$\mathsf{Eq}^p_{j,r} = \mathsf{r}^j_r\left(\mathsf{s}^1_{p,r},\dots,\mathsf{s}^{\mathsf{mln}}_{p,r},k^r_1,\dots,k^r_{\mathsf{kln}}\right) - \mathsf{s}^j_{p,r+1}$$

where $k^r_j = \mathsf{rk}^r_j\left(k_1,\dots,k_{\mathsf{kln}}\right)$.

$$\mathsf{RK}_{j,r} = \mathsf{rk}^r_j\left(k_1,\dots,k_{\mathsf{kln}}\right) - k^r_j$$

Notation 12 (system)**.** *We denote*

$$\mathcal{S} = \mathit{FieldEq}[V] \cup \bigcup_{\substack{p\in[1,\mathit{smpn}]\\ r\in[0,\mathit{rndn}]\\ j\in[1,\mathit{mln}]}} \left\{\mathit{Eq}^p_{j,r},\mathit{RK}_{j,r}\right\}$$

The equations are taken over ring[2] $\mathbf{F}_2[V]$*, i.e,* $\mathcal{S}\subseteq\mathbf{F}_2[V]$*, and they represent relations*

[1]we use "equation" and "polynomial" as synonyms. Solving an equation means finding roots of a polynomial.

[2]alternatively, we can consider boolean ring and avoid having $\mathsf{FieldEq}[V]$ in polynomial system $\mathcal{S}$

between variables of round r and $r+1$. We further denote

$$\begin{aligned}
\mathcal{S}_{\chi,\star,\star} &= \mathcal{S} \cup \bigcup_{\substack{p\in[1,\mathsf{smpn}]\\ j\in[1,\mathsf{mln}]}} \left(s^j_{p,0} - \chi^j_p\right)\\
\mathcal{S}_{\star,\gamma,\star} &= \mathcal{S} \cup \bigcup_{\substack{p\in[1,\mathsf{smpn}]\\ j\in[1,\mathsf{mln}]}} \left(s^j_{p,\mathsf{rndn}} - \gamma^j_p\right)\\
\mathcal{S}_{\star,\star,\kappa} &= \mathcal{S} \cup \bigcup_{i\in[1,\mathsf{kln}]} \{k_i - \kappa_i\}
\end{aligned}$$

For a system $\mathcal{S}$, we denote:

$$\begin{aligned}
\mathcal{S}_{\chi,\star,\kappa} &= \mathcal{S}_{\chi,\star,\star} + \mathcal{S}_{\star,\star,\kappa}\\
\mathcal{S}_{\star,\gamma,\kappa} &= \mathcal{S}_{\star,\gamma,\star} + \mathcal{S}_{\star,\star,\kappa}\\
\mathcal{S}_{\chi,\gamma,\star} &= \mathcal{S}_{\chi,\star,\star} + \mathcal{S}_{\star,\gamma,\star}\\
\mathcal{S}_{\chi,\gamma,\kappa} &= \mathcal{S}_{\chi,\star,\star} + \mathcal{S}_{\star,\gamma,\star} + \mathcal{S}_{\star,\star,\kappa}
\end{aligned}$$

We say that $\mathcal{S}_{\chi,\star,\star}$ and $\mathcal{S}_{\star,\gamma,\star}$ are open-ended. We use notation $\mathcal{S}_{\chi,\gamma,\kappa}$ to denote that we set plaintext to χ, ciphertext to γ and key to κ. The symbol $\star$ at any position means that the value is unset a priori. Hence, $\mathcal{S}_{\chi,\star,\star}$ is the system of equations when we fix the plaintexts to χ and $\mathcal{S}_{\star,\gamma,\star}$ is the system when we fix the ciphertexts to γ. We later use $\mathcal{S}_{\chi,\gamma,\star}$ which thus represents the system in which we fix both the plaintext and the ciphertext.

Definition 13. *We define a ring homomorphism $\mathsf{Eval}_{\chi,\gamma,\star} : \mathbf{F}_2[V] \to \mathbf{F}_2[V]$ which assigns variables PT, CT to an element of $\mathbf{F}_2$ to plaintext and ciphertext variables.*

$$\mathsf{Eval}_{\chi,\gamma,\star}(V) = \begin{cases} v \to \chi^j_p & \textit{if } v = s^j_{p,0}\\ v \to \gamma^j_p & \textit{if } v = s^j_{p,\mathsf{rndn}}\\ v \to v & \textit{otherwise.}\end{cases}$$

Actually, we have $\left\langle \mathsf{Eval}_{\chi,\gamma,\star}(\mathcal{S})\right\rangle = \left\langle \mathcal{S}_{\chi,\gamma,\star}\right\rangle \cap \mathbb{B}[V \setminus (\mathsf{PT} \cup \mathsf{CT})]$.

Observation 14. *The ideals $\left\langle \mathcal{S}_{\star,\gamma,\kappa}\right\rangle$ resp. $\left\langle \mathcal{S}_{\chi,\star,\kappa}\right\rangle$ are always maximal ideals for deterministic encryption.*

Equivalently, the plaintext is uniquely determined by the key κ and the ciphertext γ resp. ciphertext is uniquely determined by the key κ and the plaintext χ. Similarly, we assume that χ and γ fully characterize the key κ:

Assumption 15. *We assume that the ideal $\left\langle \mathcal{S}_{\chi,\gamma,\star}\right\rangle$ is a maximal ideal.*

Lemma 16. *An ideal $I \subseteq \mathbf{F}_2[V]$ such that $\mathsf{FieldEq}[V] \subseteq I$ is maximal, if and only if, either v or $v+1$ is in I but not both for each $v \in V$.*

Proof. *We show that if $\mathsf{FieldEq}[V] \subseteq I$ and $I + \langle v \rangle = I + \langle v+1 \rangle = \mathbf{F}_2[V]$ then $I = \mathbf{F}_2[V]$. If $1 \in I + \langle v \rangle$, then $p = 1 + Av$ for some $p \in I$ and $A \in \mathbf{F}_2[V]$. Similarly if $1 \in I + \langle v+1 \rangle$, then there is $p' \in I$ and $B \in \mathbf{F}_2[V]$ such that $p' = 1 + B(v+1)$. We have*

- $Ap + \left(A^2 + A\right) v = A(v+1) \Longrightarrow A(v+1) \in I$
- $Bp' + \left(B^2 + B\right)(v+1) = Bv \Longrightarrow Bv \in I$
- $Bp + Ap' = B(Av+1) + A(1 + B(v+1)) = A + B + AB \Longrightarrow A + B + AB \in I$
- $B(A(v+1)) + A(Bv) = AB \Longrightarrow AB \in I$
- $p + p' + A + B + AB + AB = B \in I$
- $p' + B(v+1) = 1 \Longrightarrow 1 \in I$

So, if I is maximal, either $I + \langle v \rangle \neq \mathbf{F}_2[V]$ or $I + \langle v+1 \rangle \neq \mathbf{F}_2[V]$. As the ideal cannot be proper and larger, either $v \in I$ or $v+1 \in I$. This holds for all v. The converse is trivial. □

We recall that smpn denotes the number of plaintext/ciphertext pairs. For the assumption to be satisfied, we require that smpn is large enough to uniquely characterize κ. Essentially, the key recovery problem consists of reducing each k_i polynomial modulo $\left\langle \mathcal{S}_{\chi,\gamma,\star} \right\rangle$ to obtain κ_i. In general, reducing a polynomial modulo an ideal is hard. But there are cases where this is easy. For instance, reducing modulo $\mathsf{FieldEq}[V]$ is easy.

Notation 17. *For a set $\mathcal{S} \subseteq \mathbf{F}_2[V]$ and $D \in \mathbb{N}$ we denote a set*

$$\left\lmoustache \mathcal{S} \right\rmoustache^{D} = \{f : f \in \mathcal{S} \cap \mathbb{B}[V], \mathit{deg}(f) \leq D\}$$

Furthermore, we denote

$$\overline{\left\lmoustache \mathcal{S} \right\rmoustache}^{D} = \{f : f \in \mathcal{S} \cap \mathbb{B}[V], \mathit{deg}(f) > D\}$$

This is the case of $\left\langle \mathcal{S}_{\chi,\star,\kappa} \right\rangle$ and $\left\langle \mathcal{S}_{\star,\gamma,\kappa} \right\rangle$.

Definition 18. *Let V' be a set of variables such that $V \cap V' = V_K \cup \overline{K}$ and $|V| = |V'|$. Let us consider a bijective function $\mathsf{Dup} : V \to V'$ where $\mathsf{Dup}(k) = k$ for each $k \in V_K \cup \overline{K}$ which is homomorphically extended from $\mathbf{F}_2[V]$ to $\mathbf{F}_2[V, \mathsf{Dup}(V)]$. We further define $\mathsf{Dup}(\mathsf{Dup}(v)) = \mathsf{Dup}(v)$ for all $v \in V$.*

Notation 19. *Let $q \in \mathbf{F}_2[V, \mathit{Dup}(V)]$. Let us consider the unique polynomial $q' \in \mathbb{B}[V]$ such that $q = q' \pmod{\langle v + \mathit{Dup}(v) : v \in V \rangle_{\mathbf{F}_2[V,\mathit{Dup}(V)]}}$. We denote the polynomial q' as $[\![q]\!]_V$. Actually, we consider the mapping $[\![]\!]_V$ as a ring homomorphism $[\![]\!]_V : \mathbf{F}_2[V, \mathit{Dup}(V)] \longrightarrow \mathbf{F}_2[V]$.*

The $[\![]\!]_V$ is another case where reducing a polynomial modulo an ideal is easy.

2.6 Boolean Polynomials

Notation 20. *For $q \in \mathbf{F}_2[V]$ we define $\mathit{Var}(q) \subseteq V$ as the smallest $W \subseteq V$ such that $q \in \mathbf{F}_2[W]$.*

Notation 21. *Let V be a set of variables. For $G \subseteq \mathbf{F}_2[V]$, we denote $\langle G \rangle_{\mathbf{F}_2[V]}$ the ideal of the ring $\mathbf{F}_2[V]$ spanned by G. For $W = \bigcup_{q \in G} \mathit{Var}(q)$, we denote $\langle G \rangle_{\mathbf{F}_2[W]}$ as $\langle G \rangle$.*

Theorem 22 (Theorem 41 in [BDG$^+$09])**.** *The composition*

$$\mathbb{B}[V] \hookrightarrow \mathbf{F}_2[V] \mapsto \mathbf{F}_2[V]/\mathit{FieldEq}[V]$$

is a bijection.

Corollary 23 (technical lemma)**.** $\forall p \in \mathbf{F}_2[V_K] \;\; p \in \mathit{FieldEq}[V_K] \Longleftrightarrow \forall \kappa \in \mathbf{F}_2^{kln} p(\kappa) = 0$

Notation 24. *Due to Theorem 22, for each $q \in \mathbf{F}_2[V]$, there exists a unique $q' \in \mathbb{B}[V]$ such that $q' \equiv q \mod \mathit{FieldEq}[V]$. We denote it by $q' = q \mod \mathit{FieldEq}[V]$.*

Note that reduction modulo $\mathsf{FieldEq}[V]$ is easy.

Corollary 25 (Proposition 43 in [BDG$^+$09])**.** *Polynomials of $\mathbf{F}_2[V]$ are in the same residue class modulo $\left\langle v^2 - v : v \in V \right\rangle_{\mathbf{F}_2[V]}$ iff they generate the same function.*

Theorem 26 (Theorem 44 in [BDG$^+$09])**.** *The map μ from the set of boolean polynomials $\mathbb{B}[V_K]$ to the set of boolean functions $\mathit{Func}\left(\mathbf{F}_2^{kln}, \mathbf{F}_2\right)$ by mapping a polynomial to its polynomial function is an isomorphism of $\mathbf{F}_2$-vector-spaces.*

Corollary 27 (Corollary 45 in [BDG$^+$09])**.** *Every boolean polynomial $p \neq 1$ has a zero over $\mathbf{F}_2$. Every boolean polynomial $p \neq 0$ has a one over $\mathbf{F}_2$.*

Notation 28. *For $Q \subseteq \mathbf{F}_2[V]$, we denote* ***linspan*** $(Q) = \left\{p : p = \sum_{q \in Q} a_q q, a_q \in \mathbf{F}_2\right\}$.

Notation 29. *For $q \subseteq \mathbf{F}_2[V]$, such that*

$$q = \sum_{W \subseteq V} a_W \prod_{w \in W} w^{e_{w,W}}$$

we denote

$$\deg q = \max \left\{ \sum_{W} e_{w,W} : W \subseteq V \wedge a_W \neq 0 \right\}$$

In this thesis, we work over the ring $\mathbb{B}[V]$ in which case we consider

$$\deg q = \max \{ |W| : W \subseteq V \wedge a_W \neq 0 \}$$

3

The ElimLin algorithm

Algebraic attacks consist of building an appropriate polynomial system and solving it. In ElimLin, we consider a polynomial system such as in Section 2.5 which is easy to build. Our aim is to find samples so that such system can be solved by ElimLin. The ElimLin algorithm is rarely considered by itself as a solving method for algebraic cryptanalysis. It is commonly used as the first step before applying other algebraic techniques. Therefore, we study strategies for an improvement of ElimLin which gives a basis for advances in other algebraic techniques. Our goal is to build a polynomial system so that ElimLin performs better than in a random case. As ElimLin is a building block in other techniques, our strategies will be applicable in large varieties of algebraic attacks. In this chapter, we want to evaluate the advantage of chosen-plaintext attacks over known-plaintext attacks. We define a strategy for choosing the plaintexts based on other algebraic attack (cube attack), and we show that it significantly outperforms both known-plaintext attacks and other suggested strategies of chosen-plaintext attacks. We give attacks against LBlock, KATAN32 and SIMON to support our claims.

In Section 3.1, we describe the ElimLin algorithm and give a constructive proof of its invariance. Then in Section 3.2, we give an algebraic representation of ElimLin which will be beneficial in Chapter 4. In Section 3.3, we describe our new implementation of ElimLin.
The results of this chapter were published at FSE2012 [CSSV12].

3.1 The ElimLin Algorithm

The name ElimLin is derived from the main feature of the algorithm which is **Elim**ination of **Lin**ear equations. The ElimLin algorithm takes as an input a multivariate polynomial system over $\mathbf{F}_2$ (usually of a low degree: 2, 3 or 4). The output of ElimLin is a multivariate polynomial system which has the same set of solutions and which contains (usually) higher number of linear equations. Ideally, the ElimLin algorithm returns a linear system. In this case, we say that ElimLin "broke" the system. ElimLin was proposed in [Cou06, CB07] as a tool to evaluate the resistance of symmetric cipher. The 5-round DES was analysed using ElimLin in [CB07]; the 5-round PRESENT was analysed using ElimLin in [NSZW09] and Snow-2.0 stream cipher was analysed in [CD08]. However, the analysis of success of ElimLin is an open problem. Given a multivariate polynomial system, it is hard to predict whether ElimLin recovers more linear equations and whether it can break the system completely. Our aim is to improve the performance of ElimLin with respect to both running time and number of linear equations it recovers.

ElimLin is an iterative algorithm. In each round, we perform sequentially two operations: Gauss elimination and substitution. In each round, we modify the polynomial system as follows: we increase the number of linear equations in the system and at the same time, we decrease the number of variables appearing in non-linear equations.

Gauss Elimination: We consider a linear basis of a set of linear equations given by an intersection of the vector space spanned by all monomials of degree 1 and the vector space spanned by all equations.

Substitution: The set of linear equations found in Gauss elimination step is used for substitution. Each linear equation is used to eliminate one variable from the non-linear system. Hence, the polynomial system after the substitution contain less variables - but it may contain more non-linear terms.

Notation 30. *The set* $Q \bmod \mathit{FieldEq}[W] = \{q \bmod \mathit{FieldEq}[W], q \in Q\}$ *where* $q \bmod \mathit{FieldEq}[W]$ *is defined by Notation 24.*

Lemma 31. *Let* $p \in \mathbb{B}[V]$, $\ell \in \Big\{\mathbf{F}_2[V]\Big\}^1$, $x \in \mathit{Var}(\ell)$ *and* $W \subseteq V$ *such that* $\mathit{Var}(p) \subseteq W$ *and* $\mathit{Var}(\ell) \subseteq W$. *Then,*

$$\Big(\langle \ell \rangle_{\mathbf{F}_2[V]} + p \bmod \mathit{FieldEq}[V]\Big) \cap \mathbb{B}[W \setminus \{x\}] \tag{3.1}$$

has a single polynomial.

Proof. *We denote* $\ell = x + \tilde{\ell}$ *and we select* p_0, p_1 *such that* $p = p_0 + xp_1$ *and* $x \notin \mathit{Var}(p_0)$, $x \notin \mathit{Var}(p_1)$. *This is feasible since* $p \in \mathbb{B}[V]$ *so it has no* x^2 *inside. We further denote* $q'_0 = p_1\tilde{\ell} + p_0 \bmod \mathit{FieldEq}[V]$ *and we have* $q'_0 \in \mathbb{B}[W \setminus \{x\}]$. *So, we have* q'_0 *in the set from Eq.* (3.1). *We now show it is the only element there. Let*

$$q' \in \Big(\langle \ell \rangle_{\mathbf{F}_2[V]} + p \bmod \mathit{FieldEq}[V]\Big) \cap \mathbb{B}[W \setminus \{x\}].$$

We want to show $q' = q'_0$.
Since

$$q' \in \left(\langle \ell \rangle_{\mathbf{F}_2[V]} + p \bmod \mathit{FieldEq}[V]\right) \cap \mathbb{B}[W \setminus \{x\}]$$

we have

$$q' = \ell q + p \bmod \mathit{FieldEq}[V]$$

for some $q \in \mathbf{F}_2[V]$. We further consider $\tilde{q}$ such that $q = p_1 + \tilde{q}$.
Then, we have

$$\begin{aligned} q' &= \ell q + p \bmod \mathit{FieldEq}[V] \\ &= \left(x + \tilde{\ell}\right)(p_1 + \tilde{q}) + p_0 + x p_1 \bmod \mathit{FieldEq}[V] \\ &= p_0 + p_1 \tilde{\ell} + \tilde{q} x + \tilde{q} \tilde{\ell} \bmod \mathit{FieldEq}[V] \end{aligned}$$

We compute

$$\begin{aligned} q' - q'_0 &= \left(p_0 + x\tilde{q} + p_1 \tilde{\ell} + \tilde{q}\tilde{\ell}\right) - \left(p_1 \tilde{\ell} + p_0\right) \bmod \mathit{FieldEq}[V] \\ &= \left(x + \tilde{\ell}\right)\tilde{q} \bmod \mathit{FieldEq}[V] \in \mathbb{B}[V \setminus \{x\}] \end{aligned}$$

We consider for $i \in \mathbb{N}$ polynomials $\tilde{q}_i \in \mathbf{F}_2[V \setminus \{x\}]$ such that $\tilde{q} = \sum_i \tilde{q}_i x^i$ and we denote

$$\bar{q} = \tilde{q}_0 + x \sum_{i \geq 1} \tilde{q}_i = \tilde{q}_0 + x \bar{q}_1 .$$

Then,

$$q' - q'_0 \equiv \left(x + \tilde{\ell}\right)(\tilde{q}_0 + x\bar{q}_1) \equiv \tilde{\ell}\tilde{q}_0 + x\left(\tilde{q}_0 + \tilde{\ell}\bar{q}_1 + \bar{q}_1\right) \pmod{\mathit{FieldEq}[V]}.$$

Since $q' - q'_0 \in \mathbb{B}[V \setminus \{x\}]$ and $\tilde{\ell}\tilde{q}_0$ has no x inside, we have $0 = \tilde{q}_0 + \tilde{\ell}\bar{q}_1 + \bar{q}_1 \bmod \mathit{FieldEq}[V]$. Therefore, we also have
$\tilde{q}_0 \equiv \left(\tilde{\ell} + 1\right)\bar{q}_1 \pmod{\mathit{FieldEq}[V]}$ which implies
$\tilde{\ell}\tilde{q}_0 \equiv \tilde{\ell}\left(\tilde{\ell} + 1\right)\bar{q}_1 \pmod{\mathit{FieldEq}[V]}$ and finally, we obtain
$\tilde{\ell}\tilde{q}_0 \equiv 0 \pmod{\mathit{FieldEq}[V]}$.
Therefore,

$$q' - q'_0 = \tilde{\ell}\tilde{q}_0 \bmod \mathit{FieldEq}[V] = 0$$

Hence, $q' = q'_0$ which is uniquelly defined by p and ℓ.

$\square$

Definition 32 (substitution). *Let $Q_T \subseteq \mathbf{F}_2[x_1, x_2, \ldots, x_n]$, $\ell \in \left\{\mathbf{F}_2[\mathit{Var}(Q_T)]\right\}^1$ such that*

$x_1 \in \mathit{Var}(\ell)$. Then,

$$Q'_T = \Big(\langle \ell \rangle_{\mathbf{F}_2[\mathit{Var}(Q_T)]} + Q_T \bmod \mathit{FieldEq}[\mathit{Var}(Q_T)]\Big) \cap \mathbb{B}[\mathit{Var}(Q_T) \setminus \{x_1\}]$$

is the polynomial system where we substitute x_1 by $x_1 + \ell$.

Lemma 33. *For all V such that $\mathit{Var}(Q_T) \subseteq W \subseteq V$ and $x \in \mathit{Var}(\ell) \subseteq W$*

$$Q'_T = \Big(\langle \ell \rangle_{\mathbf{F}_2[V]} + Q_T \bmod \mathit{FieldEq}[V]\Big) \cap \mathbb{B}[W \setminus \{x\}]$$

Proof. *We start from Lemma 31 and for every $q \in Q_T$, we show there exists a unique $q' \in \Big(\langle \ell \rangle_{\mathbf{F}_2[V]} + p \bmod \mathit{FieldEq}[V]\Big) \cap \mathbb{B}[W \setminus \{x\}]$. Sp, it matches the polynomial we obtain by setting $V = W = \mathit{Var}(Q_T)$* □

Essentially, to eliminate x in $q \in Q_T$ by using ℓ, we add to q a multiple of ℓ and reduce it modulo $\mathsf{FieldEq}[x_1, x_2, \ldots, x_n]$ such that x_i no longer appear. ElimLin repeats the steps above until no new linear equation is found. The precise definition of the algorithm is given in Algorithm 3.

The running time of ElimLin depends on the choices made in substitution steps 6, 13, and 15. We now give an intuition why this is the case. We consider two sets A and B of linearly independent linear equations which span the same vector space. When we use linear equations from the set A for substitution, we obtain a large amount of non-linear terms. When we use linear equations from the set B for substitution, we obtain a dense polynomial system with small amount of non-linear terms. We give such example later in Section 3.2.3. The first case is usually beneficial if our implementation of Gauss elimination is optimized for a sparse polynomial system. The second case is usually beneficial if our implementation of Gauss elimination is optimized for a dense polynomial system. However in the case A, the substitution step usually takes more time. Hence to optimize the running time of ElimLin, we would like to select the sets such that the total running time is minimal.

In what follows, we show that the span of the resulting Q_L is invariant with respect to choices made in steps 6, 13, and 15.

Example of the ElimLin computation We now consider a multivariate polynomial system over $\mathbb{B}[x_1 \ldots, x_6]$.

Algorithm 3 ElimLin algorithm. Algorithm 10.1 in [Sep12]

```
Input: $Q^0 \subseteq \mathbb{B}[V]$.
Output: $Q_T, Q_L \subseteq \mathbb{B}[V]$ such that $\langle Q^0 \rangle = \langle Q_T, Q_L \rangle$.
 1: Set $Q_L \leftarrow \emptyset$ and $Q_T \leftarrow Q^0$ and $k \leftarrow 1$.
 2: repeat
 3:   Compute linear span of $Q_T$.
 4:   Set $Q_{L'} \leftarrow \left\{ \mathbf{linspan}(Q_T) \right\}^1$.
 5:   Set flag.
 6:   for all $\ell \in Q_{L'}$ do
 7:     if $\deg \ell < 1$ then
 8:       if $\ell \neq 0$ then
 9:         Output $(\emptyset, \{1\})$.
10:       end if
11:     else
12:       Unset flag.
13:       Let $x_{t_k} \in \mathsf{Var}(\ell)$.
14:       $Q_T \leftarrow \left(\langle \ell \rangle_{\mathbf{F}_2[\mathsf{Var}(Q_T)]} + Q_T \text{ mod } \mathsf{FieldEq}[\mathsf{Var}(Q_T)]\right) \cap \mathbb{B}[\mathsf{Var}(Q_T) \setminus x_{t_k}]$
15:       $Q'_L \leftarrow \left(\langle \ell \rangle_{\mathbf{F}_2[\mathsf{Var}(Q'_L)]} + Q'_L \text{ mod } \mathsf{FieldEq}[\mathsf{Var}(Q'_L)]\right) \cap \mathbb{B}[\mathsf{Var}(Q'_L) \setminus x_{t_k}]$
16:       $Q_L \leftarrow Q_L \cup \{\ell\}$.
17:       $k \leftarrow k + 1$
18:     end if
19:   end for
20: until flag is set.
21: Output $(Q_T, Q_L)$.
```

$$\begin{cases} x_4x_6 + x_5x_6 \\ x_2x_6 + x_3x_6 \\ x_2 + x_3 + x_5 + x_6 + x_3x_4 \\ x_1 + x_3 + x_4 \\ x_1x_3 + x_1x_4 + 1 \\ x_2x_3 + x_2x_5 + x_1x_6 \\ x_3x_6 + x_3 + 1 \end{cases}$$

We consider linear equation $x_1 + x_3 + x_4$ for a substitution $x_3 = x_1 + x_4$ and we obtain

$$\begin{cases} x_4x_6 + x_5x_6 & \\ x_2x_6 + (x_1 + x_4)\,x_6 & \Rightarrow x_2x_6 + x_1x_6 + x_4x_6 \\ x_2 + (x_1 + x_4) + x_5 + x_6 + (x_1 + x_4)\,x_4 & \Rightarrow x_2 + x_1 + x_4 + x_5 + x_6 + x_1x_4 + x_4x_4 \\ x_1 + (x_1 + x_4) + x_4 & \Rightarrow 0 \\ x_1\,(x_1 + x_4) + x_1x_4 + 1 & \Rightarrow x_1x_1 + 1 \\ x_2\,(x_1 + x_4) + x_2x_5 + x_1x_6 & \Rightarrow x_1x_2 + x_4x_2 + x_2x_5 + x_1x_6 \\ (x_1 + x_4)\,x_6 + (x_1 + x_4) + 1 & \Rightarrow x_1x_6 + x_4x_6 + x_1 + x_4 + 1 \end{cases}$$

Since $x_1^2 = x_1$, we obtain a new linear equation $x_1 = 1$.

$$\begin{cases} x_4x_6 + x_5x_6 & \\ x_2x_6 + x_1x_6 + x_4x_6 & \Rightarrow x_2x_6 + x_6 + x_4x_6 \\ x_2 + x_1 + x_4 + x_5 + x_6 + x_1x_4 + x_4x_4 & \Rightarrow x_2 + 1 + x_4 + x_5 + x_6 + x_4 + x_4x_4 \\ 0 & \\ x_1x_1 + 1 & \Rightarrow 0 \\ x_1x_2 + x_4x_2 + x_2x_5 + x_1x_6 & \Rightarrow x_2 + x_4x_2 + x_2x_5 + x_6 \\ x_1x_6 + x_4x_6 + x_1 + x_4 + 1 & \Rightarrow x_6 + x_4x_6 + 1 + x_4 + 1 \end{cases}$$

This gives us a new linear equation $x_2 + 1 + x_4 + x_5 + x_6$. We preform the substitution

$x_2 = 1 + x_4 + x_5 + x_6$.

$$\begin{cases} x_4x_6 + x_5x_6 & \\ x_2x_6 + x_6 + x_4x_6 & \Rightarrow x_5x_6 + x_6 \\ x_2 + 1 + x_4 + x_5 + x_6 + x_4 + x_4x_4 & \Rightarrow 0 \\ 0 & \\ 0 & \\ x_2 + x_4x_2 + x_2x_5 + x_6 & \Rightarrow 1 + x_4 + x_5 + x_4x_6 + x_5x_6 \\ x_6 + x_4x_6 + x_4 & \Rightarrow x_6 + x_4x_6 + x_4 \end{cases}$$

We compute the linear span of this system and we obtain a linear equation $\mathsf{Eq}_2 + \mathsf{Eq}_6 + \mathsf{Eq}_7 = (x_5x_6 + x_6) + (1 + x_4 + x_5 + x_4x_6 + x_5x_6) + (x_6 + x_4x_6 + x_4) = 1 + x_5$.

$$\begin{cases} x_4x_6 + x_5x_6 & \\ x_5x_6 + x_6 & \Rightarrow 0 \\ 0 & \\ 0 & \\ 0 & \\ 1 + x_4 + x_5 + x_4x_6 + x_5x_6 & \Rightarrow x_4 + x_4x_6 + x_6 \\ x_6 + x_4x_6 + x_4 & \Rightarrow x_4 + x_4x_6 + x_6 \end{cases}$$

We compute the linear span of this system and we obtain a linear equation $\mathsf{Eq}_1 + \mathsf{Eq}_2 + \mathsf{Eq}_7 = (x_4x_6 + x_5x_6) + (x_5x_6 + x_6) + (x_4 + x_4x_6 + x_6) = x_4$. This finally gives us $x_4 = 0$ and subsequently $x_6 = 0$.
In our example, we did not use the first equation $x_4x_6 + x_5x_6$ until the very end of ElimLin. Without this equation, ElimLin is unable to find the solution. However, the solution can be found by XL algorithm if we compute $x_6\mathsf{Eq}_7 = x_4x_6 + x_4x_6x_6 + x_6x_6 = x_6$ and after the substitution, we would obtain $x_4 = 0$.

3.2 Algebraic Representation of ElimLin

In this section, we give an algebraic representation of ElimLin. Some results of this sections are standard results from algebraic theory. However, we give a constructive proof which shed some light on the internal working of ElimLin. We use this to build an optimized implementation. Techniques developed in this section help us to characterize polynomial systems where ElimLin succeeds. We give this characterization later in Chapter 4.

3.2.1 ElimLin as an Intersection of Vector Spaces

In this section, we consider different approaches for computation of ElimLin. We build on our published paper at FSE2012 [CSSV12] and we give Corollary 40 which shows an invariance of ElimLin with respect to an ordering of substitutions. ElimLin can be formalized for any polynomial ring. However, we focus on the ring of boolean polynomials. We use the notations from Section 2.5.
We now prove that the result of ElimLin depends only on the linear span of equations which are used in the substitution. I.e, we show the result does not depend on the selection of the linear equation in Step 6 and similarly, the result does not depend on the selection of variable in Step 13.

Lemma 34. *Let $\ell_1,\dots,\ell_m \in \left\lceil \mathbf{F}_2[V] \right\rfloor^1$ with $V=\{x_{t_1},\dots,x_{t_n}\}$ and $x_{t_i}\in \mathit{Var}(\ell_i)$ where $\mathit{Var}(\ell_i)\subseteq\{x_{t_i},\dots,x_{t_n}\}$. For every $q\in\mathbf{F}_2[V]$ there exists $q'\in\mathbf{F}_2[x_{t_{m+1}},\dots,x_{t_n}]$ such that $q-q'\in\langle\ell_1,\dots,\ell_m\rangle_{\mathbf{F}_2[V]}$.*

Proof. *We have $q=\sum_{i\in[0,m]} q_i x_{t_1}^i$ for some $q_i\in\mathbf{F}_2[V\setminus\{x_{t_1}\}]$. We denote*

$$\tilde{q}=\sum_{i\in[1,m]} q_i\,(x_{t_1}+\ell_1)^i$$

and we compute $q+\tilde{q}=\sum_{i\geq 1} q_i\left(x_{t_1}^i+(x_{t_1}+\ell_1)^i\right)=\ell_1 q_1'$ for some $q_1'\in\mathbf{F}_2[V]$. Hence, $q=\tilde{q}+\ell_1 q_1'$ for some $q_1'\in\mathbf{F}_2[V]$. We iterate this process to eliminate variables $x_{t_2},\dots,x_{t_m}$ and obtain $q=q'+\ell_1q_1'+\cdots+\ell_mq_m'$ with $q_i'\in\mathbf{F}_2[V]$. □

Theorem 35. *Let $V=\{x_1,\dots,x_n\}$ and $Q_T^0\subseteq\mathbb{B}[V]$ be a vector space. Let x_{t_m} be the variable substituted in the m-th substitution of Algorithm 3. Let ℓ_m be the linear equation chosen at the m-th step of Algorithm 3. We denote Q_T^m the set Q_T after the m-th substitution. Then,*

$$Q_T^m=\left(\langle\ell_1,\dots,\ell_m\rangle_{\mathbf{F}_2[V]}+Q_T^0 \bmod \mathit{FieldEq}[V]\right)\cap\mathbb{B}[x_{t_{m+1}},\dots,x_{t_n}]$$

Proof. *We will prove by induction. In the first step of induction, we have*

$$Q_T^0=\left(\langle 0\rangle_{\mathbf{F}_2[V]}+Q_T^0 \bmod \mathit{FieldEq}[V]\right)\cap\mathbb{B}[V]$$

since $Q_T^0\subseteq\mathbb{B}[V]$. So the statement is true for $m=0$. We assume

$$Q_T^{m-1}=\left(\langle\ell_1,\dots,\ell_{m-1}\rangle_{\mathbf{F}_2[V]}+Q_T^0 \bmod \mathit{FieldEq}[V]\right)\cap\mathbb{B}[x_{t_m},\dots,x_{t_n}] \qquad (3.2)$$

and we show

$$Q_T^m = \Big(\langle \ell_1, \dots, \ell_m \rangle_{\mathbf{F}_2[V]} + Q_T^0 \bmod \textsf{\textit{FieldEq}}[V]\Big) \cap \mathbb{B}[x_{t_{m+1}}, \dots, x_{t_n}].$$

Following the substitution (Def. 32) from $Q_T^{m-1} \subseteq \mathbf{F}_2[x_{t_m}, \dots, x_{t_n}]$ in Step 14 of Algorithm 3 we obtain

$$Q_T^m = \Big(\langle \ell_m \rangle_{\mathbf{F}_2[\textsf{\textit{Var}}(Q_T^{m-1})]} + Q_T^{m-1} \bmod \textsf{\textit{FieldEq}}[\textsf{\textit{Var}}\Big(Q_T^{m-1}\Big)]\Big) \cap \mathbb{B}[\textsf{\textit{Var}}\Big(Q_T^{m-1}\Big) \setminus \{x_{t_m}\}]$$

and using Lemma 33 with $W = \{x_{t_m}, \dots, x_{t_n}\}$, we obtain

$$\begin{aligned}
Q_T^m &= \Big(\langle \ell_m \rangle_{\mathbf{F}_2[V]} + Q_T^{m-1} \bmod \textsf{\textit{FieldEq}}[V]\Big) \cap \mathbb{B}[x_{t_{m+1}}, \dots, x_{t_n}] \\
&\overset{Eq.\ 3.2}{=} \Big(\langle \ell_m \rangle_{\mathbf{F}_2[V]} \\
&\quad + \underbrace{\Big(\langle \ell_1, \dots, \ell_{m-1} \rangle_{\mathbf{F}_2[V]} + Q_T^0 \bmod \textsf{\textit{FieldEq}}[V]\Big) \cap \mathbb{B}[x_{t_m}, \dots, x_{t_n}]}_{Q_T^{m-1} \textit{ from induction}} \bmod \textsf{\textit{FieldEq}}[V]\Big) \\
&\quad \cap \mathbb{B}[x_{t_{m+1}}, \dots, x_{t_n}]
\end{aligned}$$

We denote

$$\begin{aligned}
\overline{Q_T^m} &= \Big(\langle \ell_1, \dots, \ell_m \rangle_{\mathbf{F}_2[V]} + Q_T^0 \bmod \textsf{\textit{FieldEq}}[V]\Big) \cap \mathbb{B}[x_{t_{m+1}}, \dots, x_{t_n}] \\
\widetilde{Q_T^m} &= \Big(\langle \ell_m \rangle_{\mathbf{F}_2[V]} \\
&\quad + \underbrace{\Big(\langle \ell_1, \dots, \ell_{m-1} \rangle_{\mathbf{F}_2[V]} + Q_T^0 \bmod \textsf{\textit{FieldEq}}[V]\Big) \cap \mathbb{B}[x_{t_m}, \dots, x_{t_n}]}_{Q_T^{m-1} \textit{ from induction}} \bmod \textsf{\textit{FieldEq}}[V]\Big) \\
&\quad \cap \mathbb{B}[x_{t_{m+1}}, \dots, x_{t_n}]
\end{aligned}$$

and we prove $\overline{Q_T^m} = \widetilde{Q_T^m}$.

$\widetilde{Q_T^m} \subseteq \overline{Q_T^m}$**:** *We consider $p \in \widetilde{Q_T^m}$ and we show $p \in \overline{Q_T^m}$ and we express it in terms of polynomials from ideals*

$$q_i \in \mathbf{F}_2[V]$$
$$q_0 \in Q_T^0$$

$$p = \ell_m q_m + \sum_{i \in [1, m-1]} \ell_i q_i + q_0 \bmod \textsf{\textit{FieldEq}}[V]$$

where

$$\sum_{i\in[1,m-1]} \ell_i q_i + q_0 \bmod \mathit{FieldEq}[V] \in \mathbb{B}[x_{t_m},\ldots,x_{t_n}].$$

We write this as

$$p = \sum_{i\in[1,m]} \ell_i q_i + q_0 \bmod \mathit{FieldEq}[V]$$

so $p \in \overline{Q_T^m}$ *which shows* $\widetilde{Q_T^m} \subseteq \overline{Q_T^m}$

$\overline{Q_T^m} \subseteq \widetilde{Q_T^m}$**:** *We consider* $p \in \overline{Q_T^m}$ *and we will show* $p \in \widetilde{Q_T^m}$.

For some $q_0 \in Q_T^0$ *and* $q_i \in \mathbf{F}_2[V]$, *we have*

$$p = \sum_{i\in[1,m]} q_i \ell_i + q_0.$$

Using Lemma 34, we have $q'_m \in \mathbf{F}_2[x_{t_m},\ldots,x_{t_n}]$ *such that*

$$q_m = q'_m + \sum_{i\in[1,m-1]} \ell_i q_{m,i},$$

with $q_{m,i} \in \mathbf{F}_2[V]$. *Then,*

$$p = \ell_m q'_m + \sum_{i\in[1,m-1]} \ell_i (q_i + \ell_m q_{m,i}) + q_0 \bmod \mathit{FieldEq}[V].$$

Since, $q'_m \in \mathbf{F}_2[x_{t_m},\ldots,x_{t_n}]$ *we also have*

$$\sum_{i\in[1,m-1]} \ell_i (q_i + \ell_m q_{m,i}) + q_0 \in \mathbf{F}_2[x_{t_m},\ldots,x_{t_n}].$$

Hence, $p \in \widetilde{Q_T^m}$. *Thus* $\overline{Q_T^m} \subseteq \widetilde{Q_T^m}$.

□

Lemma 36. *Let i denote the i-th iteration of the repeat loop in Algorithm 3. and let* m_i *be the number of substitutions before entering the loop. We have*

$$Q_T^{m_{i+1}} + \langle \ell_{m_i+1},\ldots,\ell_{m_{i+1}} \rangle_{\mathbf{F}_2[V]} \equiv Q_T^{m_i} + \langle \ell_{m_i+1},\ldots,\ell_{m_{i+1}} \rangle_{\mathbf{F}_2[V]} \quad (\mathrm{mod}\ \mathit{FieldEq}[V]).$$

Proof. *Using Theorem 35, we have for the set of variables* $W \subseteq V$ *which were not substituted by the "for" loop*

$$Q_T^{m_{i+1}} = \Big(\langle \ell_{m_i},\ldots,\ell_{m_{i+1}} \rangle_{\mathbf{F}_2[V]} + Q_T^{m_i} \bmod \mathit{FieldEq}[V] \Big) \cap \mathbb{B}[W]. \tag{3.3}$$

The now prove the inclusion

$$Q_{\mathcal{T}}^{m_{i+1}} + \langle \ell_{m_i}, \dots, \ell_{m_{i+1}} \rangle_{\mathbf{F}_2[V]} \subseteq Q_{\mathcal{T}}^{m_i} + \langle \ell_{m_i}, \dots, \ell_{m_{i+1}} \rangle_{\mathbf{F}_2[V]} + \mathit{FieldEq}[V]. \tag{3.4}$$

Let $q \in Q_{\mathcal{T}}^{m_{i+1}} + \langle \ell_{m_i}, \dots, \ell_{m_{i+1}} \rangle_{\mathbf{F}_2[V]}$*. Then, we have for some* $p \in Q_{\mathcal{T}}^{m_{i+1}}$ *and* $p_\ell \in \langle \ell_{m_i}, \dots, \ell_{m_{i+1}} \rangle_{\mathbf{F}_2[V]}$ *that* $q = p + p_\ell$*. Due to Eq.* (3.3)*, we have*

$$p = p' + p'_\ell \bmod \mathit{FieldEq}[V]$$

for some $p' \in Q_{\mathcal{T}}^{m_i}$ *and* $p'_\ell \in \langle \ell_{m_i}, \dots, \ell_{m_{i+1}} \rangle_{\mathbf{F}_2[V]}$*. Hence,* $q = p' + p'_\ell + p_\ell \bmod \mathit{FieldEq}[V]$ *which shows Eq.* (3.4)*.*
We now prove the opposite inclusion.

$$Q_{\mathcal{T}}^{m_i} + \langle \ell_{m_i}, \dots, \ell_{m_{i+1}} \rangle_{\mathbf{F}_2[V]} \subseteq Q_{\mathcal{T}}^{m_{i+1}} + \langle \ell_{m_i}, \dots, \ell_{m_{i+1}} \rangle_{\mathbf{F}_2[V]} + \mathit{FieldEq}[V]. \tag{3.5}$$

Let $q \in Q_{\mathcal{T}}^{m_i} + \langle \ell_{m_i}, \dots, \ell_{m_{i+1}} \rangle_{\mathbf{F}_2[V]}$*.*
Let q' *and* q_ℓ *be such that* $q' \in Q_{\mathcal{T}}^{m_i}$*,* $q_\ell \in \langle \ell_{m_i}, \dots, \ell_{m_{i+1}} \rangle_{\mathbf{F}_2[V]}$ *and* $q = q' + q_\ell$*. Due to Lemma 31, we have the set*

$$\{q''\} = \left(q' + \langle \ell_{m_i}, \dots, \ell_{m_{i+1}} \rangle_{\mathbf{F}_2[V]} \bmod \mathit{FieldEq}[V]\right) \cap \mathbb{B}[W]$$

contains a single polynomial.
Hence for some $q'_\ell \in \langle \ell_{m_i}, \dots, \ell_{m_{i+1}} \rangle_{\mathbf{F}_2[V]}$ *and* $q'_F \in \mathit{FieldEq}[V]$*, we have*

$$q'' = q' + q'_\ell + q'_F.$$

Since $q'' \in Q_{\mathcal{T}}^{m_{i+1}}$*, we have* $q = q'' + q'_\ell + q'_F + q_\ell$ *which proves the inclusion.* □

Corollary 37.

$$Q_{\mathcal{T}}^{m_{i+1}} + \langle Q_{\mathcal{L}}^{m_{i+1}} \rangle_{\mathbf{F}_2[V]} \equiv Q_{\mathcal{T}}^{0} + \langle Q_{\mathcal{L}}^{m_{i+1}} \rangle_{\mathbf{F}_2[V]} \pmod{\mathit{FieldEq}[V]}.$$

Proof. *We use induction and Lemma 36.* □

Lemma 38. *We have*

$$\textbf{\textit{linspan}}\left(Q_{\mathcal{L}}^{m_{i+1}}\right) = \int Q_{\mathcal{T}}^{m_i} + \langle Q_{\mathcal{L}}^{m_i} \rangle_{\mathbf{F}_2[V]} \bmod \mathit{FieldEq}[V] \Big\{^{1}$$

Proof. *We denote* $Q_{\mathcal{L}'}^{m_i}$ *the set of linear equations discovered in i-th iteration of the repeat loop. We have*

$$\textbf{\textit{linspan}}\left(Q_{\mathcal{L}}^{m_{i+1}}\right) = \textbf{\textit{linspan}}\left(Q_{\mathcal{L}}^{m_i}, Q_{\mathcal{L}'}^{m_i}\right)$$

We prove both inclusions:

$\subseteq$**:**

$$\textbf{\textit{linspan}}\left(Q_{\mathcal{L}}^{m_i}, Q_{\mathcal{L}'}^{m_i}\right) \subseteq \int Q_{\mathcal{T}}^{m_i} + \langle Q_{\mathcal{L}}^{m_i}\rangle_{\mathbf{F}_2[V]} \bmod \textit{FieldEq}[V] \Big)^1$$

follows from the definition of $Q_{\mathcal{L}'}^{m_i}$*.*

$\supseteq$**:**

$$\int Q_{\mathcal{T}}^{m_i} + \langle Q_{\mathcal{L}}^{m_i}\rangle_{\mathbf{F}_2[V]} \bmod \textit{FieldEq}[V] \Big)^1 \subseteq \textbf{\textit{linspan}}\left(Q_{\mathcal{L}}^{m_i}, Q_{\mathcal{L}'}^{m_i}\right)$$

Let $q \in \int Q_{\mathcal{T}}^{m_i} + \langle Q_{\mathcal{L}}^{m_i}\rangle_{\mathbf{F}_2[V]} \bmod \textit{FieldEq}[V] \Big)^1$ *and we consider* $q_0 \in Q_{\mathcal{T}}^{m_i}$*,* $q_1 \in \langle Q_{\mathcal{L}}^{m_i}\rangle_{\mathbf{F}_2[V]}$*,* $q_F \in \textit{FieldEq}[V]$ *such that* $q = q_0 + q_1 + q_F$*.*

We substitute the variables in q following Lemma 34. We denote $W = \{x_{t_m}, \ldots, x_{t_n}\}$*. Then, we have* $q + q_1' + q_F' = q'$ *with* $q_1' \in \langle Q_{\mathcal{L}}^{m_i}\rangle_{\mathbf{F}_2[V]}$*,* $q_F' \in \textit{FieldEq}[V]$ *and* $q' \in \mathbb{B}[W]$*. Since* $q' = q_0 + (q_1 + q_1') + (q_F + q_F')$ *is boolean with variables in W, we obtain* $q' \in Q_{\mathcal{T}}^{m_i}$ *due to Corollary 37 and Theorem 35.*

Since q' *is linear, either it is 0 or it is in* $Q_{\mathcal{L}'}^{m_i}$ *by definition of* $Q_{\mathcal{L}'}^{m_i}$*.*

But substituting variables in a linear polynomial q by using linear polynomials is just making linear combinations. So, $q + q'$ *is a linear combination of* $Q_{\mathcal{L}}^{m_i}$ *elements.*

Hence, $q \in \textbf{\textit{linspan}}\left(Q_{\mathcal{L}}^{m_i}, Q_{\mathcal{L}'}^{m_i}\right)$*.*

□

Algorithm 4 Alternative ElimLin algorithm.

```
Input: $Q^0 \subseteq \mathbb{B}[V]$.
Output: $\widetilde{Q_{\mathcal{T}}}, \widetilde{Q_{\mathcal{L}}} \subseteq \mathbb{B}[V]$ such that $\langle Q^0 \rangle = \langle Q_{\mathcal{T}}, Q_{\mathcal{L}} \rangle$.
 1: Set $\widetilde{Q_{\mathcal{T}}} \leftarrow$ linspan $(Q^0)$ mod FieldEq$[V]$.
 2: repeat
 3:    $\widetilde{Q_{\mathcal{L}'}} \leftarrow \int \widetilde{Q_{\mathcal{T}}} \Big)^1$
 4:    if $1 \in \widetilde{Q_{\mathcal{L}'}}$ then
 5:       Output $(\emptyset, \{1\})$.
 6:    else
 7:       $\widetilde{Q_{\mathcal{T}}} \leftarrow \left\langle \widetilde{Q_{\mathcal{L}'}} \right\rangle_{\mathbf{F}_2[V]} + \widetilde{Q_{\mathcal{T}}}$ mod FieldEq$[V]$
 8:    end if
 9: until $\widetilde{Q_{\mathcal{T}}}$ unchanged.
10: Output $\left(\widetilde{Q_{\mathcal{T}}}, \widetilde{Q_{\mathcal{L}}}\right)$.
```

So, by defining $\widetilde{\mathcal{Q}_\mathcal{T}^{m_i}} = \mathcal{Q}_\mathcal{T}^{m_i} + \langle \mathcal{Q}_\mathcal{L}^{m_i} \rangle_{\mathbf{F}_2[V]} \bmod \mathsf{FieldEq}[V]$ and $\widetilde{\mathcal{Q}_\mathcal{L}^{m_i}} = \mathbf{linspan}\left(\mathcal{Q}_\mathcal{L}^{m_i}\right)$, we can compute $\widetilde{\mathcal{Q}_\mathcal{T}^{m_i}}$ and $\widetilde{\mathcal{Q}_\mathcal{L}^{m_i}}$ in sequence using Algorithm 4.

Corollary 39. *The sets $\widetilde{\mathcal{Q}_\mathcal{T}^{m_i}}$ and $\widetilde{\mathcal{Q}_\mathcal{L}^{m_i}}$ are invariant with respect to the ordering of variables in ElimLin.*

Corollary 40. *If $(\mathcal{Q}_\mathcal{T}, \mathcal{Q}_\mathcal{L})$ is the output of ElimLin, the sets **linspan** $(\mathcal{Q}_\mathcal{L})$ and $\mathcal{Q}_\mathcal{T} + \langle \mathcal{Q}_\mathcal{L} \rangle_{\mathbf{F}_2[V]} \bmod$ FieldEq$[V]$ are invariant with respect to the ordering of variables.*

Notation 41. *Let $\mathcal{Q}$ be the initial set for ElimLin. Let $\mathcal{Q}_\mathcal{T}, \mathcal{Q}_\mathcal{L}$ be the resulting sets of ElimLin (see Algorithm 3). We denote **ELres** $(\mathcal{Q}) = \mathcal{Q}_\mathcal{T} \cup \mathcal{Q}_\mathcal{L}$.*

3.2.2 Incompleteness of ElimLin

In Section 2.2.4, we introduced the mXL algorithm. For efficiency, it often runs ElimLin in preprocessing phase. We now show that ElimLin is weaker than mXL for any degree bound. We consider a minimal degree bound for which we run mXL to be $D = \max\left\{\deg q : q \in \mathcal{S}_{\chi,\gamma,\star}\right\}$. Let us consider D different linear polynomials $\ell_i \in \left\{\mathbf{F}_2[V]\right\}^1$, where $i \in [1, D]$. Let us assume that $\mathcal{S} = \{\prod_i \ell_i + 1\}$. Then, $\mathsf{mXL}_D(\mathcal{S})$ will find $\ell_i + 1 \in \langle \mathcal{S} \rangle$ while ElimLin will stop as there exists no linear polynomial in the system.

3.2.3 Sparsity and ElimLin

We now demonstrate that the order of substitutions can significantly reduce the performance. We consider a system

$$\begin{cases} x_1 + x_2 + x_3 + x_4 \\ x_2 + x_5 + x_6 + \cdots + x_{100} \\ x_1x_2 + x_1x_3 + x_1x_4 + x_5 \end{cases}$$

We first consider substitutions $x_1 = x_2 + x_3 + x_4$ and we obtain

$$\begin{cases} x_1 + x_2 + x_3 + x_4 \\ x_2 + x_5 + x_6 + \cdots + x_{100} \\ (x_2 + x_3 + x_4)x_2 + (x_2 + x_3 + x_4)x_3 + (x_2 + x_3 + x_4)x_4 + x_5 \end{cases}$$

i.e,

$$\begin{cases} x_1 + x_2 + x_3 + x_4 \\ x_2 + x_5 + x_6 + \cdots + x_{100} \\ x_2 + x_2x_3 + x_2x_4 + x_2x_3 + x_3 + x_3x_4 + x_2x_4 + x_3x_4 + x_4 + x_5 \end{cases}$$

which is simplified to

$$\begin{cases} x_2+x_5+x_6+\cdots+x_{100} \\ x_2+x_3+x_4+x_5 \end{cases}$$

In this case, it was unnecessary to create additional monomials. However, we may also run Gauss-Jordan elimination to obtain a system

$$\begin{cases} x_1+x_3+x_4+x_5+x_6+\cdots+x_{100} \\ x_2+x_5+x_6+\cdots+x_{100} \\ x_1x_2+x_1x_3+x_1x_4+x_5 \end{cases}$$

Then, the substitution $x_1 = x_3+x_4+x_5+x_6+\cdots+x_{100}$ and $x_2+x_5+x_6+\cdots+x_{100}$ leads to the same result $x_2+x_3+x_4+x_5$ but in this case, we had to consider additional 100 new monomials. This makes the matrix representation of ElimLin much larger and it leads to significant performance limitations. Hence, it may be beneficial to postpone some substitutions to keep the polynomial system sparse. The strategy for decreasing the number of monomials has been studied in the XSL method.

3.3 Optimizing ElimLin

In this section, we give details about our implementation of ElimLin. As we showed in Section 3.2.3, the sparsity is influenced by the order of substitutions and Gauss eliminations. Finding the most sparse representation is (to our knowledge) an open problem. The sparsity influences time and memory requirements of ElimLin. For ElimLin to be successful it is necessary to consider a large polynomial system, i.e, a lot of samples. Hence, we need an method to store polynomial systems efficiently and this can be achieved if we manage to keep the system sparse. Courtois provided a publicly available implementation of ElimLin in [Cou10] which is supposed to be well optimized for sparsity. However, we observed a decrease in the speed of this ElimLin implementation for the very large systems that we considered. Hence, we take a different approach. We consider $n \in \mathbb{N}$ such that n divides smpn. We split our polynomial system $Q = \mathcal{S}_{\chi,\gamma,\star}$ into n smaller disjoint subsystems $\mathcal{T}_i$ for $i \in [1,n]$ such that $|\mathcal{T}_i| = |\mathcal{T}_j|$. Our aim is to find most of linear equations from small systems with minimal requirements and then, we use these linear equations to reduce the size of the entire system. Actually, we compute

$$Q' = \bigcup_{i,j\in[1,n]} \mathbf{ELres}\left(\mathcal{T}_i \cup \mathcal{T}_j\right) \text{ followed by } \mathbf{ELres}\left(Q'\right).$$

Due to Theorem 35, we have

$$\mathbf{ELres}\left(\bigcup_{i\in[1,n]} \mathcal{T}_i\right) = \mathbf{ELres}\left(\bigcup_{i,j\in[1,n]} \mathbf{ELres}\left(\mathcal{T}_i \cup \mathcal{T}_j\right)\right)$$

As the result of ElimLin does not depend on the order of substitutions, we suggest an optimization in Algorithm 5 for handling a large amount of samples. In our optimization, we consider parameters $m,n \in \mathbb{N}$ and we split the large system into n subsystems. Actually, we split (χ,γ) into n pairs (χ_i,γ_i) in $\mathcal{S}_{\chi,\gamma,\star}$. We keep merging m subsystems at a time by additional invocation of ElimLin as in divide-conquer strategy. However, we observed that many linear equations which arise from merging different subsystems can be found more efficiently than by a standard divide-conquer strategy. Hence, we consider all $\binom{n}{2}$ pairs among all subsystems and we run ElimLin on all these pairs to recover hidden linear equations. Then, we include these linear equations whenever we merge the subsystems. This leads to a so-called "leaves preprocessing". We present this technique in Algorithm 5. In Step 5, we compute hidden linear equations among different subsystems and we use them in a recursive call in Step 9. Each internal node of the tree in Figure 3.1 is handled by a recursive call of algorithm ElimNode. The Algorithm ElimNode takes as an input boundaries and a list of result of ElimLin applied on descendant leaves. It calls recursively ElimNode in Step 8 if the boundaries are larger than m and then, it uses standard ElimLin to merge outputs of recursive calls in Step 10. The correctness of Algorithm 5 follows directly from Theorem 35. Even though we have ElimNode$(\ell,h,L') =$ ElimNode$(\ell,h,\emptyset)$ in Step 8, we found experimentally that the computation of ElimNode(ℓ,h,L') in Step 8 is more efficient with respect to both time and memory requirements than the standard divide-and-conquer approach, i.e, computation ElimNode$(\ell,h,\emptyset)$. A progressive aggregation of ElimLin systems following the tree structure was already described [Cou06].

Algorithm 5 Divide-Conquer with leaves processing

```
ElimFast:
Input: $\mathcal{S}_{\chi,\gamma,\star} \subseteq \mathbf{F}_2[V]$, $n \in \mathbb{N}$, $m \in \mathbb{N}$.
Output: $\mathcal{Q}_{\mathcal{T}}, \mathcal{Q}_{\mathcal{L}}$
 1: select $\mathcal{S}_{\chi_i,\gamma_i,\star}$ for $i \in [1,n]$ disjoint such that $\mathcal{S}_{\chi,\gamma,\star} = \bigcup_{i\in[1,n]} \mathcal{S}_{\chi_i,\gamma_i,\star}$
 2: for $i \in [1,n]$ do
 3:     $(\mathcal{Q}_{\mathcal{T}_i}, \mathcal{Q}_{\mathcal{L}_i}) \leftarrow \mathsf{ElimLin}\left(\mathcal{S}_{\chi_i,\gamma_i,\star}\right)$
 4: end for
 5: for $(i,j) \in [1,n] \times [1,n]$ do
 6:     $(\mathcal{Q}_{\mathcal{T}_i}, \mathcal{Q}_{\mathcal{L}_i}) \leftarrow \mathsf{ElimLin}\left(\mathcal{Q}_{\mathcal{T}_i} \cup \mathcal{Q}_{\mathcal{L}_i} \cup \mathcal{Q}_{\mathcal{T}_j} \cup \mathcal{Q}_{\mathcal{L}_j}\right)$
 7: end for
 8: $L$ list of $(\mathcal{Q}_{\mathcal{T}_i}, \mathcal{Q}_{\mathcal{T}_i})$ for $(i,j) \in [1,n] \times [1,n]$
 9: return $\mathsf{ElimNode}\,(\ell, n, L)$
ElimNode:
Input: $a, b \in [1,n]$, $L$ list of $\mathcal{T}_{\{i,j\}}$ for $(i,j) \in [a,b] \times [a,b]$
Output: $\mathsf{ElimLin}\left(\mathcal{Q}_{\mathcal{T}_{[a,b]}}, \mathcal{Q}_{\mathcal{L}_{[a,b]}}\right)$
 1: if $b \neq a$ then
 2:     $\mathcal{Q}_{\mathcal{T}_{[a,b]}} \leftarrow \emptyset$
 3:     $\mathcal{Q}_{\mathcal{L}_{[a,b]}} \leftarrow \emptyset$
 4:     for $t = 0$ to $m-1$ do
 5:         $\ell \leftarrow a + t\frac{b-a}{m}$
 6:         $h \leftarrow a + (t+1)\frac{b-a}{m}$
 7:         $L' \leftarrow$ list of $\left(\mathcal{Q}_{\mathcal{T}_{[a,b]}}, \mathcal{Q}_{\mathcal{L}_{[a,b]}}\right)$ for $(i,j) \in [\ell,h] \times [\ell,h]$
 8:         $\left(\mathcal{Q}_{\mathcal{T}_{[a,b]}}, \mathcal{Q}_{\mathcal{L}_{[a,b]}}\right) \leftarrow \left(\mathcal{Q}_{\mathcal{T}_{[a,b]}}, \mathcal{Q}_{\mathcal{L}_{[a,b]}}\right) \cup \mathsf{ElimNode}\,(\ell, h, L')$
 9:     end for
10:     return $\mathsf{ElimLin}\left(\mathcal{Q}_{\mathcal{T}_{[a,b]}} \cup \mathcal{Q}_{\mathcal{L}_{[a,b]}}\right)$
11: else
12:     return $\mathsf{ElimLin}\left(\mathcal{Q}_{\mathcal{T}_{[a,b]}}, \mathcal{Q}_{\mathcal{L}_{[a,b]}}\right)$
13: end if
```

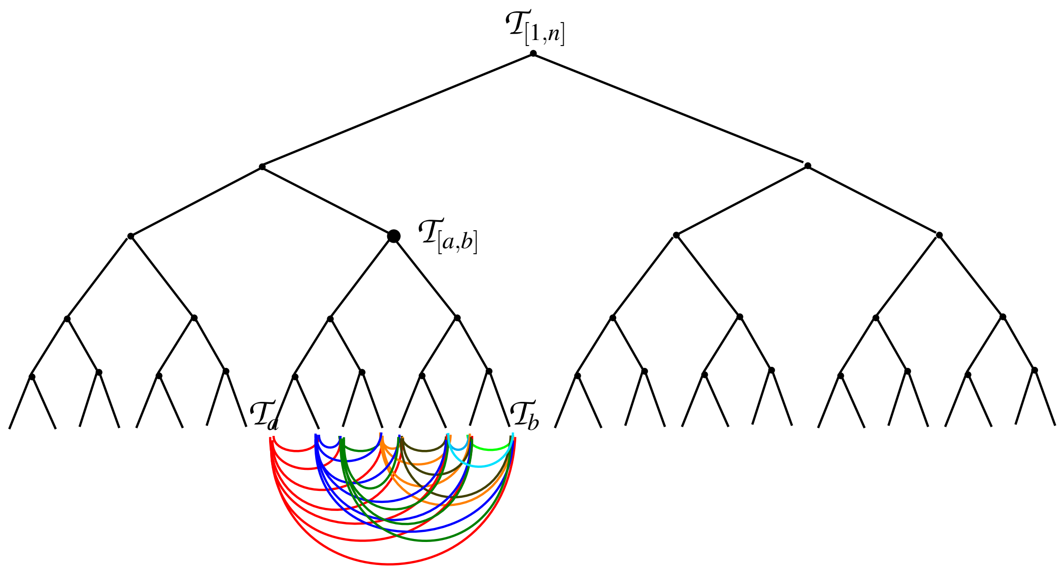


Figure 3.1: Divide-Conquer with leaves processing, Algorithm 5 for $m = 2$

4

The Selection of Samples

In this chapter, we investigate the performance of ElimLin. We first give simulations with a random selection of samples. Then, we introduce the cube selection of samples and we demonstrate that it outperforms a random strategy and more sophisticated techniques, such as selection of samples using truncated differential.
In Section 4.1, we give attack simulations without the selection of samples.
Then, we introduce our strategy for selection of samples in Section 4.2. First in Section 4.2.1, we discuss properties of systems where ElimLin succeeds. Then, we introduce our strategy for the selection of plaintexts and then, we show a link to cube attack in Section 4.2.3.
The results of this chapter were published at FSE2012 [CSSV12], ACISP14 [SSV14] and SECRYPT14 [CMS+14].

4.1 Multiple Samples Effect on ElimLin

We present attack simulations against LBlock which extend the results from [CSSV12]. However since we had no access to source code of ElimLin implementation in [Cou10], we run the tool using wine which may have led to a degradation in performance. The experiments were performed on 8-core Intel i7 CPU (Q740) running at 1.73GHz with 8GB of RAM. In both cases, we build a system of polynomial equations $\mathcal{S}_{\chi,\gamma,\star}$ where χ represents a list of multiple plaintexts and γ represents a list of corresponding ciphertexts obtained by an unknown secret key κ. This unknown key is recovered by running ElimLin $(\mathcal{S}_{\chi,\gamma,\star})$. For a comparison with a brute force attack, we consider an optimized implementation of cipher which requires 10 CPU cycles per round. All attacks in this chapter are faster than exhaustive search. In general, we consider a cipher broken even if we can recover only a few bits of the secret key.
We demonstrate the necessity of using multiple samples in algebraic attacks. We now

Table 4.1: samples: 5, rounds: 8, guessed bits: 28LSB

iteration	$\dim Q_T$	$\dim Q_L$
1	7440	6372
2	1068	208
3	860	139
4	721	70
5	651	44
6	607	25
7	582	18
8	564	3
9	561	0

Table 4.2: samples: 5, rounds: 8, guessed bits: 30LSB

iteration	$\dim Q_T$	$\dim Q_L$
1	7440	6374
2	1066	210
3	856	146
4	710	77
5	633	46
6	587	25
7	562	18
8	544	3
9	541	0

concentrate only on the results against LBlock. The results on CTC and MIBS were presented in [CSSV12]. LBlock is a Feistel-based block cipher for embedded devices. It takes 80-bit key and it operates on 64 bit blocks. It was presented at ACNS 2011 in [WZ11]. We present simulations of attacks against reduced round version of LBlock and demonstrate the impact of increased number of samples.

We compare our attack with algebraic solver called PolyBoRi which, unlike ElimLin, computes the Gröbner basis using the F4 algorithm. We consider an attack with 6 known plaintexts against 8-round LBlock. We further guessed 32-bits of secret key. Even though ElimLin can successfully break the cipher in 16 minutes, PolyBoRi crashed due to lack of memory.

In what follows, we show that the performance can be improved by selection of samples. Hence in what follows, we will concentrate on chosen plaintext attacks.

Table 4.3: samples: 5, rounds: 8, guessed bits: 32LSB

iteration	dim Q_T	dim Q_L
1	7440	6376
2	1064	214
3	850	152
4	698	93
5	605	64
6	541	26
7	515	16
8	499	3
9	496	0

Table 4.4: samples: 6, rounds: 8, guessed bits: 28LSB

iteration	dim Q_T	dim Q_L
1	8784	7524
2	1260	251
3	1009	176
4	833	93
5	740	58
6	682	36
7	646	18
8	628	4
9	624	0

Table 4.5: samples: 6, rounds: 8, guessed bits: 30LSB

iteration	dim Q_T	dim Q_L
1	8784	7526
2	1258	253
3	1005	184
4	821	103
5	718	63
6	655	37
7	618	18
8	600	4
9	596	0

Table 4.6: samples: 6, rounds: 8, guessed bits: 32LSB

iteration	dim Q_T	dim Q_L
1	8784	7528
2	1256	257
3	999	191
4	808	122
5	686	84
6	602	40
7	562	16
8	546	16
9	530	19
10	511	123
11	388	102
12	286	94
13	192	147
14	45	45

Table 4.7: samples: 7, rounds: 8, guessed bits: 28LSB

iteration	dim Q_T	dim Q_L
1	10128	8676
2	1452	294
3	1158	211
4	947	123
5	824	75
6	749	49
7	700	21
8	679	0

Table 4.8: samples: 7, rounds: 8, guessed bits: 30LSB

iteration	dim Q_T	dim Q_L
1	10128	8678
2	1450	296
3	1154	220
4	934	133
5	801	83
6	718	51
7	667	22
8	645	0

Table 4.9: samples: 7, rounds: 8, guessed bits: 32LSB

iteration	dim Q_T	dim Q_L
1	10128	8680
2	1448	300
3	1148	228
4	920	157
5	763	108
6	655	54
7	601	21
8	580	24
9	556	139
10	417	125
11	292	139
12	153	143
13	10	10

Table 4.10: samples: 8, rounds: 8, guessed bits: 28LSB

iteration	dim Q_T	dim Q_L
1	11472	9828
2	1644	337
3	1307	248
4	1059	149
5	910	90
6	820	59
7	761	23
8	738	0

Table 4.11: samples: 8, rounds: 8, guessed bits: 30LSB

iteration	dim Q_T	dim Q_L
1	11472	9830
2	1642	339
3	1303	258
4	1045	160
5	885	100
6	785	61
7	724	24
8	700	0

Table 4.12: samples: 8, rounds: 8, guessed bits: 32LSB

iteration	dim Q_T	dim Q_L
1	11472	9832
2	1640	343
3	1297	267
4	1030	189
5	841	130
6	711	65
7	646	23
8	623	21
9	602	156
10	446	145
11	301	153
12	148	141
13	7	7

Table 4.13: samples: 11, rounds: 8, guessed bits: 30LSB

iteration	dim Q_T	dim Q_L
1	15504	13286
2	2218	468
3	1750	373
4	1377	253
5	1124	161
6	963	113
7	850	30
8	820	2
9	818	2
10	816	0

Table 4.14: samples: 12, rounds: 8, guessed bits: 30LSB

iteration	$\dim Q_T$	$\dim Q_L$
1	16848	14438
2	2410	511
3	1899	411
4	1488	289
5	1199	189
6	1010	136
7	874	39
8	835	32
9	803	229
10	574	232
11	342	243
12	99	99

4.2 Improvements in the Selection of Samples

In this section, we propose a new method for the selection of samples in algebraic attack. We suggest a simple algorithm to determine sets of samples which allow ElimLin to break a higher number of rounds. First in part 4.2.1, we give a characterization of the system when ElimLin succeeds. We state Lemma 42 which introduces a polynomial which evaluates to a constant iff ElimLin succeeds. Unlike in cube attack (which we recall in Section 4.2.3), we cannot evaluate this polynomial. Hence, we make a heuristic assumption that this polynomial evaluates to a constant iff a cube attack/cube distinguisher on a reduced-round cipher is successful. We show this for LBlock, KATAN32 and SIMON. Table 4.19 shows the number of independent linear equations in key variables recovered by ElimLin in the case of SIMON. Our selection strategy is described in part 4.2.2. This selection strategy is based on cube attacks which we recall in part 4.2.3. In part 4.3, we show the performance of such a technique on LBlock, and compare our results to previous algebraic attacks based on ElimLin. In part 4.2.4, we give further insight of our method and directions for future testing and improvements.

4.2.1 Characterization of systems when ElimLin succeeds

We now explore the properties of systems for which ElimLin succeeds to recover the secret key. We use this characterization in Part 4.2.2 to derive a selection strategy for plaintexts.
We reformulate ElimLin (Algorithm 3) based on matrix operations. Let us consider matrices over $\mathbf{F}_2[V]$, i.e, the original polynomial system $Q = \{\mathsf{Eq}_1, \dots, \mathsf{Eq}_{m_0}\}$ is repre-

sented as $\begin{pmatrix} \mathsf{Eq}_1 \\ \vdots \\ \mathsf{Eq}_{m_0} \end{pmatrix}$.

$\mathsf{ElimLin}$ performs Gauss elimination and substitutions. The Gauss elimination (step 14 of Algorithm 3), corresponds to the matrix multiplication.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ \vdots & & & \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mathsf{Eq}_1 \\ \mathsf{Eq}_2 \\ \vdots \\ \mathsf{Eq}_{m_0} \end{pmatrix} = \begin{pmatrix} \mathsf{Eq}_1 \\ \mathsf{Eq}_1 + \mathsf{Eq}_2 \\ \vdots \\ \mathsf{Eq}_{m_0} \end{pmatrix}.$$

The substitution can be expressed as multiplication by a polynomial. Let us assume substitution of x_1 using linear polynomial $\mathsf{Eq}_1 = x_1 + \ell(\overrightarrow{x})$. Assume that $\mathsf{Eq}_2 = x_1 p + q$ where $x_1$does not appear in ℓ, p and q. After substitution, we obtain $\ell(\overrightarrow{x})p + q$. Hence, the substitution of x_1 in Eq_2 can be expressed as a matrix multiplication

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ p & 1 & 0 & 0 \\ & & \vdots & \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mathsf{Eq}_1 \\ \mathsf{Eq}_2 \\ \vdots \\ \mathsf{Eq}_{m_0} \end{pmatrix} = \begin{pmatrix} (x_1 + \ell(\overrightarrow{x})) \\ p\mathsf{Eq}_1 + \mathsf{Eq}_2 \\ \vdots \\ \mathsf{Eq}_{m_0} \end{pmatrix}.$$

We use the asociativity and express $\mathsf{ElimLin}(\mathcal{Q})$ by a single matrix multiplication, say $\mathsf{E} \cdot \mathcal{Q}$. We implicitely assume that computations are done modulo $\mathsf{FieldEq}[V]$.

Lemma 42. *Consider a system $\mathcal{S}$ such that $\mathsf{ElimLin}$ applied to $\mathcal{S}_{\chi,\gamma,\star}$ recovers the key bit k_j as value $c_j \in \mathbf{F}_2$. Let E be the $\mathsf{ElimLin}$ transformation on $\mathcal{S}_{\chi,\gamma,\star}$. We define $\mathcal{S}'$ as the system $\mathcal{S}_{\chi,\gamma,\star}$ where all equations $\mathsf{s}^j_{p,0} - \chi^j_p$ and $\mathsf{s}^j_{p,\mathsf{rndn}} - \gamma^j_p$ are replaced by 0. $\mathcal{S}'$ is thus equivalent to $\mathcal{S}$. Then,*

$$\mathsf{Eval}_{\chi,\gamma,\star}\left(\mathsf{E} \cdot \mathcal{S}'\right) = \mathsf{E} \cdot \mathcal{S}_{\chi,\gamma,\star}$$

and whenever the matrix $\mathsf{E} \cdot \mathcal{S}_{\chi,\gamma,\star}$ contains a polynomial $k_j + c_j$ the matrix $\mathsf{E} \cdot \mathcal{S}'$ contains a line $k_j + c_j + q'$ with $\mathsf{Eval}_{\chi,\gamma,\star}(q') = 0$.

Proof. *We consider the line α of matrix $\mathsf{E} \cdot \mathcal{S}_{\chi,\gamma,\star}$ leading to $k_j + c_j$.*

$$\begin{aligned} k_j + c_j &= \left(\mathsf{E} \cdot \mathcal{S}_{\chi,\gamma,\star}\right)_\alpha \\ &= \sum_i p_{i,\alpha} \mathsf{Eq}_i + \sum_{p_j} q_{p_j} \left(\mathsf{s}^j_{p,0} - \chi^j_p\right) + \sum_{p_j} r_{p_j} \left(\mathsf{s}^j_{p,\mathsf{rndn}} - \gamma^j_p\right) \bmod \mathsf{FieldEq}[V]. \end{aligned}$$

Let us write

$$q = \sum_i p_{i,\alpha} \mathsf{Eq}_i \bmod \mathsf{FieldEq}[V]$$

and

$$q' = q + k_j + c_j.$$

Then,

$$\mathsf{Eval}_{\chi,\gamma,\star}(q') = \mathsf{Eval}_{\chi,\gamma,\star}\left(\sum_{p_j} q_{p_j}\left(s^j_{p,0} - \chi^j_p\right) + \sum_{p_j} r_{p_j}\left(s^j_{p,rndn} - \gamma^j_p\right)\right) = 0.$$

In the first case, the line α contains polynomial $k_j + c_j$. Hence in the second case, we also obtain the polynomial $k_j + c_j$. We set

$$q' = k_j + c_j + \sum_i p_{i,\alpha}\mathsf{Eq}_i + \sum_{p_j} q_{p_j} s^j_{p,0} + \sum_{p_j} r_{p_j} s^j_{p,rndn}$$

and obtain the claim. □

The polynomial q' will be important in the selection strategy of the plaintexts. The existence of such polynomial is essential for ElimLin to be able to recover the secret key. At the same time, the chance of existence of such polynomial can be improved if we select samples based on a successful cube attack.

We consider systems $\mathcal{S}_{\chi,\star,\star}$ and $\mathcal{S}_{\chi',\star,\star}$. Then, we run ElimLin and recover matrices E and E' such that $\mathsf{ElimLin}\left(\mathcal{S}_{\chi,\star,\star}\right) = \mathsf{E}\cdot\mathcal{S}_{\chi,\star,\star}$ and $\mathsf{ElimLin}\left(\mathcal{S}_{\chi',\star,\star}\right) = \mathsf{E}'\cdot\mathcal{S}_{\chi',\star,\star}$. Due to Theorem 35, at one stage of $\mathsf{ElimLin}\mathcal{S}_{\chi\cup\chi',\star,\star}$, we encounter an ElimLin matrix which generates the same space as the matrix $\binom{\mathsf{E}}{\mathsf{E}'}$. Let us now consider the rank of the matrix $\binom{\mathsf{E}}{\mathsf{E}'}$. The rank is minimal if $\mathsf{hw}(\chi+\chi') = 0$, i.e, $\chi = \chi'$. Similarly if $\mathsf{hw}(\chi+\chi') = 1$, messages χ and χ' are very close and hence, the $\mathsf{ElimLin}\left(\mathcal{S}_{\chi,\star,\star}\right)$ and $\mathsf{ElimLin}\left(\mathcal{S}_{\chi',\star,\star}\right)$ perform many equivalent substitutions and hence, the intersection of vector spaces generated by lines of E and E' is large and the rank of the matrix $\binom{\mathsf{E}}{\mathsf{E}'}$ is smaller than for a random pair (χ,χ'). In the case of cube attack and a cube of "dimension" m, any pair of messages (χ,χ') has $\mathsf{hw}(\chi+\chi') \leq m$. This suggest that cube attack is a good candidate for selection of samples for ElimLin.

4.2.2 Cube Based Selection Strategy for Plaintexts in ElimLin

We gave a characterization of the polynomial system when ElimLin recovers the value of the key k_j in Lemma 42. We showed that ElimLin can succeed only if there exists a polynomial q in the ideal spanned by the polynomial system $\langle\mathcal{S}\rangle$, such that $\mathsf{Eval}_{\chi,\gamma,\star}(q)$ is a linear polynomial in the key variables. We now consider the polynomial q' from Lemma 42. As we cannot choose simultaneously the plaintext and the ciphertext for a single sample, we consider several different scenarios: selecting plaintexts only, ciphertexts only, selecting partly plaintexts and partly ciphertexts. The selection of related plaintexts such that corresponding ciphertexts are also related is considered in [CMS$^+$14]. These pairs are constructed using higher order and/or truncated differential cryptanalysis [Knu94]. In our scenario, we concentrate on the selection of only plaintexts. We found no advantage in the selection of only ciphertexts. The selection of part of plaintexts and part of ciphertexts is yet to be explored. The selection of related plaintexts and corresponding ciphertexts is specific to a chosen cipher. However, our goal is to determine an optimal generic selection of samples. We use Lemma 42 for the selection of plaintexts. It specifies the properties of q' which has to evaluate to 0 when we set plaintext and ciphertext variables, i.e, when we set χ and γ. However, we would like to guarantee that q' evaluates to 0 only when setting the plaintexts, as we cannot control both the plaintexts and the ciphertexts. Hence, we are looking for a set of samples that lead to an existence of such q' when we set only plaintext variables. Let $\mathsf{deg}_r(p)$ denote the total degree of the polynomial p in variables corresponding to round r, i.e, $\mathsf{s}^r_{1,1}, \dots, \mathsf{s}^r_{\mathsf{smpn},\mathsf{mln}}$. Provided the $\mathsf{deg}_0(q') < D$, we can build a set of 2^D samples, i.e, find χ, such that q' evaluates to 0. This leads us to setting values χ according to a cube recovered from cube attack.

4.2.3 Revisiting Cube Attacks

The cube attack [DS09a] can be seen as a tool to analyze a black-box polynomial which we represent by $f(x,k)$. Cube attack can be seen as a method to investigate the diffusion of the cipher and the selection of the cube is the selection of plaintexts such that diffusion is minimal. It was first described as Algebraic IV Differential Atack (AIDA). The relation of cube attacks, cube testers, AIDA and High Order Differential attack was explored in [ZGLC]. The aim is to derive a polynomial system which is easy to solve and which is satisfied for all keys, i.e, for all values of k. The attacker does this in an offline phase. Afterwards, in an online phase, the attacker finds the evaluation for each polynomial and solves the system. We query this polynomial in an offline phase for both parameters x and k. In the online phase, we are allowed to use queries only in the first parameter x, as k is set to an unknown value κ.

The objective is to recover this κ. To achieve this, we find a hidden structure of $f(x,k)$ in the offline phase and use it to derive κ in the online phase. In the offline phase, we find sets of plaintexts C_i such that $\sum_{x \in C_i} f(x,k)$ behaves like a linear function $\ell_i(k)$ and ℓ_i's are linearly independent. In the online phase, we ask the oracle for encryptions of plaintexts from C_i and solve the system of linear equations. In the following, we derive the algebraic expression of $\sum_{x \in C_i} f(x,k)$ and show that this function can indeed behave like a function $\ell(k)$. Let $f(x,k)$ be a black-box polynomial which can be for some coefficients $a_{IJ} \in \mathbf{F}_2$ expressed as

$$f(x,k) = \sum_{\substack{I \subseteq \{0,1\}^{\mathsf{mln}} \\ J \subseteq \{0,1\}^{\mathsf{kln}}}} a_{IJ} \prod_{i \in I} x_i \prod_{j \in J} k_j.$$

Definition 43. *Let $m \in \{0,1\}^{mln}$ and $t \in \{0,1\}^{mln}$ such that $t \wedge m = 0$. We define $C_{m,t} = \{x : x \wedge \bar{m} = t\}$. We call $C_{m,t}$ a "cube", m a "mask", and t a "template", and we denote $I_m = \{i : 2^i \wedge m \neq 0\}$, where 2^i represent the bitstring with 1 at position i.*

Example: Let $m = 00010110$ and $t = 11100001$. Then, we have $|C_{m,t}| = 2^3$ and

$$C_{m,t} = \left\{ \begin{array}{c} 11110111, \\ 11110101, \\ 11110011, \\ 11110001, \\ 11100111, \\ 11100101, \\ 11100011, \\ 11100001 \end{array} \right\}$$

Theorem 44. *Let $C_{m,t}$ be a cube, and $f(x,k) = \sum_{\substack{I \subseteq \{0,1\}^{mln} \\ J \subseteq \{0,1\}^{kln}}} a_{IJ} \prod_{i \in I} x_i \prod_{j \in J} k_j$. Then,*

$$\sum_{x \in C_{m,t}} f(x,k) = \sum_{\substack{J \subseteq \{0,1\}^{kln}, \\ I : I_m \subseteq I}} a_{IJ} \prod_{i \in I} t_i \prod_{j \in J} k_j = \sum_{J} a'_J \prod_{j \in J} k_j$$

for $a'_J = \sum_{I : I_m \subseteq I} a_{IJ} \prod_{i \in I} t_i$.

Proof.

$$\begin{aligned}
\sum_{x\in C_{m,t}} f(x,k) &= \sum_{x\in C_{m,t}} \sum_{IJ} a_{IJ} \prod_{i\in I} x_i \prod_{j\in J} k_j \\
&= \sum_{IJ} a_{IJ} \left(\sum_{x\in C_{m,t}} \prod_{i\in I} x_i \right) \prod_{j\in J} k_j \\
&= \sum_{IJ} a_{IJ} \left(\left(\sum_{x\in C_{m,t}} \prod_{i\in I\cap I_m} x_i \right) \prod_{i\in I\setminus I_m} t_i \right) \prod_{j\in J} k_j \\
&\overset{\star}{=} \sum_{IJ} a_{IJ} \left(1_{I_m\subseteq I} \prod_{i\in I\setminus I_m} t_i \right) \prod_{j\in J} k_j \\
&= \sum_{\substack{J,\\ I:I_m\subseteq I}} a_{IJ} \prod_{i\in I} t_i \prod_{j\in J} k_j \\
&= \sum_{J} \left(\sum_{I:I_m\subseteq I} a_{IJ} \prod_{i\in I} t_i \right) \prod_{j\in J} k_j \\
&= \sum_{J} a'_J \prod_{j\in J} k_j
\end{aligned}$$

The equality $\star$ is satisfied, since

$$\sum_{x\in C_{m,t}} \prod_{i\in I\cap I_m} x_i = \begin{cases} 0 & \text{if } I \not\supseteq I_m \\ 1 & \text{if } I \supseteq I_m \end{cases}$$

This holds because $\prod$ appears twice in the sum for every $i \in I \setminus I_m$. □

The success of cube attacks is based on finding enough cubes C_{m_i,t_i}, i.e, enough m_is, t_is, such that

$$\sum_{\chi\in C_{m_i,t_i}} f(x,k) = \sum_{J\subseteq\{0,1\}^{\mathsf{kln}}} a^i_J \prod_{j\in J} k_j \tag{4.1}$$

are linearly independent low degree equations.

In the case of block ciphers, we have mln statebits we can consider for cube attack. Hence, we characterize the cube $C_{m,t}$ by the number of linear equations obtained from cube attack.

Definition 45. *Let $C_{m,t}$ be a cube for r-round cipher and let f_j be a blackbox polynomial* *corresponding to j-th bit of the state. Then, we call cuberank the number of equations $\sum_{x\in C_{m,t}} f_j(x,k)$ which have the right-hand side in Eq. 4.1 linear.*

Even though cube attacks may be a powerful tool in algebraic cryptanalysis, it has been successful against only very few ciphers. The reduced round TRIVIUM [Can06] can be attacked for 784 and 799 rounds [FV13], and can be distinguished with 2^{30} samples up to 885 rounds [ADMS09]. The full round TRIVIUM has 1152 rounds, which means that 70% of the cipher can be broken by this simple algebraic technique. GRAIN128 [HJM07] was broken using the so called dynamic cube attack in [DS09a]. KATAN32 was attacked in [BCN$^+$10] using the so called side-channel cube attack first introduced in [DS09b]. In [ALRSS11], the authors considered cubes leading to non-linear relations among key variables. While cube attacks and cube distinguishers celebrate success in only few cases, we show that they can be used for selection of samples in other algebraic attacks.

4.2.4 ElimLin and Cube Attacks

In this section, we explain the intuition behind using a cube attack for selecting samples for ElimLin. We first elaborate on our observations about ElimLin's ability to recover the equation found by cube attack. Later, we compare our approach to classical cube attacks and we give additional observations about the behavior of ElimLin with our selection of samples.

Structure of the cube. Let E_κ denote the encryption under the key κ, and let us consider two samples for the plaintexts χ and $\chi+\Delta$, where Δ has a low Hamming weight. Many statebits in the first rounds of computation $E_\kappa(\chi)$ and $E_\kappa(\chi+\Delta)$ take the same value, as they can be expressed by the same low degree polynomial in the key and state variables. This can be detected by ElimLin and used to reduce the total number of variables of the system. Therefore, good candidates for the selection of samples are plaintexts which are pairwise close to each other — in other words, plaintexts from a cube. Let us now consider $\chi = (\chi_p : \chi_p \in C_{m,t})$. We consider a blackbox polynomial $f(x,k)$ computing the value of state variable $\mathsf{s}^j_{x,r}$ for a key κ, a plaintext x, a statebit j and r rounds. This blackbox polynomial is formalized in Chapter 5, Definition 49 as $e_\chi\big|_\kappa$. The cube attack gives an equation $\sum_{\chi_p\in C_{m,t}} f(\chi_p,k) = \ell(k)$ for a linear function ℓ. We observe that the equation $\sum_{\chi_p\in C_{m,t}} f(\chi_p,k) = \ell(k)$ is found also by ElimLin in a majority of cases. We further found that ElimLin can recover many pairs of indices (a,b), such that $\mathsf{s}^j_{a,r}$ equals to $\mathsf{s}^j_{b,r}$. We assume that this is the fundamental reason for the success of cube attack. Thanks to such simple substitutions, ElimLin can break a higher number of rounds while decreasing the running time. The strategy of selecting correlated samples was also explored in [FP10]. The authors chose messages based on an algebraic-high order differential.

ElimLin vs. Cube Attacks. The attack based on cube attack consists of an expensive offline phase, where we build the system of equations which is easy to solve, i.e, linear (or low degree) equations in the key bits; and the online phase where we find evaluations for these linear equations and solve the system. The attack based on ElimLin consists of a cheap offline phase, as the system of equations represents the encryption algorithm, and the online phase is therefore more expensive. Our attack can be seen as a mix of these two approaches. We increase the cost of the offline phase to find a good set of samples and run ElimLin on the system without the knowledge of ciphertext. Hence, we simplify the system for the online phase which is subsequently faster.

Comparison of the number of attacked rounds by Cube Attacks and by ElimLin with the same samples. In our attacks, we observed an interesting phenomena which occurs for every cipher we tested. Our first phase consists of finding a cube attack against a R round ciphers. In the next phase, we consider $R+r$ round cipher, build a system of equations, set plaintext bits correspondingly, and run ElimLin to obtain a system P. In the next step, we query the encryption oracle for ciphertexts, build a system of equations corresponding to rounds $[R, R+r]$, and run ElimLin to obtain a system C. We found that the success of ElimLin to recover the secret key of $R+r$ round cipher strongly depends on the selection of plaintexts: random samples perform worse than random cubes and random cubes preform worse than the ones which perform well in cube attack. The plaintexts selected based on a cube allow ElimLin to find more linear relations, which are in many cases of form $\mathsf{s}^j_{a,r} = \mathsf{s}^j_{b,r}$. Hence, we obtain a system with significantly less variables. This allows us to recover the secret key. In the cases of LBlock and KATAN32 we obtained $r \approx \frac{R}{3}$. These observation suggest a further research in performance of ElimLin against ciphers such as TRIVIUM and GRAIN128, as cube attacks against a significant number of rounds [FV13, DS11, ADMS09] already exists.

4.3 LBlock: Selection of plaintexts

In this section, we show that the selection of plaintexts based on the success of cube attack is a good strategy for satisfying the condition from Section 4.2.1. We give an attack against 10 rounds of LBlock. This attack outperforms the previous attempts of algebraic cryptanalysis [CSSV12]. We compare our strategy of using samples for cube attack to the strategy of selecting a random cube or a random set of samples.

Description of LBlock. LBlock is a lightweight block cipher which operates on block size of 64-bit and the key size is 80-bit. The scheme is shown in Figure 4.1. LBlock is optimized for hardware implementation, but it has a good performance in software as well. The proposal was analysed by authors in [WZ11] and subsequently analysed in [WW14, SW12, SN12]. LBlock is a modified Fiestel scheme with a permutation implemented using multiple S-boxes.

Breaking 8 rounds of LBlock. The previous result on breaking 8 rounds of LBlock using ElimLin required 6 random plaintexts, and guessing 32 least significant bits of the key (out of 80bits). These results can be found in Table 4.6. We found that if we select 8 plaintexts based on cube $C_{m,t}$ for m=0x0000000000000007 and t=0xe84fa78338cd9fb0, we break 8 rounds of LBlock without guessing any key bits. We verified this result for 100 random keys and in each case, we were able to recover the secret key using ElimLin.

Breaking 10 rounds of LBlock. We are not aware of any previous successful attempts of breaking 10 rounds of LBlock by algebraic technique. Following the approach for 8 rounds, we found 16 plaintexts based on a cube $C_{m,t}$ for m=0x0000000000003600 and t=0xe84fa78338cd89b6, we break 10-rounds of LBlock without guessing any key bits. We verified this result for 100 random keys. We were able to recover each of the 100 secret keys we tried using ElimLin. We tried to extend the attack to 11 rounds of LBlock, however we have not found any cube of dimension 5 or 6 which would allow ElimLin to solve the system.

Random vs Non-Random Selection of Plaintexts. We tested the performance of ElimLin applied to 10-round LBlock for the same number of plaintext-ciphertext pairs. Our results show that when ElimLin algorithm is applied to a set of n plaintexts from a cube, the linear span it recovers is larger than for a set of n random samples. We also show that ElimLin behaves better on some cubes, and that this behavior is invariant to affine transformation. The results are summarized in Table 4.15. In LBlock, we found no advantage between random and nonrandom template t. However in KATAN32, we found the choice of the template t is very important. In Table 4.16, we show that 69 rounds of KATAN32 was not solved for template t=0x00000000, t=0xf0000000, t=0x0f000000, etc. But at the same time, we broke 70 rounds of KATAN32 using the same mask and template t=0x39d88a02.

4.4 KATAN32: Selection of samples

KATAN is an efficient hardware oriented block cipher. It comes in three different versions: 32, 48 and 64 block sizes. They all share an 80-bit key. It also comes with an even lighter version KTANTAN which has a different key scheduling algorithm.

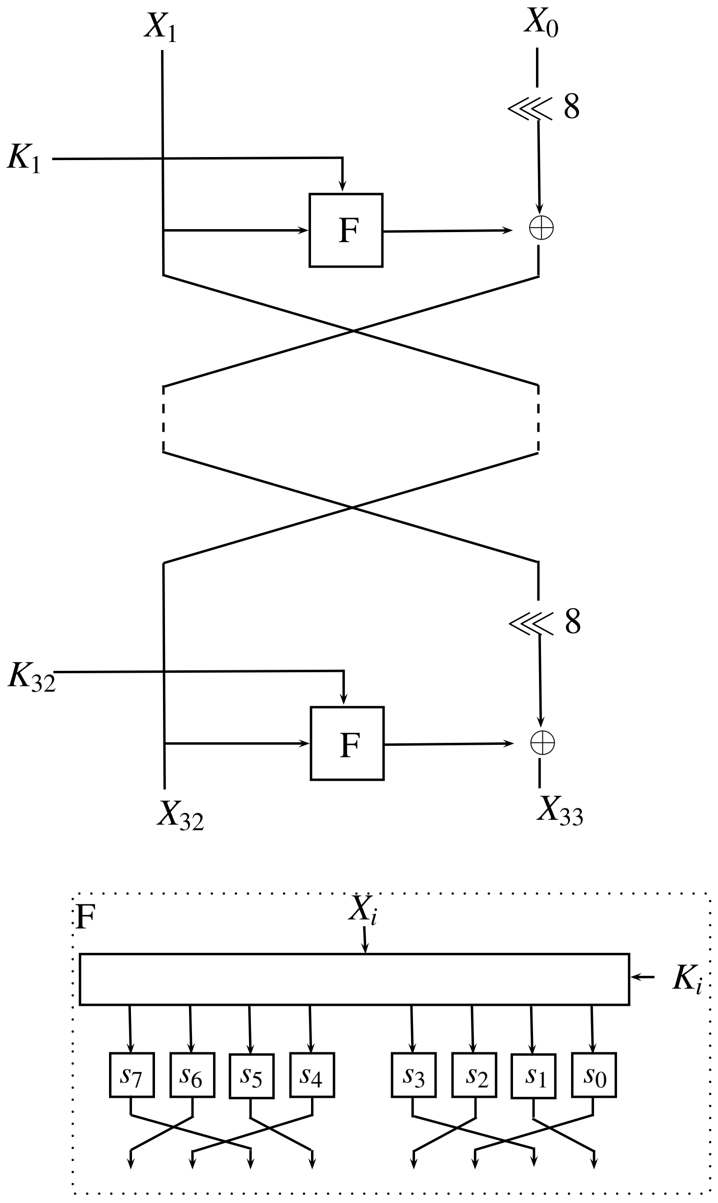


Figure 4.1: LBlock

10 rounds of LBlock: $C_{m,t}$ system of 2^4 samples		solved	remaining variables
m=0x0000000000003600	t=0xe84fa78338cd89b6	yes	0
m=0x0000000000d00001	t=0x856247de122f7eaa	yes	0
m=0x0000000000003600	random	yes	0
m=0x0000000000d00001	random	yes	0
m=random deg4	random	no	≈ 700
random set		no	≈ 2000

Table 4.15: Results on 10-round LBlock

KATAN32 uses two nonlinear functions f_a and f_b in each round. The nonlinear functions are defined as follows.
$f_a(L_1) = L_1[x_1] + L_1[x_2] + L_1[x_3]L_1[x_4] + L_1[x_5] \cdot \mathsf{IR} + k_a$
$f_b(L_2) = L_2[y_1] + L_2[y_2] + L_2[y_3]L_2[y_4] + L_2[y_5]L_2[y_6] + k_b$ where IR is a so-called irregular update rule and k_a, k_b are the two subkey bits. The values x_i, y_i are defined for each variant separately. After the computation of nonlinear functions, the registers L_1 ad L_2 are shifted so that MSB of L_i goes out and is loaded as LSB into $L_{i+1 \bmod 2}$. The representation of KATAN32 can be found in Figure 4.2. The key schedule of KATAN32 cipher loads 80-bit key into a linear feedback shift register (LFSR). At each round the positions 0 and 1 of LFSR are generated as the round's subkey k_{2i}, k_{2i+1} and LFSR is clocked twice. The feedback polynomial is

$$x^{80} + x^{61} + x^{50} + x^{13} + 1$$

Hence the keyschedule is computed recursively as follows:

$$k_i = \begin{cases} K_i & \text{if } 0 \geq i \leq 79 \\ k_{i-80} + k_{i-61} + k_{i-50} + k_{i-13} & \text{otherwise} \end{cases}$$

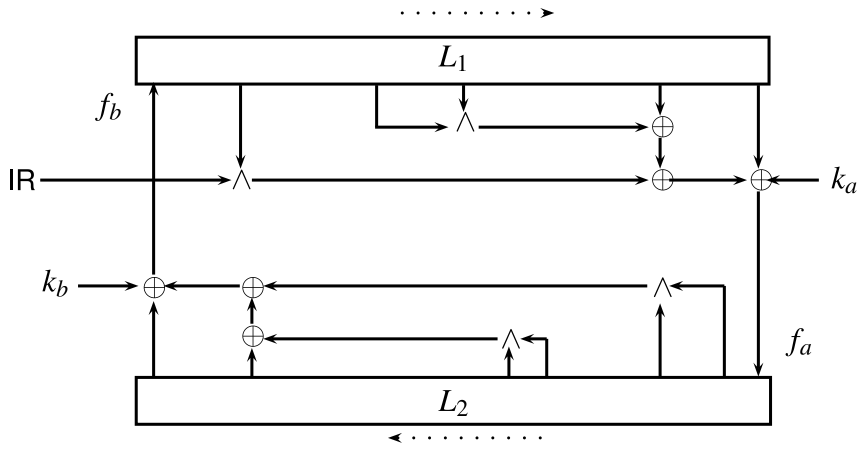


Figure 4.2: KATAN32

Previous results of algebraic cryptanalysis. The previous best algebraic attack is given by Bard et al. [BCN+10]. They use SAT solvers against KATAN and they managed to break 79 rounds of KATAN32 using SAT solver using 20 chosen plaintexts with 45 guessed key bits. Furthermore, they broke 75 rounds of KATAN32 with 35 guessed key bits. The authors further tried combine cube attack and SAT solver. They used samples based on a relatively small cubes which lead to linear equations among key variables.

Then, they converted this system into a boolean formula and applied SAT solver. Hence, the SAT solver would obtain several linear equations among key bits which should help to speed up the key search.
In our work, we take a different approach. First, we combine cube attack with ElimLin which does not perform any guessing of key bits or state variables. This means we provide less information to ElimLin and hence, the SAT solver should perform significantly better. However, this allows us to evaluate our strategy for selection of samples more precisely. Second, we apply the cube attack on a smaller number of rounds than we intend to attack. This results in finding smaller cubes. Then, we build a relatively large system of samples in our cube and we solve it by ElimLin. However, previous implementations of ElimLin could not cope with systems as large as what we obtained from cube attack and hence, we developed a new more optimized ElimLin solver.
We give the results of the attack against KATAN32 in Table 4.17. We performed the measurements on 16 core Xeon 3.3GHz with 72GB of memory. In our attacks, we do not guess any key bit and achieve a comparable number of rounds. However, we need to use more plaintext ciphertext pairs (128 – 1024 instead of 20). The main advantage of our attack is not only the fact that we do not need to guess the key bits, but also its determinism. As the success of other algebraic attacks such as SAT solvers and Gröbner basis depends on the performance of ElimLin, our results may be applied in these scenarios for improving the attacks. In Table 4.16, we show that the selection of samples is important for KATAN32. The reader can observe that in the case of 69 rounds, the template of the cube is important for ElimLin to succeed. In the case when the template was selected based on cube attack for 55 rounds, the attack using ElimLin was successful to recover the key. However, when we use the same mask but a random template, ElimLin could not recover any key bit. We can also observe when the number is maximal for this set of plaintexts: when we increase the number of rounds, ElimLin fails to recover the key. The reader can also see that an increase in the number of samples allows to break more rounds in some cases. In the case of 71 rounds, we extend the mask of the cube by one bit and in one case we can recover the key using ElimLin. In the other case, we cannot. In the case of 76 rounds, we were unable to break the system for any cube attack for 55 rounds. However, we found a cube attack of 59 rounds, which allowed ElimLin to solve the system for 76 round KATAN32 and 256 samples. In Table 4.17, we give successful results of attack by ElimLin applied on reduced round KATAN32 for various number of rounds. The previous best algebraic attacks can be found in [BCN$^+$10]. The authors guess 35 out of 80 bits of the key and solve the system using SAT solver. We can achieve the same amount of rounds without any key guessing and with a running time within several hours.

Table 4.16: Attack on KATAN32 using ElimLin: rounds vs. masks

rnd	cube rnd	mask	template	samples	success	time
69	55	m=0x00007104	t=0x39d88a02	32	yes	<1 hour
69	55	m=0x00007104	t=0x65f30240	32	yes	<1 hour
69	n.a	m=0x00007104	t=0x00000000	32	no	2 hours
69	n.a	m=0x00007104	t=0xf0000000	32	no	2 hours
69	n.a	m=0x00007104	t=0x0f000000	32	no	2 hours
69	n.a	m=0x00007104	t=0x00f00000	32	no	2 hours
70	55	m=0x00007104	t=0x39d88a02	32	no	3 hours
70	55	m=0x00007104	t=0x65f30240	32	no	3 hours
71	55	m=0x00007105	t=0x23148a40	64	yes	3 hours
71	55	m=0x00007904	t=0x20128242	64	no	7 hours
76	59	m=0x0004730c	t=0x21638040	256	yes	3 days

Table 4.17: Attack on KATAN32 using ElimLin

rnd	cube rnd	mask	template	samples	solved/tests	time
71	55	m=0x0002700c	t=0xf2b50080	64	5/5	<1 hour
70	55	m=0x0c007104	t=0xa2d88a61	128	5/5	<1 hour
70	55	m=0x00a07104	t=0x50570043	128	5/5	<1 hour
71	55	m=0x00007105	t=0x23148a40	64	10/10	3 hours
72	55	m=0x00a07104	t=0x50570043	128	20/20	7 hours
72	55	m=0x0c007104	t=0xa2d88a61	128	60/60	7 hours
73	55	m=0x0c007104	t=0xa2d88a61	128	5/5	7 hours
73	55	m=0x0002d150	t=0x20452820	128	20/20	8 hours
73	55	m=0x0002d150	t=0xffd40821	128	20/20	8 hours
74	56	m=0x10826048	t=0xca458604	128	5/5	9 hours
75	56	m=0x80214630	t=0x76942040	256	5/5	23 hours
75	56	m=0x1802d050	t=0x267129a8	256	5/5	23 hours
75	56	m=0x908a1840	t=0x6b05c0bd	256	5/5	23 hours
75	56	m=0x08030866	t=0x8620f000	256	5/5	23 hours
75	56	m=0x52824041	t=0x0d288d08	256	5/5	23 hours
75	56	m=0x10027848	t=0xcf758200	256	5/5	23 hours
76	59	m=0x0004730c	t=0x21638040	256	3/3	3 days
77	59	m=0x03057118	t=0x2cb20001	1024	3/3	8 days
78	59	m=0x03057118	t=0x2cb20001	1024	2/2	9 days

4.5 SIMON: Selection of samples

In this section, we show that the selection of samples based on the cube attack can significantly improve the performance of algebraic attacks against SIMON. In [CMS+14], the authors show that the selection of samples based on Truncated Differential gives a substantial advantage over the random selection of samples. They present an attack against 10 round SIMON using ElimLin. In what follows, we present an attack against 13-round SIMON using ElimLin. We achieve this improvement by selecting the samples such that a cube attack on 10 round SIMON can find a linear (or constant) equation in key bits.

4.5.1 General Description of SIMON

SIMON is a family of lightweight block ciphers with the aim to have optimal hardware performance [BSS+13]. It follows the classical Feistel design paradigm, operating on two n-bit halves in each round, and thus the general block size is $2n$. The SIMON block cipher with an n-bit word is denoted by SIMON-$2n$, where $n = 16, 24, 32, 48$ or 64 and if it uses an m-word key (equivalently mn-bit key), we denote it as SIMON-$2n/mn$. In this paper, we study the variant of SIMON with $n = 32$ and $m = 4$ (i.e 128-bit key). Each round of SIMON applies a non-linear, non-bijective (and as a result non-invertible) function $F : \mathbf{F}_2^n \rightarrow \mathbf{F}_2^n$ to the left half of the state which is repeated for 44 rounds. The operations used are as follows:

1. bitwise XOR, $\oplus$
2. bitwise AND, $\wedge$
3. left circular shift, S^j by j bits.

We denote the input to the i-th round by $L^{i-1}||R^{i-1}$ and in each round the left word L^{i-1} is used as input to the round function F defined by,

$$F(L^{i-1}) = (L^{i-1} \lll 1) \wedge (L^{i-1} \lll 8) \oplus (L^{i-1} \lll 2)$$

Then, the next state $L^i||R^i$ is computed as follows (cf. Fig. 4.3),

$$L^i = R^{i-1} \oplus F(L^{i-1}) \oplus K^{i-1}$$

$$R^i = L^{i-1}$$

The output of the last round is the ciphertext.

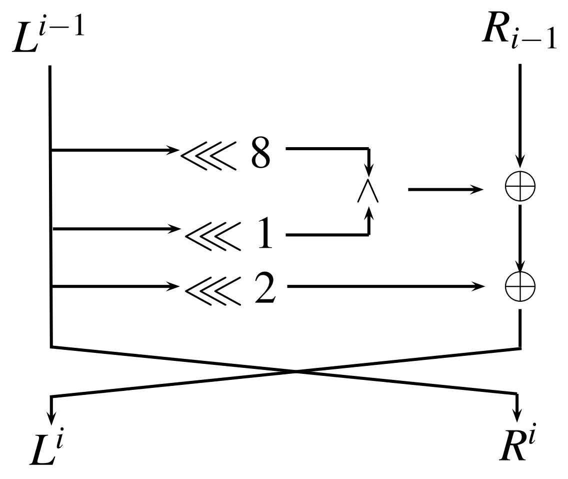


Figure 4.3: The round function of SIMON

The key schedule of SIMON is based on an LSFR-like procedure, where the *nm*-bits of the key are used to generate the keys $K_0, K_1, ..., K_{r-1}$ to be used in each round. There are three different key schedule procedures depending on the number of words that the secret key consists of ($m = 2,3,4$).
At the beginning, the first m words $K^0, K^1, ..., K^{m-1}$ are initialized with the secret key, while the remaining are generated by the LSFR-like construction. For the variant of our interest, where $m = 4$, the remaining keys are generated in the following way:

$$Y = K^{i+1} \oplus (K^{i+3} \ggg 3)$$

$$K^{i+4} = K^i \oplus Y \oplus (Y \ggg 1) \oplus c \oplus (z_j)_i$$

The constant $c =$ `0xff...fc` is used for preventing slide attacks and attacks exploiting rotational symmetries [BSS+13]. In addition, the generated subkeys are xored with a bit $(z_j)_i$, that denotes the i-th bit from the one of the five constant sequences $z_0, ..., z_4$. These sequences are defined in [BSS+13] and for our variant we use z_3. The equation generator of SIMON and SPECK ciphers can be found in [Son14].

4.5.2 Limitations of Cube Selection

We set up an offline phase of cube attack against 10 and 11 round SIMON to select plaintext/ciphertext pairs used to build a polynomial system of 13 round SIMON which is afterwards solved by ElimLin. We rank the results from offline phase of cube attack as follows. In Section 4.2.3, we reviewed the cube attacks as an attack against a black-box polynomial $f(x,k)$. In the case of n-bit block ciphers, we can consider up to n different black-box polynomial for each round. We study SIMON with $n = 32$ and

$m = 4$. Hence, we consider up to 64 different black box polynomials and we consider cubes with highest possible cuberank (Definition 45).

In our experiment, we fixed a secret key and considered 10 round cubes of cuberank 3 and 4. Then, we construct the polynomial system and run ElimLin. Table 4.18 and Table 4.19 shows how many polynomials in the key variables were recovered for each cube.
In the next step, we found 20 cubes of 2^{21} plaintexts of rank 1. We give these cubes in Table 4.20. As our implementation of ElimLin is not suitable for systems of 2^{21} plaintext/ciphertext pairs, we selected subcubes of 2^5 and tested the preformance of ElimLin against 13 rounds as in Section 4.5.2. Even though these subcubes had cuberank 64, we did not recover any polynomial in key variables. This phenomena is still an open problem.

$C_{m,t}$ system of 2^5 samples		key polynomials recovered
m	t	
0x0000001400000051	0x91E961A895DDFFAA	8
0x00000028000000A2	0x8F7049C053D5CE00	0
0x0000005000000144	0x4AEC0722CA7CD632	8
0x0000028000000A20	0x5DF80042CD90648F	9
0x0000028000000A20	0xC022AC2273E1818B	9
0x0000050000001440	0x1BB44000FFA88283	4
0x0000050000001440	0xE09C20551A6F0BB6	7
0x00000A0000002880	0x29F2E0A84802D018	8
0x00000A0000002880	0x6CBA814A4D784111	8
0x0000140000005100	0xAEE108C463EDA072	7
0x0000140000005100	0xF8C140111876A869	8
0x000028000000A200	0x92A0520276DD08EE	7
0x000028000000A200	0xACC006A4FB4E15E0	9
0x000028000000A200	0xF4491689436808E3	6
0x0000A00000028800	0x25A64EA2686516B0	8
0x0000A00000028800	0xA20456140D1077B4	8
0x0001400000051000	0xE91830236128AA78	8
0x00028000000A2000	0xAA192A4B24B483AF	7
0x0005000000144000	0x8D0A65161C88280F	7
0x000A000000288000	0x0A150A7266176D7B	7
0x000A000000288000	0x10558985C513531E	8
0x000A000000288000	0x12152B9F06875130	7
0x000A000000288000	0x4A41CA6F9B5173E7	8
0x000A000000288000	0x8021827B80554735	8
0x0014000000510000	0x8461C1640A087257	9
0x0014000000510000	0xA400D49ABE8A0E33	7
0x0014000000510000	0xB4629121E684C6F6	8
0x0028000000A20000	0x621124BAB25CF6A5	9
0x0050000001440000	0x058F5B37E2915BCF	8
0x0050000001440000	0x12AE464784A89D89	7
0x0050000001440000	0xC5A74459282A67DA	9
0x0500000014400000	0x683089C8CA1E1FD8	9
0x1400000051000000	0x6245E094A44B67EE	7
0x28000000A2000000	0x5286BB0911464FF6	8
0x4000000110000005	0xA04D0812444FC0DA	7
0x5000000044000001	0x285CEDC7A0CD7C14	8
0x5000000044000001	0x2C4C76C6B01A63D2	7
0x800000022000000A	0x70E79C3D8B02C534	7

Table 4.18: results for cuberank 3 for 10 round SIMON, attacked 13 rounds

$C_{m,t}$ system of 2^5 samples		key polynomials recovered
m	t	
0x0000001400000054	0x846870006BF32D29	8
0x00000028000000A8	0x071C214742C05A06	7
0x0000005000000150	0x1EED04016076E803	8
0x0000005000000150	0x5EF38680EF07120B	8
0x0000005000000150	0x5EF6C68922707428	6
0x0000014000000540	0x42EC563DAF599011	6
0x000028000000A800	0xC7A18482AAE8160F	10
0x0000500000015000	0x1A040F0501788339	10
0x0000500000015000	0x7105054FA0348A1F	8
0x0000A0000002A000	0x4A8E1419E3D002F5	6
0x0005000000150000	0xC1426136AEE2397A	10
0x000A0000002A0000	0x42C18178B0857A68	9
0x0014000000540000	0xC041400FD3838E7C	0
0x0028000000A80000	0x885721591251F364	9
0x0050000001500000	0x51AF1118C02D8689	11
0x0050000001500000	0x82A60186AA8E321E	11
0x0050000001500000	0xD0A00F2FDE80FEC4	9
0x0140000005400000	0x5C8909B508B02190	10
0x028000000A800000	0xAC104EF0604D3456	7
0x0500000015000000	0x3A8159DDEA8B307E	9
0x28000000A8000000	0x552E6520033B1F98	11
0x5000000050000001	0x234ED3D6A7DDF6E4	7
0x800000028000000A	0x10A948BC1D9FF684	7
0x800000028000000A	0x7A3C3E184CD06DE0	11
0xA0000000A0000002	0x0AAC7AA5101927F8	10

Table 4.19: results for cuberank 4 for 10 round SIMON, attacked 13 rounds

$C_{m,t}$ system of 2^{21} samples	
m	t
0x2202116805826bb1	0x8c244810b0699448
0x4e810001bb031b06	0x0142630840e0a480
0x46810011bb034b06	0x8062be4044c02009
0x031835000035f852	0x68e6027625000188
0xb8095149c0018586	0x02c60000234a7810
0x0a820ce2284c3281	0x841920100510017a
0x43a40081e6dc0111	0x9010df5e0002a2ce
0x82202c320340b0ec	0x50d18005cc264602
0x0463b2420187e042	0x030804bd40501f00
0x80011b0640180edf	0x570c6461a6a13120
0x22116b0004e1b142	0x1508949e68040e18
0x0870885032cb3018	0x33881725002002c0
0x8d000006be1402dd	0x0290c0d041625500
0x40c451d452118650	0x06010228a1e2612c
0x18e8122065f88200	0x631164169801355e
0x15045e8024596e00	0x404881474b8491c2
0x0804906a00a4e1b5	0x11c361119b5a0e00
0x01201cea012c38ac	0x0ed30211c0914452
0x840a5068481061f4	0x70f5218121019a01
0xba00889c6c021038	0x408b35001029ea02

Table 4.20: Cubes of 2^{21} samples against 11 round SIMON

5 Proning Techniques

In this chapter, we develop a technique called Proning. The technique is designed to derive low degree polynomials which belong to the ideal $\langle \mathcal{S}_{\chi,\gamma,\star} \rangle$ but which are not given in the description of the polynomial system $\mathcal{S}_{\chi,\gamma,\star}$. We use these low degree polynomials together with polynomials from $\mathcal{S}_{\chi,\gamma,\star}$ as an input to standard algebraic techniques such as ElimLin, mXL/F4 and SAT solvers. In the case of ElimLin, we observed increased performance on such larger system and similar results are expected for mXL/F4 and SAT.

Originally, the Proning technique was designed as an extension of cube attacks to find additional polynomials which were not found by ElimLin. Then, we extended the technique to find polynomials which would speed-up mXL/F4/SAT computation.

Our technique consist of two steps. In the first step called Universal Proning, we find universal polynomials. These are polynomials which are "satisfied" for all values of the secret key. Then in the second step, we map these universal polynomials onto polynomials from the ideal spanned by the polynomial system $\mathcal{S}_{\chi,\gamma,\star}$.

We give a memory efficient method to perform Universal Proning and we introduce a heuristic which reduces the time complexity of Universal Proning. Unlike standard techniques such as mXL/F4, Universal Proning allows us to focus on a small set of polynomials and look for universal polynomials in this set. Then, we select this set so that after the second step, we obtain so called "mutant" polynomials that would be found by the first iteration of mXL. This method is called Mutant Proning. Finally, we show how to recover mutant polynomials from subsequent iteration using a new algorithm called Iterative Proning.

In Section 5.1, we relate our technique to a well-known method for recovering S-box

equations. In Section 5.2, we introduce our technique to recover universal polynomials and give related proofs. In Section 5.2.3, we give a more efficient algorithm to recover universal polynomials with respect to memory complexity, and in Section 5.2.4, we give a heuristic method to recover universal polynomials more efficiently. In Section 5.3, we study the transformation of universal into nonuniversal polynomials and we develop an algorithm called Mutant Proning which recovers mutants from mXL. In this case, our results show that a heuristic approach can be used to recover mutants. We tested this on 75-rounds of KATAN32. However, mutants found by this algorithm form only a small subset of mutants found by mXL. Hence, we use extend Mutant Proning and develop an algorithm called Iterative Proning which mimics working of mXL.
The results in this chapter are an extension of the work published at ACISP14 [SSV14].

5.1 Dual View on Polynomial System of Cipher

In this section, we give another approach to find the system of equation $\mathcal{S}$ using the same technique used for recovering a polynomial system representing an S-box. We build a polynomial system representing the entire cipher based on the same approach. However, we consider a fixed set of samples in order to "customize" the polynomial system for a given problem. Hence, the encryption and decryption algorithm is transformed into a black box which is queried with a key and it outputs values of state bits from encryption and decryption. Due to the Kerckhoffs principle, we know the algorithm for encryption and decryption and hence, we can apply this technique to any cipher for which the algorithm is public. We formalize this in Section 5.2.

Recovering ANF of S-box

We demonstrate the method in Table 5.1. We recover an algebraic expression of an S-Box defined by a non-linear cycle (07532461). The S-Box satisfies the following equations. For an input (x_0,x_1,x_2) and output $(y_2,y_1,y_0) = S(x_2,x_1,x_0)$. These can be derived as follows. We consider input and output bits x_i and y_j which are represented by the first 6 rows and additionally, we consider monomials among input bits x_i. We build the matrix M as shown in Table 5.1 and we find the kernel of the matrix M.

$$\mathsf{ker}(M) = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}$$

This is a maximum rank matrix such that $\mathsf{ker}(M)M = 0$. Its rows are equations which hold for any input $x_2x_1x_0$ and therefore they describe the S-box in Algebraic Normal

M

	$0 \to 7$	$1 \to 0$	$2 \to 4$	$4 \to 6$	$3 \to 2$	$6 \to 1$	$5 \to 3$	$7 \to 5$
	$000_2 \to 111_2$	$001_2 \to 000_2$	$010_2 \to 100_2$	$100_2 \to 110_2$	$011_2 \to 010_2$	$110_2 \to 001_2$	$101_2 \to 011_2$	$111_2 \to 101_2$
1	1	1	1	1	1	1	1	1
y_2	1	0	1	1	0	0	0	1
y_1	1	0	0	1	1	0	1	0
y_0	1	0	0	0	0	1	1	1
x_2	0	0	0	1	0	1	1	1
x_1	0	0	1	0	1	1	0	1
x_0	0	1	0	0	1	0	1	1
x_0x_1	0	0	0	0	1	0	0	1
x_1x_2	0	0	0	0	0	1	0	1
x_0x_2	0	0	0	0	0	0	1	1
$x_0x_1x_2$	0	0	0	0	0	0	0	1

Table 5.1: Recovering Algebraic Description of an S-Box (07532461).

Form (ANF).

$$\begin{cases} 0 = 1 + y_2 + x_0 + x_1 * x_2 \\ 0 = 1 + y_1 + x_1 + x_0 + x_0 * x_2 \\ 0 = 1 + y_0 + x_2 + x_1 + x_0 + x_0 * x_1 \end{cases}$$

Hence, the ANF of an S-Box can be recovered by computing the nullspace of matrix like in Table 5.1. The Universal Proning Technique in Section 5.2 aims to recover the ANF of the encryption and the decryption functions using the same approach.

5.2 Universal Proning

In this section, we build on the concept of open-ended systems introduced in Notation 12. We introduce a universal polynomial as a polynomial which belongs to an ideal spanned by open-ended system. Alternatively, we show that a universal polynomial is a polynomial which belongs to an ideal spanned by the polynomial system independently of the value of the secret key. Hence, the name universal. Intuitively, we can see that a universal polynomial by itself cannot help to recover the secret key.

We build the two open-ended ideals over different sets of variables. The Universal Proning is a method to generate these ideals as a kernel of a well-chosen function. Then in nonuniversal proning, we study these systems when we set a common name for a pair of corresponding variables from these open-ended ideals. From the perspective of open-ended ideals, this substitution leads to a recovery of the secret key and mutant polynomials.
In Definition 46, we define the ideal $\mathcal{P}_\chi$ of open-ended system where we set plaintexts to χ. Similarly, we define the ideal $\mathcal{C}_\gamma$ of open-ended system where we set ciphertexts to γ. In Definition 47, we define the ideal $\mathcal{B}_{\chi,\gamma}$ obtained from both of these open-ended

ideals. Later in Definition 48, we restrict the keyspace to $\mathcal{K}$ and hence, we enlarge the ideal $\mathcal{B}_{\chi,\gamma}$ to $\mathcal{B}^{\mathcal{K}}_{\chi,\gamma}$.

5.2.1 Universal Proning: Overview

In this section, we introduce universal polynomials. Informally, a universal polynomial evaluates to zero for all choices of encryption key κ. We give a formal definition in Definition 47. The Universal Proning is a method to find all universal polynomial in a set of polynomials. The technique is related to hybrid approach in Gröbner basis computation [BFP09, BFP12]. In [BFSS13], the authors specialize some variables of the system to all possible values which is similar to our approach for Universal Proning where we specialize the variables in V_K.

Definition 46. *We define*

$$\mathcal{P}_\chi = \bigcap_{\kappa \in \mathbf{F}_2^{kln}} \langle \mathcal{S}_{\chi,\star,\kappa} \rangle_{\mathbf{F}_2[V]}$$
$$\mathcal{C}_\gamma = \bigcap_{\kappa \in \mathbf{F}_2^{kln}} \langle \mathcal{S}_{\star,\gamma,\kappa} \rangle_{\mathbf{F}_2[V]}$$

Intuitively, we can see that a universal polynomial cannot help to recover the secret key, but it helps to simplify the polynomial system. We combine these two notions in Definition 47.

In Lemma 55, we show that $\langle \mathcal{S}_{\chi,\star,\star} \rangle_{\mathbf{F}_2[V]} = \mathcal{P}_\chi$ and $\langle \mathcal{S}_{\star,\gamma,\star} \rangle_{\mathbf{F}_2[V]} = \mathcal{C}_\gamma$. We define an ideal which is spanned by two open-ended systems where the relation between plaintext and ciphertext is discarded.

Definition 47. *Let consider the bijective function Dup from V to V' as in Definition 18. Then, for every $\kappa \in \mathbf{F}_2^{kln}$, we consider systems $\mathcal{S}_{\chi,\star,\kappa} \subset \mathbf{F}_2[V]$ and $\mathit{Dup}\left(\mathcal{S}_{\star,\gamma,\kappa}\right) \subset \mathbf{F}_2[V']$, where we rename the variables. We consider the ring $\mathbf{F}_2[V,V']$ and define $\mathcal{B}_{\chi,\gamma} \subset \mathbf{F}_2[V,V']$*

$$\mathcal{B}_{\chi,\gamma} = \bigcap_{\kappa \in \mathbf{F}_2^{kln}} \left(\langle \mathcal{S}_{\chi,\star,\kappa} \rangle_{\mathbf{F}_2[V,\mathit{Dup}(V)]} + \langle \mathit{Dup}\left(\mathcal{S}_{\star,\gamma,\kappa}\right) \rangle_{\mathbf{F}_2[V,\mathit{Dup}(V)]} \right)$$

We say q is universal iff $q \in \mathcal{B}_{\chi,\gamma}$. Otherwise, we say q is nonuniversal.

The idea for duplication of variables and building a system such as $\mathcal{B}_{\chi,\gamma}$ was independently developed in [RM12]. The concept of universal polynomials is also related to an idea presented by Courtois in [Cou08, slide 118-120]. In Theorem 51, we show how we can construct $\mathcal{B}_{\chi,\gamma}$ algorithmically. However, this will be computationally rather expensive and hence in Definition 48, we define for $\mathcal{K} \subseteq \mathbf{F}_2^{\mathsf{kln}}$ an ideal $\mathcal{B}_{\chi,\gamma}^{\mathcal{K}}$ which can be seen as an approximation of $\mathcal{B}_{\chi,\gamma}$.

Definition 48. *For $\mathcal{K} \subseteq \mathbf{F}_2^{kln}$, we define*

$$\mathcal{B}_{\chi,\gamma}^{\mathcal{K}} = \bigcap_{\kappa \in \mathcal{K}} \left(\langle \mathcal{S}_{\chi,\star,\kappa} \rangle_{\mathbf{F}_2[V,\mathit{Dup}(V)]} + \left\langle \mathit{Dup}\left(\mathcal{S}_{\star,\gamma,\kappa}\right) \right\rangle_{\mathbf{F}_2[V,\mathit{Dup}(V)]} \right)$$

and we say $\mathcal{K}$ is consistent iff for the value $\kappa \in \mathbf{F}_2^{kln}$ such that $E_\kappa(\chi) = \gamma$, we have $\kappa \in \mathcal{K}$.

Trivially, we have $\mathcal{B}_{\chi,\gamma}^{\mathbf{F}_2^{\mathsf{kln}}} = \mathcal{B}_{\chi,\gamma}$.

Definition 49. *Let us define the function $e_\chi : \mathbf{F}_2[V] \to \mathit{Func}\left(\mathbf{F}_2^{kln}, \mathbf{F}_2\right)$, such that $e_\chi(m)$ is the function mapping κ in $\mathbf{F}_2^{kln}$ to the reduction of the polynomial m modulo[1] $\langle \mathcal{S}_{\chi,\star,\kappa} \rangle$. We further denote for $\mathcal{K} \subseteq \mathbf{F}_2^{kln}$ the function $e_\chi|_{\mathcal{K}} : \mathbf{F}_2[V] \to \mathit{Func}(\mathcal{K}, \mathbf{F}_2)$ so that $e_\chi|_{\mathcal{K}}(q) = e_\chi(q)|_{\mathcal{K}}$. Similarly, let us define the function $d_\gamma : \mathbf{F}_2[\mathit{Dup}(V)] \to \mathit{Func}\left(\mathbf{F}_2^{kln}, \mathbf{F}_2\right)$, such that $d_\gamma(m)$ is the function mapping κ in $\mathbf{F}_2^{kln}$ to the reduction of the polynomial m modulo $\mathit{Dup}\left(\langle \mathcal{S}_{\star,\gamma,\kappa} \rangle\right)$ and we denote $d_\gamma|_{\mathcal{K}} : \mathbf{F}_2[\mathit{Dup}(V)] \to \mathit{Func}(\mathcal{K}, \mathbf{F}_2)$ so that $d_\gamma|_{\mathcal{K}}(q) = d_\gamma(q)|_{\mathcal{K}}$. Moreover, let us define $f_{\chi,\gamma} : \mathbf{F}_2[V, \mathit{Dup}(V)] \to \mathit{Func}\left(\mathbf{F}_2^{kln}, \mathbf{F}_2\right)$, such that $f_{\chi,\gamma}(m)$ is the function mapping κ in $\mathbf{F}_2^{kln}$ to the reduction of the polynomial m modulo $\left\langle \mathcal{S}_{\chi,\star,\kappa}, \mathit{Dup}\left(\mathcal{S}_{\star,\gamma,\kappa}\right) \right\rangle_{\mathbf{F}_2[V,\mathit{Dup}(V)]}$. We denote $f_{\chi,\gamma}|_{\mathcal{K}} : \mathbf{F}_2[V, \mathit{Dup}(V)] \to \mathit{Func}(\mathcal{K}, \mathbf{F}_2)$ so that $f_{\chi,\gamma}|_{\mathcal{K}}(q) = f_{\chi,\gamma}(q)|_{\mathcal{K}}$. By abuse of notation, we denote $f_{\chi,\gamma}|_{\{\kappa\}}$ as $f_{\chi,\gamma}|_\kappa$.*

We note that reductions of q modulo $\langle \mathcal{S}_{\chi,\star,\kappa} \rangle$, $\langle \mathcal{S}_{\star,\gamma,\kappa} \rangle$ or $\left\langle \mathcal{S}_{\chi,\star,\kappa}, \mathsf{Dup}\left(\mathcal{S}_{\star,\gamma,\kappa}\right) \right\rangle$ are easy as we just have to follow the specifications of the encryption or decryption algorithms to evaluate all variables. Then, we can evaluate q on these variables.

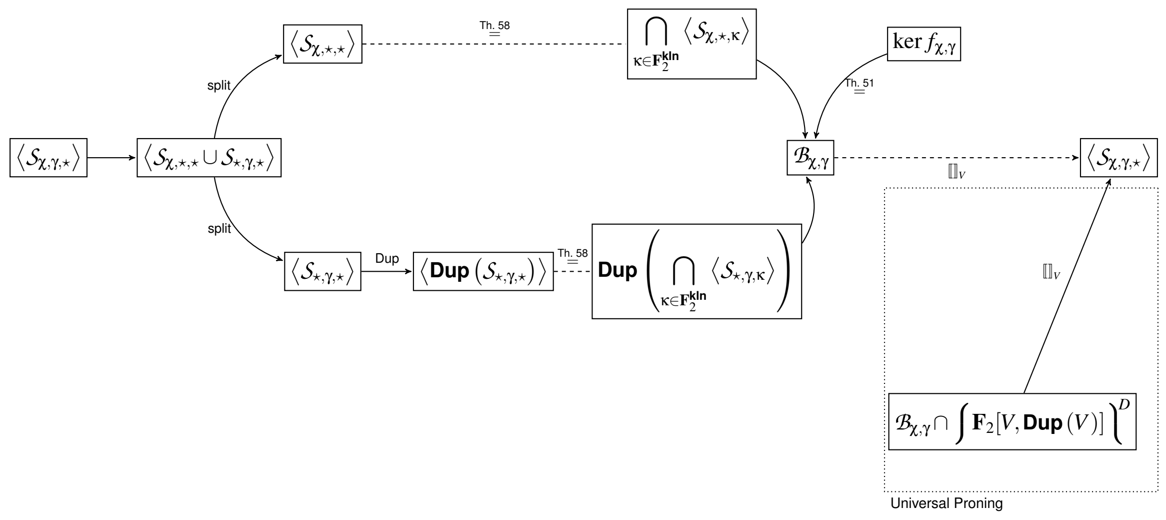
Notation 50. *For $Q \subseteq \mathbf{F}_2[V, \mathit{Dup}(V)]$, we denote*

$$\mathcal{K}_{\chi,\gamma}(Q) = \left\{ \kappa \in \mathbf{F}_2^{kln} \mid \forall q \in Q \; f_{\chi,\gamma}(q)|_\kappa = 0 \right\}.$$

Hence, $\mathcal{K}_{\chi,\gamma}(Q)$ is a restriction of variety of space $\mathbf{F}_2^{|V|}$ to variety of space $\mathbf{F}_2^{kln}$ which represents variables V_K.

For instance $\mathcal{K}_{\chi,\gamma}(\emptyset) = \mathbf{F}_2^{\mathsf{kln}}$, $\mathcal{K}_{\chi,\gamma}\left(\mathcal{S}_{\chi,\star,\star}\right) = \mathbf{F}_2^{\mathsf{kln}}$, $\mathcal{K}_{\chi,\gamma}\left(\mathcal{S}_{\chi,\star,\star} \cup \mathsf{Dup}\left(\mathcal{S}_{\star,\gamma,\star}\right)\right) = \mathbf{F}_2^{\mathsf{kln}}$ and using Assumption 15, we also have $\mathcal{K}_{\chi,\gamma}\left(\mathcal{S}_{\chi,\gamma,\star}\right) = \{\kappa\}$. Moreover using Theorem 51, for any $\mathcal{K} \subseteq \mathbf{F}_2^{\mathsf{kln}}$, we have $\mathcal{K}_{\chi,\gamma}\left(\mathcal{B}_{\chi,\gamma}^{\mathcal{K}}\right) = \mathcal{K}$.

[1]As $\mathcal{S}_{\chi,\star,\kappa}$ is a maximal ideal the reduction modulo it is in $\mathbf{F}_2$. Equivalently, the ideal reduction is equivalent to the evaluation of the polynomial.



$\langle \mathcal{S}_{\chi,\star,\star}\rangle$
Th. 58
$\bigcap_{\kappa\in\mathbf{F}_2^{kln}} \langle \mathcal{S}_{\chi,\star,\kappa}\rangle$
$\ker f_{\chi,\gamma}$
split
Th. 51
$\langle \mathcal{S}_{\chi,\gamma,\star}\rangle$
$\langle \mathcal{S}_{\chi,\star,\star} \cup \mathcal{S}_{\star,\gamma,\star}\rangle$
$\mathcal{B}_{\chi,\gamma}$
$[\![]\!]_V$
$\langle \mathcal{S}_{\chi,\gamma,\star}\rangle$
split
$\langle \mathcal{S}_{\star,\gamma,\star}\rangle$
Dup
$\langle \mathbf{Dup}(\mathcal{S}_{\star,\gamma,\star})\rangle$
Th. 58
$\mathbf{Dup}\left(\bigcap_{\kappa\in\mathbf{F}_2^{kln}} \langle \mathcal{S}_{\star,\gamma,\kappa}\rangle\right)$
$[\![]\!]_V$
$\mathcal{B}_{\chi,\gamma} \cap \int \mathbf{F}_2[V, \mathbf{Dup}(V)] \Big\rceil^D$
Universal Proning

Figure 5.1: Solving system by Universal Proning

We first split the system $\mathcal{S}_{\chi,\gamma,\star}$ into two open-systems $\mathcal{S}_{\chi,\star,\star}$ and $\mathcal{S}_{\star,\gamma,\star}$. Then, we consider two ideals of universal polynomials $\langle \mathcal{S}_{\chi,\star,\star}\rangle_{\mathbf{F}_2[V]}$ and $\langle \mathcal{S}_{\star,\gamma,\star}\rangle_{\mathbf{F}_2[V]}$ and we use Theorem 55 to show their relation to ideals $\mathcal{P}_\chi$ and $\mathcal{C}_\gamma$. Furthermore, ideals $\mathcal{P}_\chi$ and $\mathcal{C}_\gamma$ can be generated as a kernel of appropriate functions (Theorem 51). The change of variables (Dup) is necessary for that. In Definition 47, we unified these two ideals into the ideal $\mathcal{B}_{\chi,\gamma}$. Then in Lemma 57, we show that $\langle \mathcal{S}_{\chi,\gamma,\star}\rangle = [\![\mathcal{B}_{\chi,\gamma}]\!]_V$, which allows us to explore $\langle \mathcal{S}_{\chi,\gamma,\star}\rangle$ using Universal Proning and the homomorphism $[\![]\!]_V$.

Let us now consider the kernel of functions $e_\chi(m)$, resp. $d_\gamma(m)$ resp. $f_{\chi,\gamma}(m)$. It is a set of polynomials which are zero for all keys and plaintext χ, resp. γ resp. a pair of both (χ,γ).

Theorem 51. *For $\mathcal{K} \subseteq \mathbf{F}_2^{kln}$, we have*

$$\mathcal{P}_\chi = \mathit{ker}(e_\chi)\,,\ \mathit{Dup}(\mathcal{C}_\gamma) = \mathit{ker}(d_\gamma)\,,\ \mathcal{B}_{\chi,\gamma} = \mathit{ker}(f_{\chi,\gamma})\,,\ \mathcal{B}^{\mathcal{K}}_{\chi,\gamma} = \mathit{ker}\left(f_{\chi,\gamma}\Big|_{\mathcal{K}}\right)$$

Proof. *The functions $e_\chi(m)$, $d_\gamma(m)$ and $f_{\chi,\gamma}(m)$ are linear. Hence,*

$$\begin{aligned} m \in \mathcal{P}_\chi \iff & \quad (\forall\kappa:\ m \in \langle \mathcal{S}_{\chi,\star,\kappa}\rangle) \iff & e_\chi(m) = 0 \iff & \quad m \in \mathit{ker}(e_\chi) \\ m \in \mathcal{C}_\gamma \iff & \quad (\forall\kappa:\ m \in \langle \mathcal{S}_{\star,\gamma,\kappa}\rangle) \iff & d_\gamma(m) = 0 \iff & \quad m \in \mathit{ker}(d_\gamma) \end{aligned}$$

$$
\begin{aligned}
m \in \mathcal{B}_{\chi,\gamma} \iff & \quad (\forall \kappa :\ m \in (\langle \mathcal{S}_{\chi,\star,\kappa} \rangle + \langle \mathit{Dup}(\mathcal{S}_{\star,\gamma,\kappa}) \rangle)) \iff & f_{\chi,\gamma}(m) = 0 \\
& \iff & m \in \mathit{ker}(f_{\chi,\gamma}) \\
m \in \mathcal{B}^{\mathcal{K}}_{\chi,\gamma} \iff & \quad (\forall \kappa \in \mathcal{K} :\ m \in (\langle \mathcal{S}_{\chi,\star,\kappa} \rangle + \langle \mathit{Dup}(\mathcal{S}_{\star,\gamma,\kappa}) \rangle)) \iff & f_{\chi,\gamma}\Big|_{\mathcal{K}}(m) = 0 \\
& \iff & m \in \mathit{ker}\left(f_{\chi,\gamma}\Big|_{\mathcal{K}} \right)
\end{aligned}
$$

$\square$

Usage of Universal Proning. So far, we gave definitions which allow us to formalize the algorithm to find universal polynomials. We give this algorithm in Algorithm 6. Universal Proning is a technique to find an ideal spanned by the polynomial system $\mathcal{S}_{\chi,\gamma,\star}$ which is shown in Section 5.2.2. The advantage of Universal Proning will become apparent when we consider a restriction to a subvectorspace R of $\mathbf{F}_2[V, \mathsf{Dup}(V)]$, for instance vectorspace of polynomials of degree up to some $D \in \mathbb{N}$. This vector space can be selected based on various cryptanalytic techniques. In our experiments, we use cube attacks to find a small set of variables W such that $\mathbb{B}[W]$ contains a universal polynomial. Alternatively, we expect that linear or differential cryptanalysis would be good candidates for such selection. The intuition behind Universal Proning is the following (cf. Figure 5.1):

- We split the system $\mathcal{S}_{\chi,\gamma,\star}$ into two open-ended systems: $\mathcal{S}_{\chi,\star,\star}$ and $\mathcal{S}_{\star,\gamma,\star}$.
- We rename variables in $\mathcal{S}_{\star,\gamma,\star}$. This does not change the system itself, but it allows us to drop all relations between plaintexts and ciphertexts.
- We consider sum of ideals spanned by these systems and we obtain

$$\langle \mathcal{S}_{\chi,\star,\star} \rangle_{\mathbf{F}_2[V,\mathsf{Dup}(V)]} + \langle \mathsf{Dup}\,(\mathcal{S}_{\star,\gamma,\star}) \rangle_{\mathbf{F}_2[V,\mathsf{Dup}(V)]} = \mathcal{B}_{\chi,\gamma}.$$

- As there are no relations among plaintexts and ciphertexts in such ideal, this is an ideal of universal polynomials.
- Furthermore, the ideal $\mathcal{B}_{\chi,\gamma}$ can be computed as a kernel of the function $f_{\chi,\gamma}$ defined in Definition 49. This is shown in Theorem 51.
- We have $\langle \mathcal{S}_{\chi,\gamma,\star} \rangle_{\mathbf{F}_2[V]} = [\![\mathcal{B}_{\chi,\gamma}]\!]_V$ as shown in Theorem 58 and hence, Universal Proning allows us to generate ideal $\langle \mathcal{S}_{\chi,\gamma,\star} \rangle$.
- We consider various selections of R to obtain "interesting" polynomials of ideal $\langle \mathcal{S}_{\chi,\gamma,\star} \rangle$. I.e, we study the action of operation $[\![]\!]_V$ and characterize polynomials

which lead to a nonuniversal polynomial. This method will be called Mutant Proning and it is introduced in Section 5.3.

Algorithm 6 Universal Proning

```
Input: $\chi, \gamma$, $\mathcal{K} \subseteq \mathbf{F}_2^{\mathsf{kln}}$, vector space $R \subseteq \mathbf{F}_2[V, \mathsf{Dup}(V)]$
Output: $\mathsf{ker}\left( f_{\chi,\gamma}\Big|_{\mathcal{K}} \right) \cap R$
1: select a linear basis $B$ of $R$
2: $M \leftarrow$ matrix of dimension $|B| \times |\mathcal{K}|$
3: for all $b \in B$ do
4:     for all $\kappa \in \mathcal{K}$ do
5:         $M_{b,\kappa} \leftarrow f_{\chi,\gamma}|_{\kappa}(b)$
6:     end for
7: end for
8: find $N$ of maximal size with full rank such that $NM = 0$ using Gauss elimination
9: return the set of all $\sum_{b \in B} N_{i,b} b$ for all $i$
```

5.2.2 Universal Proning: Details

We now extend the technique from Section 5.1 to build a polynomial system for a cipher. However, in this case, we intend to recover the ANF of the encryption function only for specified plaintexts (χ) and the ANF of the decryption function for specified ciphertexts (γ). We remind that each variable of the system corresponds to some sample (index p), see Notation 11. Hence, each variable can be seen as a function of a secret key and hence, each polynomial of a system of equations $\mathcal{S}_{\chi,\star,\star}$ or $\mathcal{S}_{\star,\gamma,\star}$ can be seen as a function of a secret key (cf. Definition 49).

Lemma 52.

$$\langle \mathcal{S}_{\chi,\gamma,\star} \rangle_{\mathbf{F}_2[V]} = \bigcap_{\kappa \in \mathbf{F}_2^{kln}} \langle \mathcal{S}_{\chi,\gamma,\kappa} \rangle_{\mathbf{F}_2[V]}$$

Proof. *We have*

$$\langle \mathcal{S}_{\chi,\gamma,\kappa} \rangle_{\mathbf{F}_2[V]} = \begin{cases} \mathbf{F}_2[V] & \text{if } \kappa \text{ is an incorrect key.} \\ \langle \mathcal{S}_{\chi,\gamma,\star} \rangle_{\mathbf{F}_2[V]} & \text{if } \kappa \text{ is the correct key due to Assumption 15.} \end{cases}$$

Hence,

$$\bigcap_{\kappa \in \mathbf{F}_2^{kln}} \langle \mathcal{S}_{\chi,\gamma,\kappa} \rangle_{\mathbf{F}_2[V]} = \mathbf{F}_2[V] \cap \langle \mathcal{S}_{\chi,\gamma,\star} \rangle_{\mathbf{F}_2[V]} = \langle \mathcal{S}_{\chi,\gamma,\star} \rangle_{\mathbf{F}_2[V]} .$$

□

Lemma 53. *Let K be a ring and $W \subseteq V$. Then for $G \subseteq K[W]$, we have*

$$\langle G\rangle_{K[V]} \cap K[W] = \langle G\rangle_{K[W]}.$$

Proof. *We write $R = K[W]$ and for $W' = V \setminus W$ we obtain $K[V] = R[W']$. For $G \subseteq R$, we have*

$$\langle G\rangle_{K[V]} \cap K[W] = \langle G\rangle_{R[W']} \cap R.$$

So we have to prove $\langle G\rangle_{R[W']} \cap R = \langle G\rangle_R$. I.e, we reduce to the $W = \emptyset$ case.

$$\langle G\rangle_R \subseteq \langle G\rangle_{R[W']} \cap R$$

is trivial. If $h \in \langle G\rangle_{R[W']} \cap R$, we can write $h = \sum_{g\in G} h_g g$ with $h_g \in R[W']$. For each monomial $m \in R[W']$, $m \neq 1$, the coefficient of m in this equation gives $0 = \sum_g h_{g,m} g$ where $h_{g,m}$ is the coefficient of h_g in m. If we write $h'_g = h_{g,1}$, we have $\sum_{g\in G} h'_g g = h$. So, $h \in \langle G\rangle_R$. □

Theorem 54. *Let $\mathcal{J} \subseteq \mathbf{F}_2[V]$ be an ideal such that $\mathsf{FieldEq}[V] \subseteq \mathcal{J}$ and such that for every $\kappa \in \mathbf{F}_2^{kln}$ the ideal $\mathcal{J}_\kappa = \mathcal{J} + \langle k_i - \kappa_i : i \in [1, kln]\rangle$ is maximal. Then, $\mathcal{J} = \bigcap_{\kappa\in\mathbf{F}_2^{kln}} \mathcal{J}_\kappa$.*

Proof. *In what follows, we denote $E = \mathbf{F}_2[V]/\mathsf{FieldEq}[V]$ and consider ideals over $E[V]$ in addition to ideals over $\mathbf{F}_2[V]$. We denote by $\langle \mathcal{S}\rangle_E$ the ideal over $E[V]$ spanned by $\mathcal{S} \subseteq \mathbf{F}_2[V]$. Let I be an ideal of the affine algebra $E[V]$. Let us consider the set*

$$\mathcal{H}_I = \{\mathcal{T} : I \subseteq \mathcal{T} \text{ and } \mathcal{T} \text{ is a maximal ideal over } E[V]\}.$$

The Hilbert Nullstellensatz [Bos12, Chapter 3, Corollary 6] (N) states that for an ideal $I \subseteq E[V]$. Then, $\sqrt{I} = \bigcap_{\mathcal{T}\in\mathcal{H}_I} \mathcal{T}$. We define sets $\mathcal{H}_{\mathcal{J}}^{\overline{K}} = \{\langle \mathcal{J}_\kappa\rangle_E : \kappa \in E^{kln}, 1 \notin \langle \mathcal{J}_\kappa\rangle_E\}$ and $\mathcal{H}_{\mathcal{J}}^{K} = \{\langle \mathcal{J}_\kappa\rangle_E : \kappa \in \mathbf{F}_2^{kln}\}$.

i) We express E as a ring over $\mathbf{F}_2$ (see [DF04, Theorem 6, page 517] and [DF04, chapter 13.4, page 536]) and using Lemma 53 we obtain

$$\langle \mathcal{J}_\kappa\rangle_E \cap \mathbf{F}_2[V] = \mathcal{J}_\kappa$$

ii) Moreover, $\forall v \in V, v^2 - v \in \langle \mathcal{J}\rangle_E$ we have

$$\kappa \in E^{kln} \setminus \mathbf{F}_2^{kln} \Rightarrow 1 \in \langle \mathcal{J}_\kappa\rangle_E.$$

Therefore,

$$\left\{\langle \mathcal{J}_\kappa\rangle_E : \kappa \in \mathbf{F}_2^{kln}\right\} = \left\{\langle \mathcal{J}_\kappa\rangle_E : \kappa \in E^{kln}, 1 \notin \mathcal{J}_\kappa\right\}.$$

Hence, we have

$$\mathcal{H}_{\mathcal{J}}^{\overline{K}} = \mathcal{H}_{\mathcal{J}}^{K}.$$

iii) *Using Lemma 16, we obtain that every maximal ideal $\mathcal{T} \in \mathcal{H}_I$ corresponds to an affine point in $\mathbf{F}_2^{|V|}$ and hence, it corresponds to some key $\kappa \in E[V]$. We know 1 cannot be in a maximal ideal and therefore $\mathcal{T} \in \mathcal{H}_{\langle \mathcal{J}\rangle_E}$ implies $\mathcal{T} = \langle \mathcal{J}_\kappa\rangle_E$ for some $\kappa \in E^{kln}$ such that $1 \notin \langle \mathcal{J}_\kappa\rangle_E$. Thus, $\mathcal{T} \in \mathcal{H}_{\mathcal{J}}^{\overline{K}}$:*

$$\mathcal{H}_{\langle \mathcal{J}\rangle_E} \subseteq \mathcal{H}_{\mathcal{J}}^{\overline{K}}.$$

Then, for each $\kappa \in E^{kln}$ such that $1 \notin \langle \mathcal{J}\rangle_E$, we have $\kappa \in \mathbf{F}_2^{kln}$ due to ii. So, $\mathcal{J}_\kappa$ is maximal in $\mathbf{F}_2[V]$. So $\langle \mathcal{J}_\kappa\rangle_E$ is maximal in $E[V]$. So

$$\mathcal{H}_{\langle \mathcal{J}\rangle_E} \supseteq \mathcal{H}_{\mathcal{J}}^{\overline{K}}.$$

iv) *By definition of a radical ideal, $\sqrt{\mathcal{J}} = \left\{q \in \mathbf{F}_2[V], \exists i\ q^i \in \mathcal{J}\right\}$. As $\forall v \in V, v^2 - v \in \mathcal{J}$ we have $q^i - q \in \mathcal{J}$ for all q. So, $\sqrt{\mathcal{J}} = \mathcal{J}$. With the same reasoning on $\langle \mathcal{J}\rangle_E$, we obtain*

$$\sqrt{\langle \mathcal{J}\rangle_E} = \langle \mathcal{J}\rangle_E.$$

We have

$$\langle \mathcal{J}\rangle_E \overset{iv}{=} \sqrt{\langle \mathcal{J}\rangle_E} \overset{(N)}{=} \bigcap_{\mathcal{T}\in\mathcal{H}_{\langle \mathcal{J}\rangle_E}} \mathcal{T} \overset{iii}{=} \bigcap_{\mathcal{T}\in\mathcal{H}_{\mathcal{J}}^{\overline{K}}} \mathcal{T} \overset{ii}{=} \bigcap_{\mathcal{T}\in\mathcal{H}_{\mathcal{J}}^{K}} \mathcal{T} \overset{\text{def}\ \mathcal{H}_{\mathcal{J}}^{K}}{=} \bigcap_{\kappa\in\mathbf{F}_2^{kln}} \langle \mathcal{J}_\kappa\rangle_E. \tag{5.1}$$

Finally, we have

$$\bigcap_{\kappa\in\mathbf{F}_2^{kln}} \mathcal{J}_\kappa \overset{i}{=} \left(\bigcap_{\kappa\in\mathbf{F}_2^{kln}} \langle \mathcal{J}_\kappa\rangle_E\right) \cap \mathbf{F}_2[V] \overset{(5.1)}{=} \langle \mathcal{J}\rangle_E \cap \mathbf{F}_2[V] \overset{i}{=} \mathcal{J}.$$

□

Lemma 55. *We have $\left\langle \mathcal{S}_{\chi,\star,\star}\right\rangle_{\mathbf{F}_2[V]} = \mathcal{P}_\chi$ and $\left\langle \mathcal{S}_{\star,\gamma,\star}\right\rangle_{\mathbf{F}_2[V]} = \mathcal{C}_\gamma$*

Proof.
We get the result directly using Theorem 54 for $\mathcal{J} = \left\langle \mathcal{S}_{\chi,\star,\star}\right\rangle_{\mathbf{F}_2[V]}$ and $\mathcal{J} = \left\langle \mathcal{S}_{\star,\gamma,\star}\right\rangle_{\mathbf{F}_2[V]}$.
□

Lemma 56.

$$\mathcal{B}_{\chi,\gamma} = \langle \mathcal{S}_{\chi,\star,\star} \rangle_{\mathbf{F}_2[V,Dup(V)]} + \langle Dup\left(\mathcal{S}_{\star,\gamma,\star}\right) \rangle_{\mathbf{F}_2[V,Dup(V)]}$$

Proof. *We get the result directly using Theorem 54 for*

$$\mathcal{J} = \langle \mathcal{S}_{\chi,\star,\star} \rangle_{\mathbf{F}_2[V,Dup(V)]} + \langle Dup\left(\mathcal{S}_{\star,\gamma,\star}\right) \rangle_{\mathbf{F}_2[V,Dup(V)]}$$

$\mathcal{J}_\kappa$ is maximal for every $\kappa \in \mathbf{F}_2^{kln}$ since $\mathcal{S}_{\chi,\star,\kappa}$ corresponds to the deterministic encryption and $Dup\left(\mathcal{S}_{\star,\gamma,\kappa}\right)$ corresponds to the deterministic decryption. □

Lemma 57. *We have $\langle \mathcal{S}_{\chi,\gamma,\star} \rangle_{\mathbf{F}_2[V]} = \llbracket \mathcal{B}_{\chi,\gamma} \rrbracket_V$ and for a consistent $\mathcal{K} \subseteq \mathbf{F}_2^{kln}$ we also have*

$$\langle \mathcal{S}_{\chi,\gamma,\star} \rangle_{\mathbf{F}_2[V]} = \left\llbracket \mathcal{B}_{\chi,\gamma}^{\mathcal{K}} \right\rrbracket_V$$

Proof. *We have $q + \underbrace{\langle v + Dup(v) : v \in V \rangle}_{J}$ is the set of all polynomials equal to q modulo J. The only one in $\mathbf{F}_2[V]$ is $\llbracket q \rrbracket_V$. So, $\llbracket q \rrbracket_V = (q+J) \cap \mathbf{F}_2[V]$. So, $\llbracket \mathcal{B}_{\chi,\gamma} \rrbracket_V = \left(\mathcal{B}_{\chi,\gamma} + J\right) \cap \mathbf{F}_2[V]$. Using Lemma 56, we have*

$$\begin{aligned}
\llbracket \mathcal{B}_{\chi,\gamma} \rrbracket_V &= \left(\mathcal{B}_{\chi,\gamma} + J\right) \cap \mathbf{F}_2[V] \\
&\overset{L.\ 56}{=} \left(\langle \mathcal{S}_{\chi,\star,\star} \rangle_{\mathbf{F}_2[V,Dup(V)]} + \langle Dup\left(\mathcal{S}_{\star,\gamma,\star}\right) \rangle_{\mathbf{F}_2[V,Dup(V)]} + J \right) \cap \mathbf{F}_2[V] \\
&= \langle \mathcal{S}_{\chi,\star,\star}, Dup\left(\mathcal{S}_{\star,\gamma,\star}\right), J \rangle_{\mathbf{F}_2[V,Dup(V)]} \cap \mathbf{F}_2[V] \\
&= \langle \mathcal{S}_{\chi,\star,\star}, \mathcal{S}_{\star,\gamma,\star}, J \rangle_{\mathbf{F}_2[V,Dup(V)]} \cap \mathbf{F}_2[V] \\
&= \langle \mathcal{S}_{\chi,\gamma,\star} \rangle
\end{aligned}$$

Similarly, whenever $\kappa \in \mathcal{K}$ we obtain $\left\llbracket \mathcal{B}_{\chi,\gamma}^{\mathcal{K}} \right\rrbracket_V = \langle \mathcal{S}_{\chi,\gamma,\star} \rangle_{\mathbf{F}_2[V]}$. □

Theorem 58. *We have*

- $\mathcal{P}_\chi = \langle \mathcal{S}_{\chi,\star,\star} \rangle_{\mathbf{F}_2[V]}$
- $\mathcal{C}_\gamma = \langle \mathcal{S}_{\star,\gamma,\star} \rangle_{\mathbf{F}_2[V]}$
- $\mathcal{B}_{\chi,\gamma} = \left\langle \langle \mathcal{S}_{\chi,\star,\star} \rangle_{\mathbf{F}_2[V]}, Dup\left(\langle \mathcal{S}_{\star,\gamma,\star} \rangle_{\mathbf{F}_2[V]} \right) \right\rangle_{\mathbf{F}_2[V,Dup(V)]}$
- $\mathcal{B}_{\chi,\gamma} = \langle \mathcal{S}_{\chi,\star,\star} \rangle_{\mathbf{F}_2[V,Dup(V)]} + \langle Dup\left(\mathcal{S}_{\star,\gamma,\star}\right) \rangle_{\mathbf{F}_2[V,Dup(V)]}$

Furthermore, $\langle \mathcal{S}_{\chi,\gamma,\star} \rangle_{\mathbf{F}_2[V]} = \llbracket \mathcal{B}_{\chi,\gamma} \rrbracket_V$.

Proof. *Follows directly from Lemma 55, Lemma 56 and Lemma 57.* □

Following the definition, the ideals $\mathcal{P}_\chi$, $\mathcal{C}_\gamma$ and $\mathcal{B}_{\chi,\gamma}$ are ideals of all universal polynomials for given set of plaintexts χ, given set of ciphertexts γ and given set of both plaintexts χ and ciphertexts γ.

5.2.3 Fast Universal Proning

In Section 5.2.1, we proposed Algorithm 6 for building a polynomial system. We now consider a linearly independent set $B \subseteq R$ (step 1 in Algorithm 7). Algorithmically, we compute $\mathsf{ker}\left(f_{\chi,\gamma}|_{\mathcal{K}}\right) \cap \mathbf{linspan}(B)$ as a left kernel of a boolean matrix M of dimension $\dim B \times |\mathcal{K}|$. We fill the matrix M as follows: for $b \in B$ and $\kappa \in \mathcal{K}$ we set $M_{b,\kappa} \leftarrow f_{\chi,\gamma}|_{\kappa}(b)$. We define $\mathsf{ker}(M) := \{v; vM = 0\}$ and we have $\mathsf{ker}\left(f_{\chi,\gamma}|_{\mathcal{K}}\right) \simeq \mathsf{ker}(M)$. For the purpose of this section, we assume[1] $\dim(R) \ll |\mathcal{K}|$ and we show that the computation of $\mathsf{ker}(M)$ can be done more efficiently than the straightforward approach given in Algorithm 6 and it can be distributed among multiple threads running on different machines.

Similarly as in Algorithm 6, we consider a matrix M of dimension $|B| \times |\mathcal{K}|$. In our improvement (Algorithm 7), we consider $t = \frac{|\mathcal{K}|}{|B|}$ and we consider t submatrices M^i where M^i for $i \in [1,t]$ has dimension $|B| \times |B|$. Then, we compute $\mathsf{ker}(M) = \bigcap_i \mathsf{ker}\left(M^i\right)$.

Algorithm 7 Memory Efficient Implementation of $\mathcal{B}_{\chi,\gamma}^{\mathcal{K}}$

```
Input: $\chi, \gamma$, a set of keys $\mathcal{K}$, vector space $R \subseteq \mathbf{F}_2[V, \mathsf{Dup}(V)]$
Output: $\mathcal{B}_{\chi,\gamma}^{\mathcal{K}} \cap R$
 1: select linear basis $B$ of $R$
 2: $N \leftarrow$ identity matrix of dimension $|B| \times |B|$
 3: $t \leftarrow \frac{|\mathcal{K}|}{|B|}$
 4: for $i \in [1,t]$ do
 5:     $M^i \leftarrow$ matrix of dimension $|B| \times |B|$
 6:     $\mathcal{K}_i \leftarrow$ subset of $\mathcal{K}$ with keys of indices $[i|B|, (i+1)|B| - 1]$
 7:     for all $b \in B$ do
 8:         for all $\kappa \in \mathcal{K}_i$ do
 9:             $M^i_{b,\kappa} \leftarrow f_{\chi,\gamma}|_{\kappa}(b)$
10:         end for
11:         $N^i \leftarrow \mathsf{ker}\left(M^i\right)$ using Gauss elimination
12:         $N \leftarrow N \cap N^i$ using Gauss elimination
13:     end for
14: end for
15: return $N$
```

We use Lemma 59 to show the correctness of Step 11 in Algorithm 7.

Lemma 59. *Let M be a matrix of dimension $|B| \times |\mathcal{K}|$ where $|\mathcal{K}| = t|B|$ for some $t \in \mathbb{N}$. Let us consider t submatrices M_i of dimension $|B| \times |B|$ such that $M = M_1 \| M_2 \| \ldots \| M_t$.*

[1]The rationales for this assumption were given in Section 5.2.4.

Then,

$$\mathit{ker}(M) = \bigcap_{i\in[1,t]} \mathit{ker}(M_i)$$

Proof. *We prove this directly from definition. We have*

$$\mathit{ker}(M) := \{v; vM = 0\}$$

and

$$M = M_1 \| M_2 \| \dots \| M_t$$

For

$$v \in \bigcap_{i\in[1,t]} \mathit{ker}(M_i)$$

we have

$$\begin{aligned} vM &= vM_1 \| vM_2 \| \dots \| vM_t \\ &= 0 \quad \| 0 \quad \| \dots \| 0 \end{aligned}$$

Hence, we obtain

$$\mathit{ker}(M) \supseteq \bigcap_{i\in[1,t]} \mathit{ker}(M_i)$$

On the other hand, for

$$v \in \mathit{ker}(M)$$

we have

$$\begin{aligned} vM &= 0 \quad \| 0 \quad \| \dots \| 0 \\ &= vM_1 \| vM_2 \| \dots \| vM_t \end{aligned}$$

Hence, we obtain

$$\mathit{ker}(M) \subseteq \bigcap_{i\in[1,t]} \mathit{ker}(M_i)$$

□

We now discuss time and memory complexity of Algorithm 7 and compare this to Algorithm 6. The memory complexity of Algorithm 6 is given by $O(|B|\,|\mathcal{K}|)$. However, Algorithm 7 has memory complexity $O\left(|B|^2\right)$. The best implementation of Algorithm 6 and Algorithm 7 achieve the same asymptotic complexity $O\left(|B|^{\omega}\frac{|\mathcal{K}|}{|B|}\right)$, however, the straightforward implementation of Gauss elimination leads to complexity $O\left(|B|^2 |\mathcal{K}|\right)$.

5.2.4 Heuristic Universal Proning

In what follows, we explain the heuristic approach for recovering universal polynomials. Note that Universal Proning requires to go over all the key space. Hence, for an algorithm to remain competitive with respect to exhaustive search, it is necessary to consider a relatively small $\mathcal{K}$ of $\mathbf{F}_2^{\mathsf{kln}}$ of size K and compute $\mathcal{B}_{\chi,\gamma}^{\mathcal{K}}$ instead of $\mathcal{B}_{\chi,\gamma}$. However, there will always exist a polynomial $q \in \mathcal{B}_{\chi,\gamma}^{\mathcal{K}} \setminus \mathcal{B}_{\chi,\gamma}$, i.e, a nonuniversal polynomial q which will be recognised as universal by Algorithm 6 for parameter $\mathcal{K}$. In what follows, we explain a strategy to (heuristically) avoid them. To avoid this, we will restrict the space where we look for universal polynomials to a vector space $R \subseteq \mathbf{F}_2[V, \mathsf{Dup}(V)]$ so that the existance of $q \in R \cap \mathcal{B}_{\chi,\gamma}^{\mathcal{K}} \setminus \mathcal{B}_{\chi,\gamma}$ is less likely for a sufficiently large $\mathcal{K}$. Hence, instead of looking for universal polynomials in $R \cap \mathcal{B}_{\chi,\gamma}$, we consider only $R \cap \mathcal{B}_{\chi,\gamma}^{\mathcal{K}}$. We need $\mathcal{K}$ and R to verify

$$\emptyset = R \cap \left(\mathcal{B}_{\chi,\gamma}^{\mathcal{K}} \setminus \mathcal{B}_{\chi,\gamma}\right). \tag{5.2}$$

Actually, it would be sufficient to avoid having a q such that $[\![q]\!]_V \notin \langle \mathcal{S}_{\chi,\gamma,\star} \rangle$ (which follows from Theorem 58) but this is hard to test without the secret key.

On the choice of R. The choice of R is discussed in Section 5.3. Ideally, we want to select R as small as possible and such that

$$\{0\} \neq [\![R \cap \mathcal{B}_{\chi,\gamma}]\!]_V . \tag{5.3}$$

Otherwise, we would not find any additional polynomial to add to polynomial system $\mathcal{S}_{\chi,\gamma,\star}$. However, for a small R, we are likely to have $\{0\} = R \cap \mathcal{B}_{\chi,\gamma}$. For the purpose of this section, we assume that R contains polynomials of a low degree.

On size of $\mathcal{K}$. Our goal is to satisfy Eq. (5.2) with a small set $\mathcal{K} \subseteq \mathbf{F}_2^{\mathsf{kln}}$ selected uniformly at random. Let $R \subseteq \mathbf{F}_2[V, \mathsf{Dup}(V)]$ and $\kappa \in \mathbf{F}_2^{\mathsf{kln}}$ be selected uniformly at random. Let $\gamma = E_\kappa(\chi)$ and let (χ, γ) be our list of samples. Let $K_{R,\chi}$ be such that for a randomly selected $\mathcal{K} \subseteq \mathbf{F}_2^{\mathsf{kln}}$ where $|\mathcal{K}| = K_{R,\chi}$ the condition in Eq. (5.2) is satisfied with a high probability. However, this condition is expensive to verify and hence, we consider the condition

$$\left[\!\!\left[R \cap \mathcal{B}_{\chi,\gamma}^{\mathcal{K}}\right]\!\!\right]_V \subseteq \langle \mathcal{S}_{\chi,\gamma,\star} \rangle \tag{5.4}$$

which is implied by Eq. (5.2). We now give Algorithm 8 which finds the value $K_{R,\chi}$. We note that the condition Eq. (5.4) is easy to verify when we know the secret key - this is done in Step 19 of Algorithm 8.

Algorithm 8 Find $K_{R,\chi}$

```
Input: $\chi$, vector space $R \subseteq \mathbf{F}_2[V, \mathsf{Dup}(V)]$
Output: $K_R$
 1: $\kappa^\star \leftarrow \mathbf{F}_2^{\mathsf{kln}}$ selected uniformly at random
 2: $\gamma = E_{\kappa^\star}(\chi)$
 3: select linear basis $B$ of $R$
 4: $t \leftarrow 0$
 5: $N_0 \leftarrow R$
 6: repeat
 7:     $t \leftarrow t+1$
 8:     $M \leftarrow$ matrix of dimension $|B| \times |B|$
 9:     $\mathcal{K}_t \leftarrow$ subset of $\mathbf{F}_2^{\mathsf{kln}}$ of size $|B|$ selected uniformly at random
10:     for all $b \in B$ do
11:         for all $\kappa \in \mathcal{K}_t$ do
12:             $M_{b,\kappa} \leftarrow f_{\chi,\gamma}|_\kappa(b)$
13:         end for
14:         $T \leftarrow \mathsf{ker}(M)$ using Gauss elimination
15:         $N_t \leftarrow N_{t-1} \cap T$ using Gauss elimination
16:     end for
17:     set mark
18:     for all $q \in N_t$ do
19:         if $f_{\chi,\gamma}(q)\big|_{\kappa^\star} \neq 0$ then
20:             unset mark
21:         end if
22:     end for
23: until mark set
24: return $t \cdot |B|$ {we used $t \cdot |B|$ keys until condition given in Eq. (5.4) was satisfied
    (Step 19).}
```

Empirical results. We run Algorithm 8 multiple times and for a fixed R and a fixed χ, the algorithm returned the same $K_{R,\chi}$ independently of the choice of the secret key in Step 1. As conditions in Eq. (5.2) and Eq. (5.4) are not equivalent, we verified whether it is sufficient to test the condition in Eq. (5.2). We replaced the test in Step 19 by a check whether $0 = \dim(N_t + 1) - \dim(N_t)$ for the last $0.5t$ iterations. In our experiments, we observed that once the condition in Eq. (5.4) was satisfied, the $\dim(N_t)$ was invariant in subsequent iterations. We now give a more detailed analysis of Algorithm 8. We consider the evolution of the sets N_t and we look at the logarithm of the ratio $\frac{|N_t|}{|N_{t+1}|}$, i.e, we look at the difference $\dim(N_t) - \dim(N_{t+1})$. This tells us how much the dimension of $\mathcal{B}^{\mathcal{K}}_{\chi,\gamma}$ drops when new keys are added into $\mathcal{K}$. Once it does not drop for sufficiently large number of new keys, we can assume $R \cap \mathcal{B}^{\mathcal{K}}_{\chi,\gamma} = R \cap \mathcal{B}_{\chi,\gamma}$. However, in our tests we found that it was sufficient to test condition given in Eq. (5.4). We show that this difference behaves similarly for different reduced round versions of KATAN32 and various choices of R. We plot these differences in Figures 5.2-5.9. In our tests, we considered R of dimension up to 50 000 and in each case, we needed $K \geq 50 \cdot \dim(R)$. We used the following values for R:

$$R = \mathbf{linspan}\left(\bigcup_{\substack{r\in[35,45] \\ p\in[1,\mathsf{smpn}] \\ j,j'\in[1,\mathsf{mln}]}} \left\{ \mathsf{s}^{j}_{p,r} \cdot \mathsf{s}^{j'}_{p,r} \right\} \cup \bigcup_{j,j'\in[1,\mathsf{kln}]} \left\{ k_j \cdot k_{j'} \right\} \right) \tag{5.5}$$

$$R = \mathbf{linspan}\left(\bigcup_{\substack{p\in[1,\mathsf{smpn}] \\ j,j'\in[1,\mathsf{mln}]}} \left\{ \mathsf{s}^{j}_{p,45} \cdot \mathsf{s}^{j'}_{p,45} \right\} \cup \bigcup_{j,j'\in[1,\mathsf{kln}]} \left\{ k_j \cdot k_{j'} \right\} \right) \tag{5.6}$$

$$R = \mathbf{linspan}\left(\bigcup_{\substack{p\in[1,\mathsf{smpn}] \\ j,j'\in[1,\mathsf{mln}]}} \left\{ \mathsf{s}^{j}_{p,50} \cdot \mathsf{s}^{j'}_{p,50} \right\} \cup \bigcup_{j,j'\in[1,\mathsf{kln}]} \left\{ k_j \cdot k_{j'} \right\} \right) \tag{5.7}$$

Finishing the attack. Once we found R and $K_{R,\chi}$, we can perform the attack for an unknown key. We select a random set of keys $\mathcal{K} \subseteq \mathbf{F}_2^{\mathsf{kln}}$ of size $|\mathcal{K}| = K_{R,\chi}$. Then, we perform Algorithm 6 and recover

$$Q = R \cap \mathcal{B}^{\mathcal{K}}_{\chi,\gamma}$$

Finally, we compute $[\![Q]\!]_V$ which we use as additional input to other algebraic solvers such as mXL/F4.

5.3 Mutant Proning

In Section 5.2, we developed an algorithm called Universal Proning. The output of this algorithm is not very helpful because it consists of polynomials which are satisfied

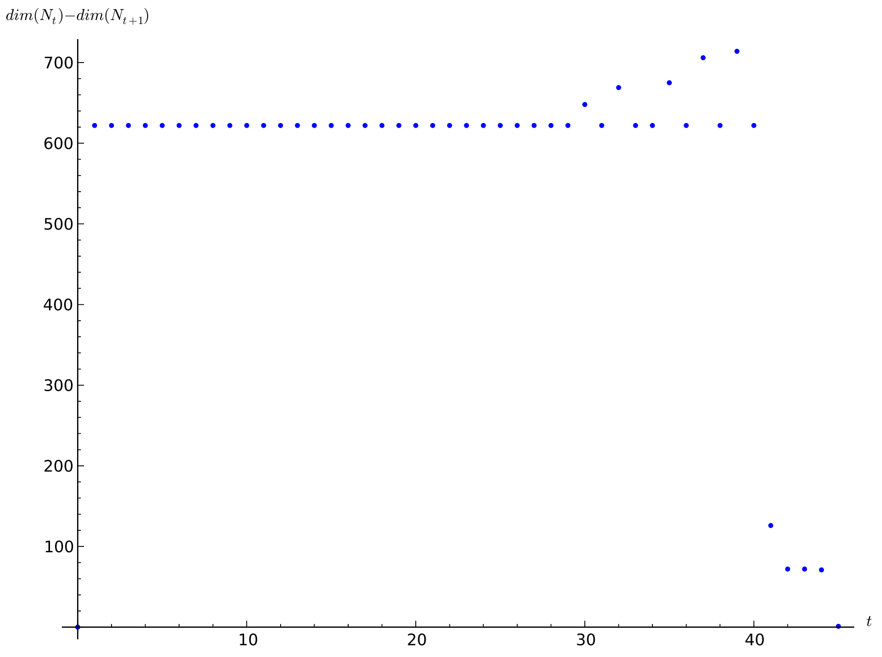


Figure 5.2: 62-round KATAN32, *R* set in Eq. (5.5)

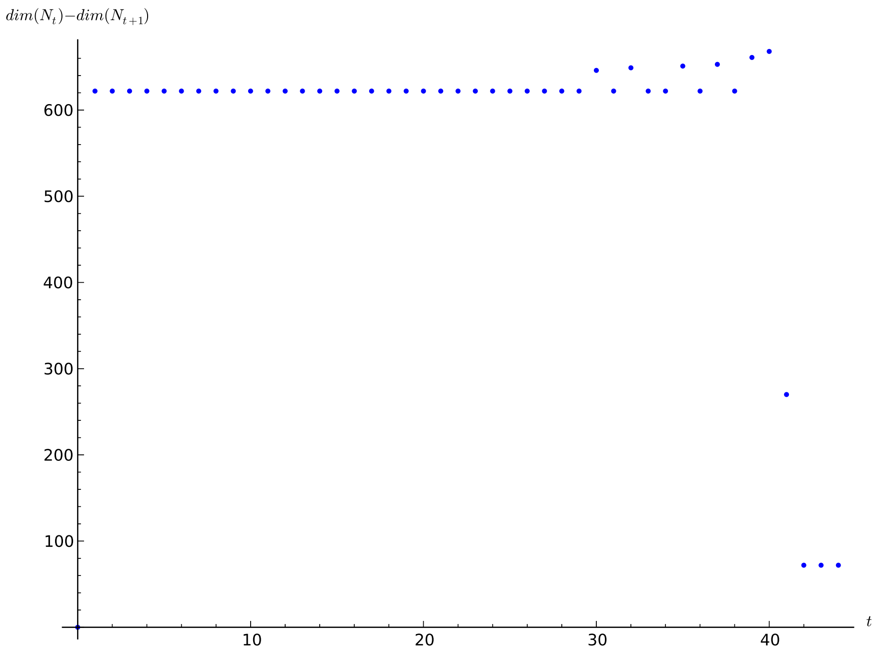


Figure 5.3: 64-round KATAN32, *R* set in Eq. (5.5)

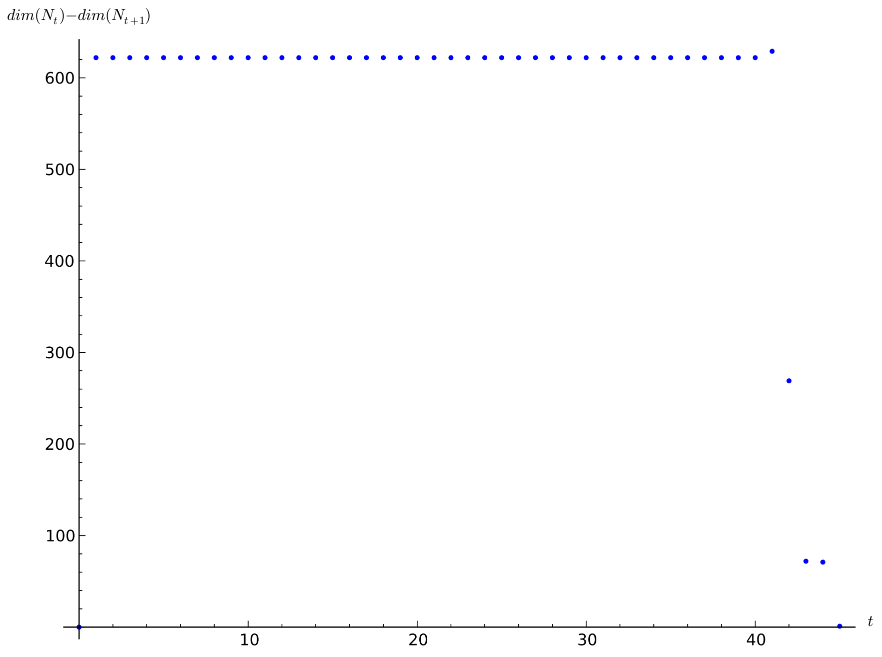


Figure 5.4: 75-round KATAN32, R set in Eq. (5.5)

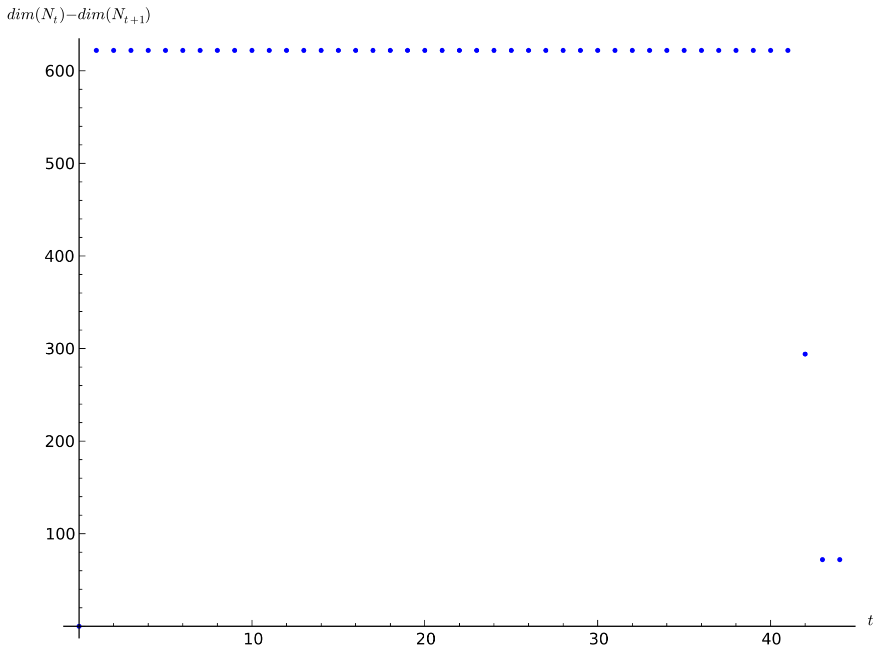


Figure 5.5: 76-round KATAN32, R set in Eq. (5.5)

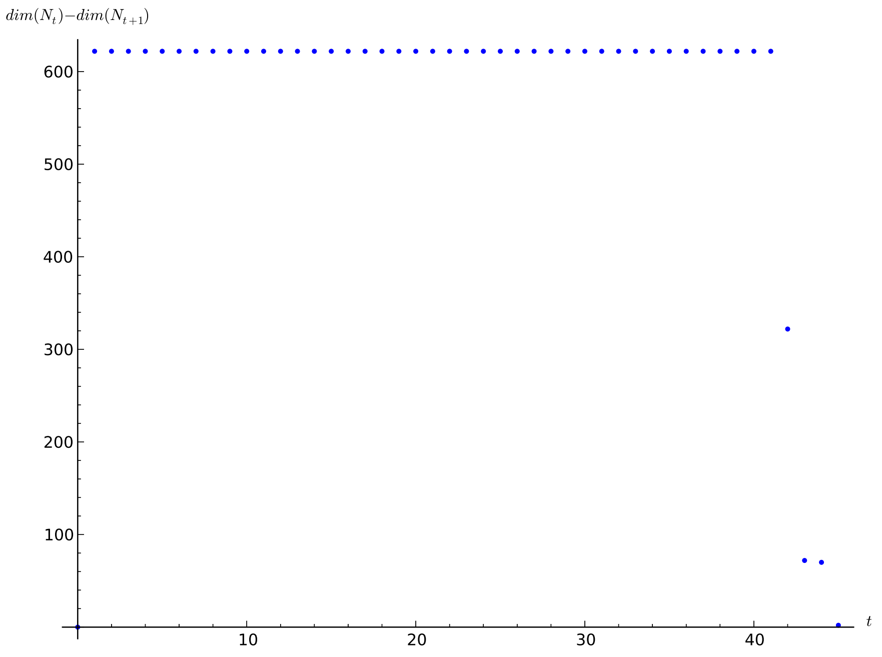


Figure 5.6: 77-round KATAN32, R set in Eq. (5.5)

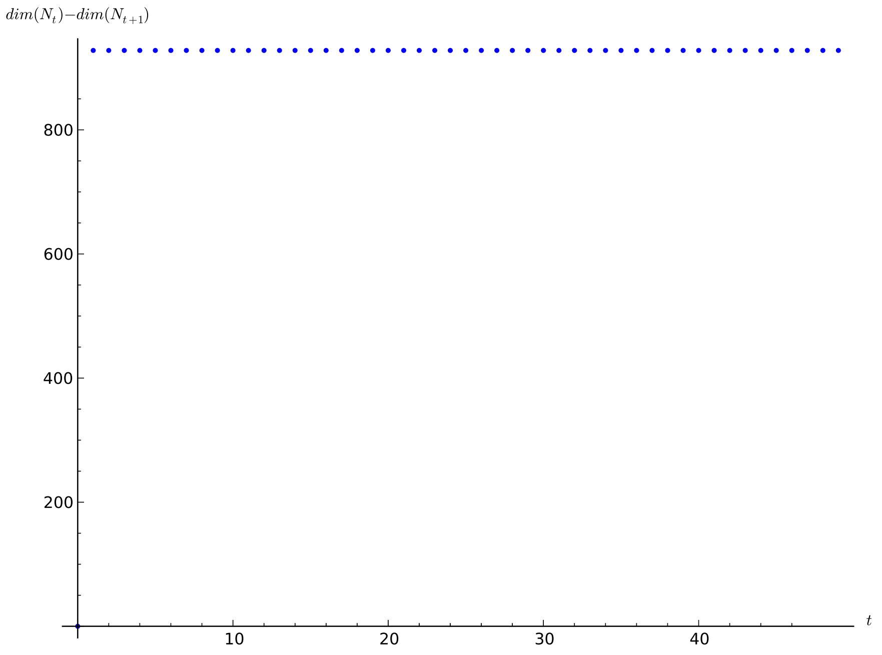


Figure 5.7: 62-round KATAN32, R set in Eq. (5.7)

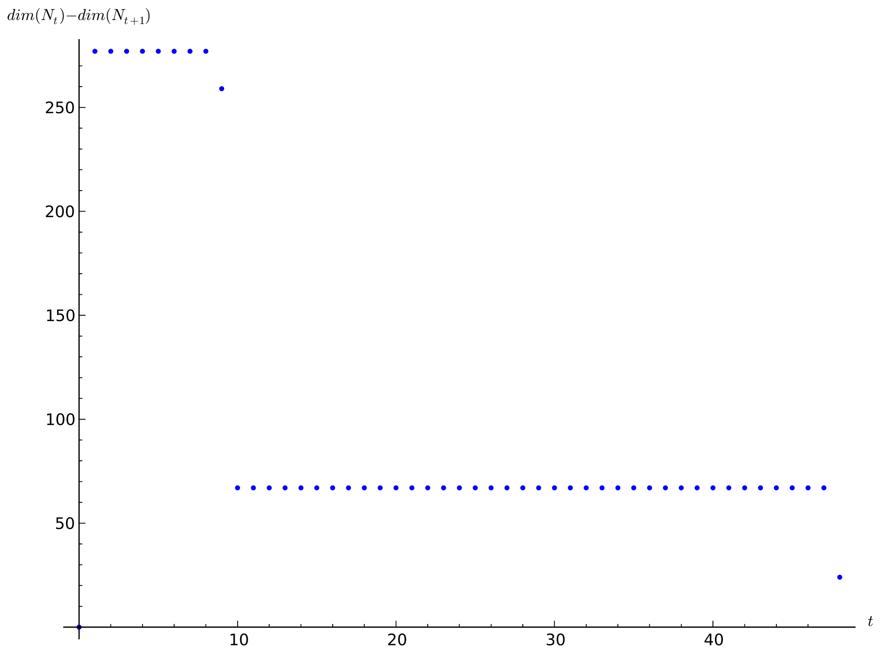


Figure 5.8: 62-round KATAN32, *R* set in Eq. (5.6)

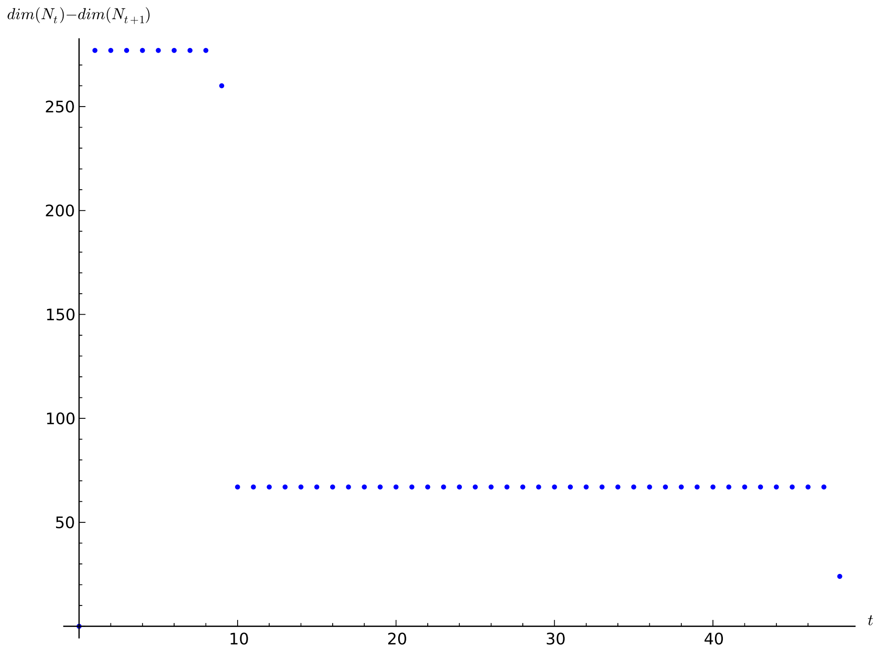


Figure 5.9: 64-round KATAN32, *R* set in Eq. (5.6)

for all secret keys. However, we can transform the output of Algorithm 6 using back-substitution into polynomials which are nonuniversal. We introduce an algorithm called Mutant Proning. The Mutant Proning recovers mutant polynomials which can be used to speed-up mXL/F4 computation.

Intuition: recover polynomial in key variables. We give an intuition for a special case: we recover polynomials which belong to the ideal spanned by our polynomial system and which contain only key variables (we call these polynomials as keynomials). Formally, we recover a subset of $\mathbf{F}_2[V_K] \cap \langle \mathcal{S}_{\chi,\gamma,\star} \rangle$. Let us consider the set of polynomials

$$\mathcal{B}_{\chi,\gamma} \cap (\langle v + \mathsf{Dup}(v) : v \in V \rangle + \mathbf{F}_2[V_K]),$$

which can be obtained from Universal Proning, i.e, Algorithm 6 with

$$R = \langle v + \mathsf{Dup}(v) : v \in V \rangle + \mathbf{F}_2[V_K].$$

Using Theorem 58, we obtain

$$\llbracket \mathcal{B}_{\chi,\gamma} \cap (\langle v + \mathsf{Dup}(v) : v \in V \rangle + \mathbf{F}_2[V_K]) \rrbracket_V \subseteq \langle \mathcal{S}_{\chi,\gamma,\star} \rangle .$$

I.e, we obtain polynomials in key variables which after back-substitution belong to the ideal $\langle \mathcal{S}_{\chi,\gamma,\star} \rangle$. Above, we considered $R = \langle v + \mathsf{Dup}(v) : v \in V \rangle + \mathbf{F}_2[V_K]$ as an input to Algorithm 6. In Lemma 66 to show that it is sufficient to consider a smaller vector space O (introduced in Notation 61) instead of the ideal $\langle v + \mathsf{Dup}(v) : v \in V \rangle$ which reduces the memory requirements of Universal Proning.

General case. Experimentally, we verified that (as expected) low degree polynomials in vector space $\mathcal{B}_{\chi,\gamma} \cap (\langle v + \mathsf{Dup}(v) : v \in V \rangle + \mathbf{F}_2[V_K])$ are very rare which means that usually, we have

$$\{0\} = \left\llbracket \int \mathcal{B}_{\chi,\gamma} \cap (\langle v + \mathsf{Dup}(v) : v \in V \rangle + \mathbf{F}_2[V_K]) \Big(^{D} \right\rrbracket_V$$

for a small $D \in \mathbb{N}$. Hence, we concentrate on recovering nonuniversal polynomials as

$$\left\llbracket \mathcal{B}_{\chi,\gamma} \cap \left(\int \langle v + \mathsf{Dup}(v) : v \in V \rangle \Big(^{a} + \int \mathbf{F}_2[V] \Big(^{b} \right) \right\rrbracket_V$$

for $a, b \in \mathbb{N}$. We call this Mutant Proning.

Definition 60 (zeromial)**.** *Let $q \in \mathbf{F}_2[V, \mathit{Dup}(V)]$. We call q a zeromial iff $\llbracket q \rrbracket_V = 0$.*

D \ $\lvert V_S\rvert$	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}
5	$2^{43.07}$	$2^{48.08}$	$2^{53.08}$	$2^{58.09}$	$2^{63.09}$	$2^{68.09}$	$2^{73.09}$	$2^{78.09}$	$2^{83.09}$
6	$2^{50.48}$	$2^{56.49}$	$2^{62.50}$	$2^{68.50}$	$2^{74.50}$	$2^{80.50}$	$2^{86.50}$	$2^{92.50}$	$2^{98.50}$
7	$2^{57.67}$	$2^{64.68}$	$2^{71.69}$	$2^{78.69}$	$2^{85.69}$	$2^{92.69}$	$2^{99.70}$	$2^{106.70}$	$2^{113.70}$
8	$2^{64.66}$	$2^{72.68}$	$2^{80.69}$	$2^{88.69}$	$2^{96.69}$	$2^{104.69}$	$2^{112.70}$	$2^{120.70}$	$2^{128.70}$
9	$2^{71.48}$	$2^{80.05}$	$2^{89.51}$	$2^{98.52}$	$2^{107.52}$	$2^{116.52}$	$2^{125.53}$	$2^{134.53}$	$2^{143.53}$
10	$2^{78.14}$	$2^{88.17}$	$2^{98.19}$	$2^{108.20}$	$2^{118.20}$	$2^{128.20}$	$2^{138.20}$	$2^{148.20}$	$2^{158.20}$
11	$2^{84.67}$	$2^{95.71}$	$2^{106.73}$	$2^{117.73}$	$2^{128.74}$	$2^{139.74}$	$2^{150.74}$	$2^{161.74}$	$2^{172.74}$
12	$2^{91.07}$	$2^{103.11}$	$2^{115.14}$	$2^{127.15}$	$2^{139.15}$	$2^{151.16}$	$2^{163.16}$	$2^{175.16}$	$2^{187.16}$
13	$2^{97.35}$	$2^{110.40}$	$2^{123.43}$	$2^{136.45}$	$2^{149.45}$	$2^{162.46}$	$2^{175.46}$	$2^{188.46}$	$2^{201.46}$
14	$2^{103.52}$	$2^{117.59}$	$2^{131.62}$	$2^{149.45}$	$2^{159.64}$	$2^{173.65}$	$2^{187.65}$	$2^{201.65}$	$2^{215.65}$
15	$2^{109.60}$	$2^{124.67}$	$2^{139.71}$	$2^{154.73}$	$2^{169.74}$	$2^{184.74}$	$2^{199.74}$	$2^{214.74}$	$2^{229.74}$
16	$2^{115.57}$	$2^{131.66}$	$2^{147.70}$	$2^{163.72}$	$2^{179.73}$	$2^{195.74}$	$2^{211.74}$	$2^{227.74}$	$2^{243.74}$
17	$2^{121.46}$	$2^{138.56}$	$2^{155.61}$	$2^{172.63}$	$2^{189.65}$	$2^{206.65}$	$2^{223.65}$	$2^{240.66}$	$2^{257.66}$
18	$2^{127.27}$	$2^{145.38}$	$2^{163.43}$	$2^{181.46}$	$2^{199.47}$	$2^{217.48}$	$2^{235.48}$	$2^{253.49}$	$2^{271.49}$

Table 5.2: $\binom{|V_S|}{D}$

Notation 61. *We denote*

$$O = \langle v + \mathit{Dup}(v),\ v \in V \rangle \cap (\mathbf{F}_2[V] + \mathbf{F}_2[\mathit{Dup}(V)])$$

In Table 5.2, we consider only the state variables (see Notation 11). We give a table of the dominating term $\binom{|V_S|}{D}$ in $\sum_{d \leq D} \binom{|V_S|}{d}$ for selected values of D and $|V_S|$.

Lemma 62 (existence of nonuniversal zeromial)**.** *We have* $\mathsf{kln} \neq 0 \Longrightarrow O \not\subseteq \mathcal{B}_{\chi,\gamma}$.

Proof. *Let* κ *be an incorrect key for* $\mathcal{S}_{\chi,\gamma,\star}$ *and* v *be a plaintext variable such that* $\mathcal{S}_{\star,\gamma,\kappa}$ *assigns* v *to a value which does not match* χ. *Let us consider the zeromial* $q = v + \mathit{Dup}(v)$. *Due to mismatch, we have* $f_{\chi,\gamma}(q)\,|_{\kappa} = 1$ *so* $f_{\chi,\gamma}(q) \neq 0$ *so* $q \notin \mathcal{B}_{\chi,\gamma}$. *Hence,* $O \not\subseteq \mathcal{B}_{\chi,\gamma}$. □

In what follows, we first show how we can construct a universal polynomial from a nonuniversal polynomial. We show this in Lemma 63. We restrict $f_{\chi,\gamma}$ on equations which are not satisfied for all keys and define canonical mapping μ from $\mathbb{B}[V_K]$ to $\mathsf{Func}\left(\mathbf{F}_2^{\mathsf{kln}}, \mathbf{F}_2\right)$. We consider the following chain:

$$\mathbf{F}_2[V, \mathsf{Dup}(V)] \setminus \ker\left(f_{\chi,\gamma}\right) \xrightarrow{f_{\chi,\gamma}} \mathsf{Func}\left(\mathbf{F}_2^{\mathsf{kln}}, \mathbf{F}_2\right) \setminus \{0\} \xleftarrow{\mu} \mathbb{B}[V_K] \setminus \{0\} \qquad (5.8)$$

Lemma 63. *Let* $q \in \mathbf{F}_2[V, \mathit{Dup}(V)]$. *We have* $\left(q + \mu^{-1}\left(f_{\chi,\gamma}(q)\right)\right) \in \mathcal{B}_{\chi,\gamma}$.

Proof. *Let $F \in \mathsf{Func}\left(\mathbf{F}_2^{kln}, \mathbf{F}_2\right)$. Then $\mu^{-1}(F)$ is a polynomial defining the same function as F. Therefore, we have $f_{\chi,\gamma}\left(\mu^{-1}(F)\right) = F$. We consider $F = f_{\chi,\gamma}(q)$ and we obtain*

$$0 = F + f_{\chi,\gamma}\left(\mu^{-1}(F)\right) = f_{\chi,\gamma}(q) + f_{\chi,\gamma}\left(\mu^{-1}\left(f_{\chi,\gamma}(q)\right)\right) = f_{\chi,\gamma}\left(q + \mu^{-1}\left(f_{\chi,\gamma}(q)\right)\right).$$

Hence,

$$q + \mu^{-1}\left(f_{\chi,\gamma}(q)\right) \in \mathsf{ker}\left(f_{\chi,\gamma}\right) = \mathcal{B}_{\chi,\gamma}.$$

□

Definition 64 (keynomial)**.** *We call a polynomial $q \in \mathbb{B}[V_K]$ a keynomial.*

Lemma 65 (keynomials are nonuniversal)**.** $\mathbb{B}[V_K] \cap \mathcal{B}_{\chi,\gamma} = \{0\}$.

Proof. *We first show $\mathbf{F}_2[V_K] \cap \mathcal{B}_{\chi,\gamma} = \mathsf{FieldEq}[V_K]$. From the definition of $\mathcal{S}$, we have $\mathsf{FieldEq}[V_K] \subseteq \mathcal{B}_{\chi,\gamma}$. Using Theorem 51, we have $\mathcal{B}_{\chi,\gamma} = \ker\left(f_{\chi,\gamma}\right)$. Let us consider a polynomial $q \in \mathbf{F}_2[V_K]$ such that $q \notin \mathsf{FieldEq}[V_K]$. Based on Corollary 25, q defines a nonzero boolean function and hence, $1+q$ has a root, i.e, $\kappa \in \mathbf{F}_2^{kln}$ such that $q(\kappa) = 1$. Therefore, $q \notin \ker\left(f_{\chi,\gamma}\right)$ and hence, $q \notin \mathcal{B}_{\chi,\gamma}$. Hence, we have $\mathbf{F}_2[V_K] \cap \mathcal{B}_{\chi,\gamma} = \mathsf{FieldEq}[V_K]$. As $\mathbb{B}[V_K] \cap \mathsf{FieldEq}[V_K] = \{0\}$ we obtain the result.* □

We will use Lemma 62 together with Lemma 63 to construct a keynomial. Using Lemma 65, we know that this keynomial is nonuniversal and hence, we can derive some information about the secret key. In Lemma 66, we show that every polynomial from $\mathcal{B}_{\chi,\gamma}$ has a "representantive" in the vector space $\mathcal{P}_\chi + \mathsf{Dup}\left(\mathcal{C}_\gamma\right)$. We use this to avoid recovering "equivalent" universal polynomials and decrease computational requirements.

Lemma 66 (equivalent universal polynomials)**.** *For each $q \in \mathcal{B}_{\chi,\gamma} \cap \mathbb{B}[V_S, \mathsf{Dup}(V_S)]$ there exists*

$$q' \in \left(\mathcal{P}_\chi + \mathsf{Dup}\left(\mathcal{C}_\gamma\right)\right)$$

such that $[\![q - q']\!]_V = 0$.

Proof. *Let $q \in \mathcal{B}_{\chi,\gamma}$. Following Theorem 58, we can write $q = ap + b\mathsf{Dup}(c)$ for $p \in \mathcal{P}_\chi$, $c \in \mathcal{C}_\gamma$ and $a, b \in \mathbf{F}_2[V, \mathsf{Dup}(V)]$. We set $q' = [\![ap]\!]_V + \mathsf{Dup}([\![bc]\!]_V)$ and from the construction of q' we have $[\![q - q']\!]_V = 0$. As $[\![ap]\!]_V \in \mathcal{P}_\chi \cap \mathbb{B}[V]$ and $[\![bc]\!]_V \in \mathcal{C}_\gamma \cap \mathbb{B}[V]$ we have*

$$q' \in \left(\mathcal{P}_\chi + \mathsf{Dup}\left(\mathcal{C}_\gamma\right)\right).$$

□

Lemma 66 shows that it is sufficient to consider only the vector space O instead of the ideal $\langle v + \mathsf{Dup}(v) : v \in V\rangle$. In Lemma 67, we give the algorithmic relation between mXL/F4 and Universal Proning. In Figure 5.11, we give a graphical representation for better intuition.

Lemma 67 (relation of Universal Proning with mXL/F4)**.** *Let $m \in \left\langle \mathcal{S}_{\chi,\gamma,\star} \right\rangle$. Then, there exists*

$$m' \in \int \mathbb{B}[V] + \mathbb{B}[\mathit{Dup}(V)] \Big(^{\deg m}$$

and

$$q_0 \in \int \langle v + \mathit{Dup}(v) : v \in V \rangle \cap (\mathbb{B}[V] + \mathbb{B}[\mathit{Dup}(V)]) \Big(^{\mathit{level}_{\mathcal{P}_\chi \cup \mathcal{C}_\gamma}(m)}$$

such that $m = [\![m' + q_0]\!]_V$ and $m' + q_0 \in \mathcal{B}_{\chi,\gamma}$.

Proof. *We express*

$$m = p + c$$

where $p \in \mathcal{P}_\chi$ and $c \in \mathcal{C}_\gamma$ and $\deg p, \deg c \leq \mathit{level}(m)$. Let us now consider

$$q = p + \mathit{Dup}(c)$$

and

$$r' \in \int \mathbb{B}[V] + \mathbb{B}[\mathit{Dup}(V)] \Big(^{\deg m}$$

and

$$r'' \in \overline{\int \mathbb{B}[V] + \mathbb{B}[\mathit{Dup}(V)] \Big(^{\deg m}}$$

such that $q = r' + r''$. We set

$$m' = r' + [\![r'']\!]_V \text{ and } q_0 = r'' + [\![r'']\!]_V .$$

We have $m = p + c = [\![q]\!]_V = [\![r' + r'']\!]_V = [\![m']\!]_V + [\![q_0]\!]_V$. As $q = p + \mathit{Dup}(c)$ and due to Theorem 58, we have $p \in \mathcal{P}_\chi, c \in \mathcal{C}_\gamma$. Hence, $q \in \mathcal{P}_\chi + \mathit{Dup}\left(\mathcal{C}_\gamma\right) \subseteq \mathcal{B}_{\chi,\gamma}$. I.e, we have

$$m' = r' + [\![r']\!]_V \in \int \mathbb{B}[V] + \mathbb{B}[\mathit{Dup}(V)] \Big(^{\deg m}.$$

It remains to show that

$$q_0 \in \int \langle v + \mathit{Dup}(v) : v \in V \rangle \cap (\mathbb{B}[V] + \mathbb{B}[\mathit{Dup}(V)]) \Big(^{\mathit{level}_{\mathcal{P}_\chi \cup \mathcal{C}_\gamma}(m)}.$$

We have

$$\deg q_0 = \deg\left(r'' + [\![r'']\!]_V\right) \leq \mathit{level}_{\mathcal{P}_\chi \cup \mathcal{C}_\gamma}(m).$$

As $[\![q_0]\!]_V = 0$ and $\mathit{ker}([\![]\!]_V) = \langle v + \mathit{Dup}(v) : v \in V \rangle$, we have $q_0 \in \langle v + \mathit{Dup}(v) : v \in V \rangle$. We now show $q_0 \in \mathbb{B}[V] + \mathbb{B}[\mathit{Dup}(V)]$. We have $r'' \in \mathbb{B}[V] + \mathbb{B}[\mathit{Dup}(V)]$ and $[\![r'']\!]_V \in \mathbb{B}[V]$ and therefore $r'' + [\![r'']\!]_V = q_0 \in \mathbb{B}[V] + \mathbb{B}[\mathit{Dup}(V)]$. □

We now introduce Mutant Proning. Our goal is to obtain new polynomials from ideal $\left\langle \mathcal{S}_{\chi,\gamma,\star} \right\rangle$ because additional polynomials usually help in mXL/F4 to speed up the computation. We already know $\mathcal{S}_{\chi,\gamma,\star} \subseteq \langle \mathcal{S}_{\chi,\gamma,\star} \rangle$ which are given a priori. But running mXL/F4 only using these polynomials is usually expensive. Similarly, we are not very interested in sets of universal polynomials $\langle \mathcal{S}_{\chi,\star,\star} \rangle_{\mathbf{F}_2[V]}$ resp. $\langle \mathcal{S}_{\star,\gamma,\star} \rangle_{\mathbf{F}_2[V]}$ as these polynomials are satisfied for every value of secret key. Hence, we concentrate on finding nonuniversal polynomials of the ideal $\langle \mathcal{S}_{\chi,\gamma,\star} \rangle$.

Let us consider a polynomial $m \in \langle \mathcal{S}_{\chi,\gamma,\star} \rangle$ such that $\deg m < \mathsf{level}_{\mathcal{P}_\chi \cup \mathcal{C}_\gamma}(m)$, i.e, m is a mutant. In Lemma 68, we show that m is nonuniversal and in Lemma 67, we show how we can construct such mutant using Universal Proning. In Figure 5.11, we give a schematic view of such construction.

Mutants. In mXL, we can discover two types of mutants: universal and nonuniversal. This is because mXL operates on the system $\mathcal{S}_{\chi,\gamma,\star}$ and hence, it can find a polynomial $m \in \langle \mathcal{S}_{\chi,\star,\star} \rangle$ such that $\deg m < \mathsf{level}_{\mathcal{S}_{\chi,\star,\star}}(m)$. Such a polynomial is universal mutant. Universal mutants allow to reduce degree which is reached by mXL before it finds the secret key. However, universal mutants do not allow us to derive any information about the secret key. Hence, it is more interesting to look for nonuniversal mutants.

Lemma 68 (nonuniversal mutants). *Let $m \in \langle \mathcal{S}_{\chi,\gamma,\star} \rangle$ such that $\deg m < \mathsf{level}_{\mathcal{P}_\chi \cup \mathcal{C}_\gamma}(m)$. Then, both m and $\mathsf{Dup}(m)$ are nonuniversal.*

Proof. *As $\deg m < \mathsf{level}_{\mathcal{P}_\chi \cup \mathcal{C}_\gamma}(m)$, we have $m \notin \mathcal{P}_\chi$. Otherwise, we would have $\deg m = \mathsf{level}_{\mathcal{P}_\chi \cup \mathcal{C}_\gamma}(m)$. Similarly, we have $m \notin \mathcal{C}_\gamma$ as otherwise, we have $\deg m = \mathsf{level}_{\mathcal{P}_\chi \cup \mathcal{C}_\gamma}(m)$. Hence, we also have $\mathsf{Dup}(m) \notin \mathsf{Dup}(\mathcal{C}_\gamma)$. Using Lemma 63, we have*

$$m + \mu^{-1}\left(f_{\chi,\gamma}(m)\right) \in \mathcal{B}_{\chi,\gamma}$$

and

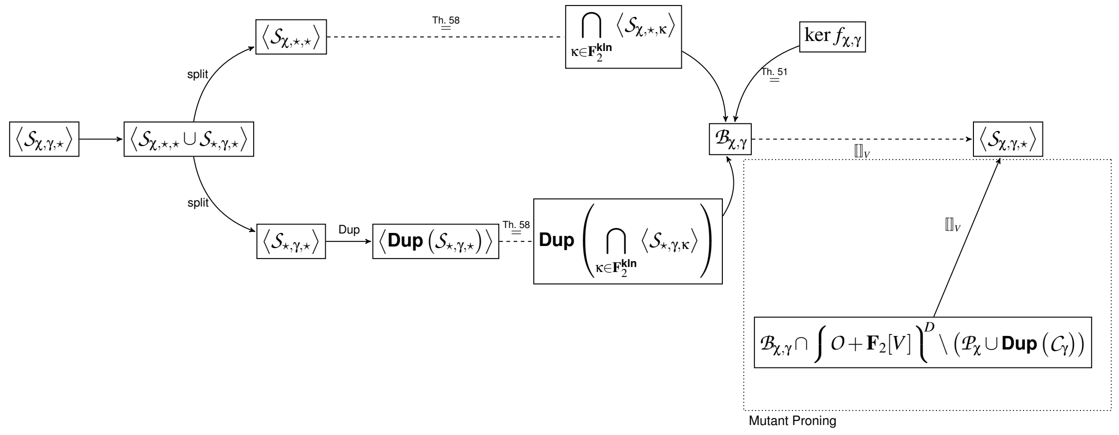
$$\mathsf{Dup}(m) + \mu^{-1}\left(f_{\chi,\gamma}(\mathsf{Dup}(m))\right) \in \mathcal{B}_{\chi,\gamma}.$$

By Theorem 51, $m \notin \mathcal{P}_\chi$ implies $e_\chi|_\kappa(m) \neq 0$. So, there is some κ which evaluates m to 1. So, $f_{\chi,\gamma}|_\kappa(m) \neq 1$. Therefore, $0 \neq \mu^{-1}\left(f_{\chi,\gamma}(m)\right)$. Due to Lemma 65, we deduce $\mu^{-1}\left(f_{\chi,\gamma}(m)\right) \notin \mathcal{B}_{\chi,\gamma}$ so, $m \notin \mathcal{B}_{\chi,\gamma}$. Similarly as $\mathsf{Dup}(m) \notin \mathsf{Dup}(\mathcal{C}_\gamma)$, we have $0 \neq \mu^{-1}\left(f_{\chi,\gamma}(\mathsf{Dup}(m))\right)$ and due to Lemma 65, we deduce $\mu^{-1}\left(f_{\chi,\gamma}(\mathsf{Dup}(m))\right) \notin \mathcal{B}_{\chi,\gamma}$ so, $\mathsf{Dup}(m) \notin \mathcal{B}_{\chi,\gamma}$. Hence, m and $\mathsf{Dup}(m)$ are nonuniversal. □

In Lemma 67, we showed that any mutant m can be constructed from a universal polynomial $m' + q_0$. We depict the scenario in Figure 5.11. This leads us to a new algorithm called Mutant Proning given in Algorithm 9.

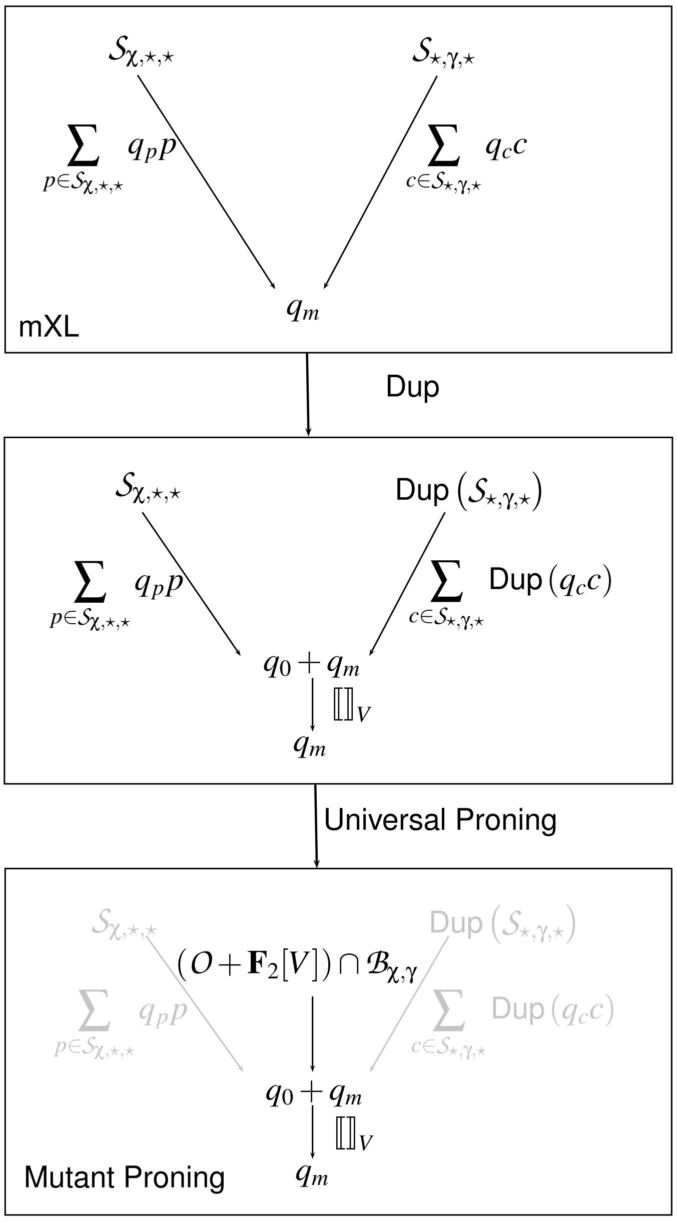
Algorithm 9 Mutant Proning

```
Input: $a, b \in \mathbb{N}$ such that $a > b$, $W \subseteq V$, $\mathcal{K} \subseteq \mathbf{F}_2^{\mathsf{kln}}$.
Output: set of mutants of level at most $a$ and degree at most $b$
 1: $R \leftarrow \lmoustache O \rmoustache^{a} + \lmoustache \mathbb{B}[W] + \mathbb{B}[\mathsf{Dup}\,(W)] \rmoustache^{b}$
 2: select a linear basis $B$ of linspan $(R)$
 3: $M \leftarrow$ matrix of dimension $|B| \times |\mathcal{K}|$
 4: for all $b \in B$ do
 5:     for all $\kappa \in \mathcal{K}$ do
 6:         $M_{b,\kappa} \leftarrow f_{\chi,\gamma}|_{\kappa}\,(b)$
 7:     end for
 8: end for
 9: find $N$ of maximal size with full rank such that $NM = 0$ using Gauss elimination
10: return the set of all $\sum_{b \in B} [\![ N_{i,b} b ]\!]_V$ for all $i$ and such that $\sum_{b \in B} N_{i,b} b \notin$
    $\lmoustache \mathbb{B}[W] \cup \mathbb{B}[\mathsf{Dup}\,(W)] \rmoustache^{b}$
```



$\langle \mathcal{S}_{\chi,\gamma,\star} \rangle$
$\langle \mathcal{S}_{\chi,\star,\star} \cup \mathcal{S}_{\star,\gamma,\star} \rangle$
split
split
$\langle \mathcal{S}_{\chi,\star,\star} \rangle$
Th. 58 $=$
$\bigcap_{\kappa \in \mathbf{F}_2^{\mathsf{kln}}} \langle \mathcal{S}_{\chi,\star,\kappa} \rangle$
$\ker f_{\chi,\gamma}$
Th. 51 $=$
$\mathcal{B}_{\chi,\gamma}$
$[\![\,]\!]_V$
$\langle \mathcal{S}_{\chi,\gamma,\star} \rangle$
$\langle \mathcal{S}_{\star,\gamma,\star} \rangle$
Dup
$\langle \mathbf{Dup}\,(\mathcal{S}_{\star,\gamma,\star}) \rangle$
Th. 58 $=$
$\mathbf{Dup}\left(\bigcap_{\kappa \in \mathbf{F}_2^{\mathsf{kln}}} \langle \mathcal{S}_{\star,\gamma,\kappa} \rangle \right)$
$[\![\,]\!]_V$
$\mathcal{B}_{\chi,\gamma} \cap \lmoustache O + \mathbf{F}_2[V] \rmoustache^{D} \setminus (\mathcal{P}_\chi \cup \mathbf{Dup}\,(\mathcal{C}_\gamma))$
Mutant Proning

Figure 5.10: Recovering new mutants with Mutant Proning



$\mathcal{S}_{\chi,\star,\star}$
$\mathcal{S}_{\star,\gamma,\star}$
$\sum_{p\in\mathcal{S}_{\chi,\star,\star}} q_p p$
$\sum_{c\in\mathcal{S}_{\star,\gamma,\star}} q_c c$
q_m
mXL
Dup
Dup $(\mathcal{S}_{\star,\gamma,\star})$
$\sum_{c\in\mathcal{S}_{\star,\gamma,\star}}$ Dup $(q_c c)$
$q_0 + q_m$
$[\![]\!]_V$
Universal Proning
$(\mathcal{O}+\mathbf{F}_2[V])\cap\mathcal{B}_{\chi,\gamma}$
Mutant Proning

Figure 5.11: Mutant Proning as a dual view on mXL.

Mutant recovery. We verified the results on KATAN32 with reduced round KATAN32. In our experiments, we attacked 75-round KATAN32 using Algorithm 9. We modified the Step 1 of Algorithm 9, and we selected

$$R = \mathbf{linspan}\left(\bigcup_{\substack{r\in[35,45] \\ p\in[1,\mathsf{smpn}] \\ j,j'\in[1,\mathsf{mln}]}} \left\{ \mathsf{s}^{j}_{p,r}\cdot \mathsf{s}^{j'}_{p,r} \right\} \cup \bigcup_{j,j'\in[1,\mathsf{kln}]} \left\{ k_j \cdot k_{j'} \right\} \right)$$

in order to reduce the memory requirements of Algorithm 9. These rounds correspond to approximately half of the cipher as we would expect mXL to find the first mutants among these variables. Hence, we obtained $\dim(R) \approx 50000$ and we selected $\mathcal{K} \subseteq \mathbf{F}_2^{\mathsf{kln}}$ uniformly at random such that $|\mathcal{K}| \approx 50\cdot\dim(R)$. Using these parameters, we recovered a set of mutants M which was consistent with our polynomial system, i.e, we had $1 \notin \langle \mathcal{S}_{\chi,\gamma,\star} + M \rangle$.

5.4 Iterative Proning

In this section, we propose an extension of Mutant Proning algorithm called Iterative Proning. Similarly as in Mutant Proning, our aim is to recover low degree polynomials of ideal spanned by our polynomial system, i.e, $\langle \mathcal{S}_{\chi,\gamma,\star} \rangle$. However, when we restrict Mutant Proning to set R as in Algorithm 9, we do not recover all low degree polynomials of the ideal $\langle \mathcal{S}_{\chi,\gamma,\star} \rangle$. In Iterative Proning, we intend to recover additional polynomials in the set R which were not found by Mutant Proning. We motivate our approach by the second iteration of mXL. In mXL, we recover mutants Q and use them as reductors. In Universal Proning, the computation $\bmod Q$ is equivalent to computation $\bmod \left(\mu^{-1}\left(f_{\chi,\gamma}(Q)\right)\right)$ which can be seen as filtering of the keyspace. Hence, we proceed as follows. We first recover mutants using mutant proning. Then, we use these mutants to restrict the keyspace to $\mathcal{K}_{\chi,\gamma}(Q)$. In the restricted keyspace, mutants from set Q behave like universal polynomials, i.e, $Q \subseteq \mathcal{B}_{\chi,\gamma}^{\mathcal{K}_{\chi,\gamma}(Q)}$; and hence, some new mutants may appear. We look for mutants using Mutant Proning introduced in Section 5.3. We select the set Q to be small so that we can sample $\mathcal{K}_{\chi,\gamma}(Q)$ efficiently. Then, we select the vector space in which we look for new mutants R and a subset $\mathcal{K}_i \subseteq \mathcal{K}_{\chi,\gamma}(Q)$ as in Section 5.2.4 and we compute a new set of mutants using Mutant Proning. We give the algorithm in Algorithm 10 and graphical representation in Figure 5.12. We iterate until we find enough mutants so that mXL/F4 is efficient and until we keep discovering new mutants for sets Q where $\mathcal{K}_{\chi,\gamma}(Q)$ can be sampled efficiently.

$$\begin{tikzcd}
\boxed{\mathcal{K}_0 \subseteq_R \mathbf{F}_2^{\mathsf{kln}}} \arrow[dr, "i \leftarrow 0"'] & \overset{\text{reasonably large subset}}{\boxed{\mathcal{K}_i \subseteq_R \mathcal{K}_{\chi,\gamma}(M_i)}} \arrow[d, "i \leftarrow i+1"] & \overset{\text{small subset}}{\boxed{M_i \subseteq_R \langle \mathcal{T} + \mathcal{O} \rangle}} \arrow[l] & \\
& \boxed{\mathcal{B}_{\chi,\gamma}^{\mathcal{K}_i} \cap R} \arrow[r, "\llbracket \cdot \rrbracket_V"] & \boxed{\mathcal{T} \leftarrow \mathcal{T} \cup \left\llbracket \mathcal{B}_{\chi,\gamma}^{\mathcal{K}_i} \cap R \right\rrbracket_V} \arrow[u] \arrow[r, phantom, "\overset{\text{heuristically}}{\subseteq}"] & \langle \mathcal{S}_{\chi,\gamma,\star} \rangle
\end{tikzcd}$$

Figure 5.12: Iterative Proning

Empirical results. In our experiments, we focused on 65 round KATAN32 and we aimed to evaluate the contribution of Step 19 of Algorithm 10, i.e, considering a subset of the "filtered" keyspace $\mathcal{K}_{\chi,\gamma}(M_i)$. Following Step 12 of Algorithm 10, we selected uniformly at random a set $M_i \subseteq \{v + \mathsf{Dup}(v) : v \in V\}$ of size $|M_i| = 4$. Then, we selected a set of keys uniformly at random $\mathcal{K}_i \subseteq \mathcal{K}_{\chi,\gamma}(M_i)$ (cf. Step 19). This set of keys was then used as an input for Algorithm 9. For a comparison, we run this algorithm with a set of keys selected uniformly at random $\mathcal{K} \subseteq \mathbf{F}_2^{\mathsf{kln}}$. In our experiments, we considered R given by

$$R = \mathbf{linspan}\left(\bigcup_{\substack{r \in [35,45] \\ p \in [1,\mathsf{smpn}] \\ j,j' \in [1,\mathsf{mln}]}} \left\{ \mathsf{s}_{p,r}^{j} \cdot \mathsf{s}_{p,r}^{j'} \right\} \cup \bigcup_{j,j' \in [1,\mathsf{kln}]} \{ k_j \cdot k_{j'} \} \right) \tag{5.9}$$

and we observed that $\left| R \cap \left(\mathcal{B}_{\chi,\gamma}^{\mathcal{K}_i} \setminus \mathcal{B}_{\chi,\gamma}^{\mathcal{K}} \right) \right| > 0$ in approximately 2% of cases (over the choices of different samples). Similarly, we tested Step 12 of Algorithm 10 for mutant polynomials. We first run Algorithm 9 with set of keys selected uniformly at random $\mathcal{K} \subseteq \mathbf{F}_2^{\mathsf{kln}}$ as in Step 1 of Algorithm 10 and we obtained a set of mutants M. Then, we considered $M_i \subseteq M$ such that $|M_i| = 4$ and we selected a set of keys uniformly at random $\mathcal{K}_i \subseteq \mathcal{K}_{\chi,\gamma}(M_i)$ (cf. Step 19). We observed that $\left| R \cap \left(\mathcal{B}_{\chi,\gamma}^{\mathcal{K}_i} \setminus \mathcal{B}_{\chi,\gamma}^{\mathcal{K}} \right) \right| > 0$ in approximately 5% of cases (over the choices of different samples). However in our experiments, we considered a fixed set R in Step 9 which restricted the number of new mutants we could recover and we expect improved results for a better selection of R.

5.5 Speeding-up standard algebraic techniques.

We proposed Universal Proning, Mutant Proning and Iterative Proning. Each of these techniques allow us to recover polynomials of ideal $\langle \mathcal{S}_{\chi,\gamma,\star} \rangle$. We now describe how

Algorithm 10 Heuristic Iterative Proning

```
Input: $B, D, t \in \mathbb{N}$, samples $(\chi, \gamma)$
Output: polynomial system
 1: $\mathcal{K} \subseteq \mathbf{F}_2^{\mathsf{kln}}$ select uniformly at random
 2: $\mathcal{S}_{\chi,\gamma,\star} \leftarrow$ build a polynomial system $\mathcal{S}_{\chi,\gamma,\star}$ corresponding to the cipher
 3: $\mathcal{T} \leftarrow \emptyset$
 4: $\mathcal{U} \leftarrow \emptyset$
 5: $i \leftarrow 0$
 6: select random $\mathcal{K}_0 \subseteq \mathbf{F}_2^{\mathsf{kln}}$
 7: repeat
 8:     compute $\int O \Big(^{a}$ using Notation 61
 9:     select at random a vector space $R \subseteq O + \mathbb{B}[V] + \mathbb{B}[\mathsf{Dup}(V)]$ of dimension at most
        $D$.
10:     $M' \leftarrow R \cap \mathcal{B}_{\chi,\gamma}^{\mathcal{K}_i}$ using Algorithm 7
11:     $\mathcal{T} \leftarrow \mathcal{T} \cup [\![M']\!]_V$
12:     select random $M_i \subseteq$ linspan$(\mathcal{T} \cup O) \setminus \mathcal{U}$ maximal such that $|M_i| \leq B$
13:     $\mathcal{U} \leftarrow M_i \cup \mathcal{U}$
14:     if $|M_i| = 0$ then
15:         $a \leftarrow a + 1$
16:         $b \leftarrow b + 1$
17:     end if
18:     $i \leftarrow i + 1$
19:     select random $\mathcal{K}_i \subseteq \mathcal{K}_{\chi,\gamma}(M_i)$ such that $|\mathcal{K}_i| \geq t \dim(R)$ using Algorithm 11.
20: until the system $\mathcal{T} + \mathcal{S}_{\chi,\gamma,\star}$ can be solved efficiently
21: return $\mathcal{T}$
```

Algorithm 11 Computation of random subset of $\mathcal{K}_{\chi,\gamma}(Q)$

Input: $Q \subseteq \mathbf{F}_2[V, \mathsf{Dup}(V)]$, $n \in \mathbb{N}$
Output: set of keys $\mathcal{K}$ such that $Q \subseteq \mathcal{B}^{\mathcal{K}}_{\chi,\gamma}$ and $|\mathcal{K}| = n$.

```
1:  $\mathcal{K} \leftarrow \emptyset$
2:  **while** $|\mathcal{K}| < m$ **do**
3:      $\kappa \in \mathbf{F}_2^{\mathsf{kln}}$ select randomly uniformly
4:      unset flag
5:      **for** $q \in Q$ **do**
6:          **if** $f_{\chi,\gamma}(q)\,|_{\kappa} \neq 0$ **then**
7:              set flag
8:          **end if**
9:      **end for**
10:     **if** flag is unset **then**
11:         $\mathcal{K} \leftarrow \mathcal{K} \cup \{\kappa\}$
12:     **end if**
13: **end while**
14: return $\mathcal{K}$
```

each technique contributes to speeding up the standard algebraic techniques such as ElimLin/mXL/F4/SAT solvers.

Universal Proning is designed to recover polynomials of the ideal $\langle \mathcal{S}_{\chi,\gamma,\star} \rangle$. I.e, it allows us to recover the ideal $\langle \mathcal{S}_{\chi,\star,\star} \rangle$ and the ideal $\langle \mathcal{S}_{\star,\gamma,\star} \rangle$. The running time of $\mathsf{mXL}\left(\langle \mathcal{S}_{\chi,\star,\star} \rangle, \langle \mathcal{S}_{\star,\gamma,\star} \rangle\right)$ is better than the running time of $\mathsf{mXL}\left(\mathcal{S}_{\chi,\star,\star}, \mathcal{S}_{\star,\gamma,\star}\right)$ as in the first case, we avoid unnecessary restarts when a universal mutant is found. These universal mutants can be found by Universal Proning. We show that these universal mutants actually improve the performance of ElimLin. The samples were selected based on technique introduced in Chapter 4. In our experiments, we considered various versions of reduced round KATAN32 and the vector space R given in Eq. (5.10).

$$R = \mathbf{linspan}\left(\bigcup_{\substack{r \in [30,50] \\ p \in [1,\mathsf{smpn}] \\ j, \in [1,\mathsf{mln}]}} \{\mathsf{s}^j_{p,r}\} \cup \bigcup_{j \in [1,\mathsf{kln}]} \{k_j\} \right). \tag{5.10}$$

We give results in Tables 5.3- 5.7. We denote:

- T_E the time required to compute $\mathsf{ElimLin}\left(\mathcal{S}_{\chi,\gamma,\star}\right)$.
- T_{EP} the time required to compute $\mathsf{ElimLin}\left(\left[\!\left[\mathcal{B}^{\mathcal{K}}_{\chi,\gamma} \cap R\right]\!\right]_V + \mathcal{S}_{\chi,\gamma,\star}\right)$.

Table 5.3: Comparison of attacks on 67-round KATAN32 using ElimLin and Universal Proning

m	t	L_E	L_{EP}	T_E	T_{EP}	T_P
0x00003106	0xee39ca21	41	44	48s	54s	45s
0x0000320c	0x501f8002	0	49	44s	44s	51s
0x0000700c	0x25b98002	0	51	39s	30s	47s
0x0000700c	0x9de08802	0	48	35s	112s	53s
0x00007104	0x39d88a02	45	50	35s	37s	48s
0x00007104	0x65f30240	44	44	47s	19s	45s
0x00017004	0x58d68920	0	55	43s	109s	47s
0x00043404	0x8a10c862	0	52	36s	33s	53s
0x00043404	0x9498080a	0	46	40s	21s	52s
0x02003104	0xa1308860	0	0	41s	91s	45s
0x02007004	0xb1270a48	0	50	56s	70s	53s
0x10007004	0x05ce0020	0	49	57s	49s	48s
0x10013004	0x05d4c022	0	0	44s	168s	52s
0x21003004	0x1414c10a	51	51	51s	26s	49s
0x4000300c	0x884c8a60	0	0	39s	165s	46s
0x40003104	0x12394002	44	45	40s	17s	51s
0x40003804	0x93998022	0	50	36s	61s	54s

- T_P the time required to compute $\left[\!\left[\mathcal{B}^{\mathcal{K}}_{\chi,\gamma}\cap R\right]\!\right]_V$, $K_{R,\chi} = 10\dim\left(\mathcal{B}^{\mathcal{K}}_{\chi,\gamma}\right)$ and $\mathcal{K}\subseteq\mathbf{F}_2^{\mathsf{kln}}$ was selected uniformly at random such that $|\mathcal{K}| = K_{R,\chi}$.
- $L_E = \left|\int \mathsf{ElimLin}\left(\mathcal{S}_{\chi,\gamma,\star}\right)\cap\mathbf{F}_2[V_K]\Big\rbrace^1\right|$
- $L_{EP} = \left|\int \mathsf{ElimLin}\left(\left[\!\left[\mathcal{B}^{\mathcal{K}}_{\chi,\gamma}\cap R\right]\!\right]_V + \mathcal{S}_{\chi,\gamma,\star}\right)\cap\mathbf{F}_2[V_K]\Big\rbrace^1\right|$
- $V_E = \left|\mathsf{VarElimLin}\left(\mathcal{S}_{\chi,\gamma,\star}\right)\right|$
- $V_{EP} = \left|\mathsf{VarElimLin}\left(\left[\!\left[\mathcal{B}^{\mathcal{K}}_{\chi,\gamma}\cap R\right]\!\right]_V + \mathcal{S}_{\chi,\gamma,\star}\right)\cap\mathbf{F}_2[V_K]\right|$

The cost of Universal Proning in our experiment was higher than the cost of ElimLin. This is due to the fact that our implementation of ElimLin was tuned to keep the polynomial system sparse unlike Universal Proning which was based on reference implementation. In general, the asymptotic memory complexity of ElimLin is $O\left(|V|^2\right)$ and the asymptotic memory complexity of Universal Proning is only $O\left(K_{R,\chi}|V|\right)$. However, we did not observe corresponding improvement of running time in practice.

Empirical results. In Table 5.3, we give results for 67-round KATAN32 for the cube selection of samples. The figures demonstrate that using Universal Proning improves the performance of ElimLin with respect to the number of linear equations in the key variables we recover. Similar results can be observed in Table 5.5 for 70-round KATAN32 and Table 5.6 for 71-round KATAN32. In the case of 72-round KATAN32, we could not derive the linear equation in the key variables for any cube but we observed that Universal Proning allowed us to reduce the number of variables of system Q_T obtained by ElimLin (cf. Algorithm 3 in Chapter 3). The results are given in Table 5.7. Hence, Universal Proning is an efficient method to reduce the number of variables of a polynomial system. This usually improves the running time of mXL/F4. The total running time of ElimLin with Universal Proning is given by the sum $T_{EP} + T_P$. In the case of our implementation of Universal Proning, the running time of ElimLin is always smaller than $T_{EP} + T_P$. However for a chosen-plaintext attack (for instance our cube selection of samples), we can recover most of these polynomials in the preprocessing phase. Hence, we perform Universal Proning only once and ignore the time T_P in the attack required by Universal Proning. In some cases, we obtain $T_E \geq T_{EP}$ while in other cases, we obtain $T_E < T_{EP}$. The inconsistency is a result of our heuristic optimization of ElimLin. In rare cases, it may happen that a linear equation from Universal Proning leads to a substitution which slows down ElimLin. However in general, this method leads to a significant speedup when we consider a large number of samples. When we considered samples $\chi_i \in C_{m,t}$ where for $m = \texttt{0x6200c310}$, $t = \texttt{0x8cdc2002}$ and

$$R = \textbf{linspan}\left(\bigcup_{\substack{r\in[30,50] \\ p\in[1,\mathsf{smpn}] \\ j,j'\in[1,\mathsf{mln}]}} \left\{ \mathsf{s}^{j}_{p,r} \cdot \mathsf{s}^{j'}_{p,r} \right\} \cup \bigcup_{j,j'\in[1,\mathsf{kln}]} \left\{ k_j \cdot k_{j'} \right\} \right),$$

we obtained $T_P = 10\ 414s$ and $T_{EP} = 17\ 243s$ to recover the secret key. However, ElimLin without Universal Proning did not recover any linear equation in key variables in $T_E \leq 100\ 000s$.

Mutant Proning is designed to recover nonuniversal polynomials which would be discovered in early stages of mXL and Iterative Proning is designed to recover nonuniversal polynomials which would be discovered in later stages of mXL. In our experiments, we focused on recovering nonuniversal mutants which are unlikely to be found by ElimLin computation. We set

$$R = \int \mathbf{F}_2[V, \mathsf{Dup}(V)] \Big\rceil^{1} + \int \mathbf{F}_2[W, \mathsf{Dup}(W)] \Big\rceil^{2} + \int \mathbf{F}_2[V_K] \Big\rceil^{2}. \qquad (5.11)$$

Table 5.4: Running time of ElimLin on 68-round KATAN32 with/without Universal Proning

m	t	L_E	L_{EP}	T_E	T_{EP}	T_P
0x02003104	0xa1308860	0	0	93s	68s	134s
0x02007004	0xb1270a48	0	57	109s	75s	153s
0x21003004	0x1414c10a	48	51	207s	65s	157s
0x40003104	0x12394002	0	48	138s	77s	160s
0x40003804	0x93998022	0	0	127s	96s	175s
0x40007004	0x368f036a	0	43	134s	171s	155s

Table 5.5: Success of ElimLin on 70-round KATAN32 with/without Universal Proning

m	t	L_E	L_{EP}	T_E	T_{EP}	T_P
0x000a0c41	0x2975a208	0	56	1413s	263s	342s
0x00420c41	0x182c2280	0	58	969s	202s	344s
0x20020c41	0x1d9d6288	56	61	1341s	351s	343s
0x00060c41	0x01e86280	0	61	904s	668s	343s
0x00041982	0x296ba001	62	63	636s	176s	342s

Table 5.6: Success of ElimLin on 71-round KATAN32 with/without Universal Proning

m	t	L_E	L_{EP}	T_E	T_{EP}	T_P
0x000a0c41	0x2975a208	0	55	602s	11905s	337s
0x00420c41	0x182c2280	0	0	511s	1359s	346s
0x20020c41	0x1d9d6288	0	61	493s	3416s	341s
0x00060c41	0x01e86280	0	0	503s	445s	352s
0x00041982	0x296ba001	0	62	883s	2132s	351s

Table 5.7: Speeding up ElimLin on 72-round KATAN32 with/without Universal Proning

m	t	V_E	V_{EP}	T_E	T_{EP}	T_P
0x000a0c41	0x2975a208	1361	1166	416s	1627s	328s
0x00420c41	0x182c2280	1368	1197	620s	1383s	325s
0x20020c41	0x1d9d6288	1348	1163	558s	2397s	327s
0x00060c41	0x01e86280	1368	1197	453s	500s	342s
0x00041982	0x296ba001	1354	1167	527s	1872s	334s

Table 5.8: Speeding up ElimLin on 75-round KATAN32 with/without Universal Proning

m	t	L_E	L_{EP}	T_E	T_{EP}	T_P
0x6200c310	0x8cdc2002	53	54	272 804s	19 641s	10 218s
0x05030c41	0x503ce288	0	0	573 308s	272 920s	10 644s
0x0220d310	0xf44f2020	0	59	274 277s	20 941s	10 880s
0x4410d210	0x834f21c2	0	73	210 058s	12 938s	11 239s
0x00068c49	0xc7407020	0	74	327 425s	20 450s	10 384s

where

$$W = \bigcup_{\substack{r\in[40,43]\\ p\in[1,\mathsf{smpn}]\\ j\in[1,\mathsf{mln}]}} \{\mathsf{s}^j_{p,r}\}\,.$$

We applied Mutant Proning technique for samples $\chi_i \in C_{m,t}$ for $m = \texttt{0x00003106}$,$t = \texttt{0xee39ca21}$, We selected $\kappa \in \mathbf{F}_2^{\mathsf{kln}}$ uniformly at random and we set $\gamma_i = E_\kappa(\chi_i)$. We managed to recover a nonuniversal mutant m for 85-round KATAN32 such that m was not in $\mathbf{linspan}(Q_{\mathcal{L}} + Q_{\mathcal{T}})$. However, we did not recover sufficient number of these mutants to improve the attack using ElimLin. The comparison with standard tools for Gröbner basis computation was not possible because our polynomial system $\mathcal{S}_{\chi,\gamma,\star}$ was too large and both polybori-0.8.0 and XL [Cou10] crashed. Actually, for R given in Eq. (5.11), we managed to recover a mutant $m \in [\![\mathcal{B}_{\chi,\gamma} \cap R]\!]_V$ which had only two monomials. We also applied a variant of Mutant Proning and recovered directly the polynomials in the key variables[2]. Using this variant of Mutant Proning against toy version of KATAN32 where the entropy of the secret key was reduced to 15 bits, we could obtain linear equations in key variables using Mutant Proning. Afterwards, we derived the secret key by solving the linear system using Gauss elimination. In our case, the complexity of this approach

[2]This method was introduced at the beginning of Section 5.3.

was close to an exhaustive search. This was also due to our selection of R at the Step 1 of Algorithm 9. We expect that a more sophisticated selection of R would lead to better results.

To conclude, the Proning techniques allow to speed-up algebraic attacks. It allows us to recover some hidden polynomials of the polynomial system representing the cipher. We observe a significant increase of performance for ElimLin especially for large number of samples. As ElimLin is a preprocessing step for F4/mXL and some SAT solvers, we expect similar results for these algorithms.

6 Conclusion

In Chapter 3, we revisited the ElimLin algorithm. As a result, we developed an optimized version of the algorithm which allowed us to perform algebraic cryptanalysis of reduced round KATAN32, LBlock and SIMON. In Chapter 4, we considered several strategies for selection of samples in algebraic attacks, and we showed that a selection strategy based on cube attacks allows us to break higher number of rounds for all tested ciphers. In [SSV14], we predicted possible advances of ElimLin for TRIVIUM which was later shown in [QW14]. These results show that selection of samples in algebraic attacks is very important and it can lead to significant improvements.

In Chapter 5, we developed a new method for solving a polynomial system arising from deterministic symmetric cipher. We verified that our method recovers mutants more efficiently than other more general algebraic techniques (such as ElimLin, mXL/F4) when we can sufficiently restrict the space where we look for these mutants. We used Universal Proning to find many linear equations which would also be found by ElimLin. We recovered them using Universal Proning at a reduced cost. This speeded-up computation of ElimLin as we reduced the number of iterations performed by ElimLin and similar results are expected for mXL/F4 and SAT solvers.

The cost of Universal Proning, Mutant Proning and Iterative Proning is reduced when we consider only a small set of polynomials R where we perform the proning. In our experiments, the selection of R was based on an ad-hoc strategy and the selection of samples was based on cube attacks. However, we expect that linear, differential and high order differential techniques may be beneficial in the selection of samples and in the selection of the set R.

7

List of Symbols

level: Definition 3, page 11

mutant: Definition 4, page 11

$\textbf{Mac}_F(d)$**:** Definition 2, page 10

$\textbf{mXL}_D$**:** Algorithm 2, page 13

$\mathbb{B}$**:** Definition 7, page 16

$\textbf{FieldEq}[V]$**:** Definition 8, page 16

kln, mln, smpn, rndn: Notation 9, page 16

$\mathbf{s}^j_{p,r}$, k_i, k^r_i**:** Notation 10, page 16

$\textbf{PT}, \textbf{CT}, V_K, V_S, \overline{K}, V$**:** Notation 11, page 17

$\mathcal{S}, \mathcal{S}_{\chi,\star,\kappa}, \mathcal{S}_{\star,\gamma,\kappa}, \mathcal{S}_{\chi,\gamma,\star}, \mathcal{S}_{\chi,\gamma,\kappa}, \mathcal{S}_{\chi,\star,\star}, \mathcal{S}_{\chi,\star,\star}, \mathcal{S}_{\star,\star,\kappa}$**:** Notation 12, page 17

open-ended system: Notation 12, page 17

$\textbf{Eval}_{\chi,\gamma,\star}$**:** Definition 13, page 18

$\lmoustache\rmoustache$**:** Notation 17, page 19

Dup: Definition 18, page 19

$[\![\,]\!]_V$**:** Notation 19, page 20

$\textbf{Var}(q)$**:** Notation 20, page 20

Bibliography

[ABH10] Martin Albrecht, Gregory Bard, and William Hart. Algorithm 898: Efficient multiplication of dense matrices over gf(2). *ACM Trans. Math. Softw.*, 37(1):9:1–9:14, January 2010. Cited on page 13

[ABP11] Martin R. Albrecht, Gregory V. Bard, and Clément Pernet. Efficient Dense Gaussian Elimination over the Finite Field with Two Elements, November 2011. Cited on page 13

[ACFP11] Martin R. Albrecht, Carlos Cid, Jean-Charles Faugère, and Ludovic Perret. On the Relation Between the Mutant Strategy and the Normal Selection Strategy in Gröbner Basis Algorithms. *IACR Cryptology ePrint Archive*, 2011:164, 2011. Cited on page 8, 12

[ADMS09] Jean-Philippe Aumasson, Itai Dinur, Willi Meier, and Adi Shamir. Cube testers and key recovery attacks on reduced-round MD6 and Trivium. In Orr Dunkelman, editor, *FSE 2009*, volume 5665 of *LNCS*, pages 1–22, Leuven, Belgium, February 22–25, 2009. Springer, Berlin, Germany. Cited on page 54, 55

[AFI$^+$04] Gwénolé Ars, Jean-Charles Faugère, Hideki Imai, Mitsuru Kawazoe, and Makoto Sugita. Comparison between XL and Gröbner basis algorithms. In Pil Joong Lee, editor, *ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 338–353, Jeju Island, Korea, December 5–9, 2004. Springer, Berlin, Germany. Cited on page 11

[AHDHS07] Sultan Al-Hinai, Ed Dawson, Matthew Henricksen, and Leonie Ruth Simpson. On the security of the LILI family of stream ciphers against algebraic attacks. In Josef Pieprzyk, Hossein Ghodosi, and Ed Dawson, editors, *ACISP 07*, volume 4586 of *LNCS*, pages 11–28, Townsville, Australia, July 2–4, 2007. Springer, Berlin, Germany. Cited on page 4

[ALM$^+$11] Jean-Philippe Aumasson, Gaëtan Leurent, Willi Meier, Florian Mendel, Nicky Mouha, Raphael C.-W. Phan, Yu Sasaki, and Petr Susil. Tuple cryptanalysis of ARX with application to BLAKE and Skein. In *ECRYPT2 Hash*, 2011. Cited on page 5

[ALRSS11] Shekh Faisal Abdul-Latip, Mohammad Reza Reyhanitabar, Willy Susilo, and Jennifer Seberry. Extended cubes: enhancing the cube attack by extracting low-degree non-linear equations. In Bruce S. N. Cheung, Lucas Chi Kwong Hui, Ravi S. Sandhu, and Duncan S. Wong, editors, *ASIACCS 11*, pages 296–305, Hong Kong, China, March 22–24, 2011. ACM Press. Cited on page 4, 54

[AVBP11] Martin R. Albrecht, Gregory V. Bard, and Clement Pernet. Efficient Dense Gaussian Elimination over the Finite Field with Two Elements. Research report, 2011. Cited on page 13

[BB11] Stanislav Bulygin and Johannes Buchmann. Algebraic cryptanalysis of the round-reduced and side channel analysis of the full PRINTCipher-48. In Dongdai Lin, Gene Tsudik, and Xiaoyun Wang, editors, *CANS 11*, volume 7092 of *LNCS*, pages 54–75, Sanya, China, December 10–12, 2011. Springer, Berlin, Germany. Cited on page 14

[BBD$^+$10] Johannes Buchmann, Stanislav Bulygin, Jintai Ding, Wael Said Abd Elmageed Mohamed, and Fabian Werner. Practical algebraic cryptanalysis for dragon-based cryptosystems. In Swee-Huay Heng, Rebecca N. Wright, and Bok-Min Goi, editors, *CANS 10*, volume 6467 of *LNCS*, pages 140–155, Kuala Lumpur, Malaysia, December 12–14, 2010. Springer, Berlin, Germany. Cited on page 4

[BCDM10] Johannes Buchmann, Daniel Cabarcas, Jintai Ding, and Mohamed Saied Emam Mohamed. Flexible partial enlargement to accelerate Gröbner basis computation over f_2. In Daniel J. Bernstein and Tanja Lange, editors, *AFRICACRYPT 10*, volume 6055 of *LNCS*, pages 69–81, Stellenbosch, South Africa, May 3–6, 2010. Springer, Berlin, Germany. Cited on page 12

[BCN$^+$10] Gregory V. Bard, Nicolas Courtois, Jorge Nakahara, Pouyan Sepehrdad, and Bingsheng Zhang. Algebraic, AIDA/cube and side channel analysis of KATAN family of block ciphers. In Guang Gong and Kishan Chand Gupta, editors, *INDOCRYPT 2010*, volume 6498 of *LNCS*, pages 176–196, Hyderabad, India, December 12–15, 2010. Springer, Berlin, Germany. Cited on page 8, 54, 59, 60

[BD03] Alex Biryukov and Christophe De Cannière. Block ciphers and systems of quadratic equations. In Thomas Johansson, editor, *FSE 2003*, volume 2887 of *LNCS*, pages 274–289, Lund, Sweden, February 24–26, 2003. Springer, Berlin, Germany. Cited on page 7

[BDG$^+$09] Michael Brickenstein, Alexander Dreyer, Gert-Martin Greuel, Markus Wedler, and Oliver Wienand. New developments in the theory of groebner bases and applications to formal verification. *Journal of Pure and Applied Algebra*, 213(8):1612 – 1635, 2009. Theoretical Effectivity and Practical Effectivity of Groebner Bases. Cited on page 20

[BDM14] Kim Batselier, Philippe Dreesen, and Bart De Moor. On the null spaces of the macaulay matrix. *Linear Algebra and its Applications*, 460(0):259 – 289, 2014. Cited on page 10

[BDN$^+$10] Stéphane Badel, Nilay Dagtekin, Jorge Nakahara, Khaled Ouafi, Nicolas Reffé, Pouyan Sepehrdad, Petr Susil, and Serge Vaudenay. ARMADILLO: A multi-purpose cryptographic primitive dedicated to hardware. In Stefan Mangard and François-Xavier Standaert, editors, *CHES 2010*, volume 6225 of *LNCS*, pages 398–412, Santa Barbara, California, USA, August 17–20, 2010. Springer, Berlin, Germany. Cited on page 5

[BFP09] Luk Bettale, Jean-Charles Faugère, and Ludovic Perret. Hybrid approach for solving multivariate systems over finite fields. *JOURNAL OF MATHEMATICAL CRYPTOLOGY*, pages 177–197, 2009. Cited on page 72

[BFP12] Luk Bettale, Jean-Charles Faugère, and Ludovic Perret. Solving polynomial systems over finite fields: Improved analysis of the hybrid approach. In *Proceedings of the 37th International Symposium on Symbolic and Algebraic Computation*, ISSAC '12, pages 67–74, New York, NY, USA, 2012. ACM. Cited on page 72

[BFSS13] Magali Bardet, Jean-Charles Faugère, Bruno Salvy, and Pierre-Jean Spaenlehauer. On the Complexity of Solving Quadratic Boolean Systems. *Journal of Complexity*, 29(1):53–75, February 2013. Cited on page 72

[Bos12] S. Bosch. *Algebraic Geometry and Commutative Algebra*. Universitext. Springer London, 2012. Cited on page 77

[BS91] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, 4(1):3–72, 1991. Cited on page 2

[BSS$^+$13] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK families of lightweight block ciphers. Cryptology ePrint Archive, Report 2013/404, 2013. `http://eprint.iacr.org/2013/404` . Cited on page 62, 63

[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. *J. Symb. Comput.*, 41(3-4):475–511, 2006. Cited on page 8, 9

[Can06] Christophe Cannière. Trivium: A Stream Cipher Construction Inspired by Block Cipher Design Principles. In SokratisK. Katsikas, Javier López, Michael Backes, Stefanos Gritzalis, and Bart Preneel, editors, *Information Security*, volume 4176 of *Lecture Notes in Computer Science*, pages 171–186. Springer Berlin Heidelberg, 2006. Cited on page 54

[CB07] Nicolas Courtois and Gregory V. Bard. Algebraic cryptanalysis of the data encryption standard. In Steven D. Galbraith, editor, *11th IMA International Conference on Cryptography and Coding*, volume 4887 of *LNCS*, pages 152–169, Cirencester, UK, December 18–20, 2007. Springer, Berlin, Germany. Cited on page 4, 24

[CBW07] Nicolas T. Courtois, Gregory V. Bard, and David Wagner. Algebraic and slide attacks on keeloq. Cryptology ePrint Archive, Report 2007/062, 2007. `http://eprint.iacr.org/2007/062` . Cited on page 13

[CBW08] Nicolas Courtois, Gregory V. Bard, and David Wagner. Algebraic and slide attacks on KeeLoq. In Kaisa Nyberg, editor, *FSE 2008*, volume 5086 of *LNCS*, pages 97–115, Lausanne, Switzerland, February 10–13, 2008. Springer, Berlin, Germany. Cited on page 4

[CD08] Nicolas Courtois and Blandine Debraize. Algebraic description and simultaneous linear approximations of addition in Snow 2.0. In Liqun Chen, Mark Dermot Ryan, and Guilin Wang, editors, *ICICS 08*, volume 5308 of *LNCS*, pages 328–344, Birmingham, UK, October 20–22, 2008. Springer, Berlin, Germany. Cited on page 24

[CKPS00] Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 392–407, Bruges, Belgium, May 14–18, 2000. Springer, Berlin, Germany. Cited on page 11

[CL05] Carlos Cid and Gaëtan Leurent. An analysis of the XSL algorithm. In Bimal K. Roy, editor, *ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 333–352, Chennai, India, December 4–8, 2005. Springer, Berlin, Germany. Cited on page 8, 14

[CM03] Nicolas Courtois and Willi Meier. Algebraic attacks on stream ciphers with linear feedback. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 345–359, Warsaw, Poland, May 4–8, 2003. Springer, Berlin, Germany. Cited on page 4

[CMS$^+$14] Nicolas Courtois, Theodosis Mourouzis, Guangyan Song, Pouyan Sepehrdad, and Petr Susil. Combined algebraic and truncated differential cryptanalysis on reduced-round simon. In Mohammad S. Obaidat, Andreas Holzinger, and Pierangela Samarati, editors, *SECRYPT 2014 - Proceedings of the 11th International Conference on Security and Cryptography, Vienna, Austria, 28-30 August, 2014*, pages 399–404. SciTePress, 2014. Cited on page 6, 41, 51, 62

[Cop93] Don Coppersmith. Solving linear equations over gf(2): block lanczos algorithm. *Linear Algebra and its Applications*, 192(0):33 – 60, 1993. Cited on page 13

[Cou01] Nicolas Courtois. The security of hidden field equations (HFE). In David Naccache, editor, *CT-RSA 2001*, volume 2020 of *LNCS*, pages 266–281, San Francisco, CA, USA, April 8–12, 2001. Springer, Berlin, Germany. Cited on page 4

[Cou03] Nicolas Courtois. Fast algebraic attacks on stream ciphers with linear feedback. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 176–194, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Berlin, Germany. Cited on page 4

[Cou04a] Nicolas Courtois. Algebraic attacks on combiners with memory and several outputs. In Choonsik Park and Seongtaek Chee, editors, *ICISC 04*, volume 3506 of *LNCS*, pages 3–20, Seoul, Korea, December 2–3, 2004. Springer, Berlin, Germany. Cited on page 4

[Cou04b] Nicolas Courtois. Algebraic attacks over GF(2^k), application to HFE challenge 2 and Sflash-v2. In Feng Bao, Robert Deng, and Jianying Zhou, editors, *PKC 2004*, volume 2947 of *LNCS*, pages 201–217, Singapore, March 1–4, 2004. Springer, Berlin, Germany. Cited on page 4

[Cou06] Nicolas T. Courtois. How fast can be algebraic attacks on block ciphers ? Cryptology ePrint Archive, Report 2006/168, 2006. http://eprint.iacr.org/ . Cited on page 24, 37

[Cou08] Nicolas T. Courtois. A new frontier in symmetric cryptanalysis. Invited talk, Indocrypt, 2008. http://www.nicolascourtois.com/papers/front_indocrypt08_2p.pdf . Cited on page 73

[Cou10] Nicolas T. Courtois. Tools for Experimental Algebraic Cryptanalysis. http://www.cryptosystem.net/aes/tools.html , 2010. Cited on page 15, 36, 41, 103

[CP02] Nicolas Courtois and Josef Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 267–287, Queenstown, New Zealand, December 1–5, 2002. Springer, Berlin, Germany. Cited on page 8

[CSSV12] Nicolas T. Courtois, Pouyan Sepehrdad, Petr Susil, and Serge Vaudenay. ElimLin algorithm revisited. In Anne Canteaut, editor, *FSE 2012*, volume 7549 of *LNCS*, pages 306–325, Washington, DC, USA, March 19–21, 2012. Springer, Berlin, Germany. Cited on page 5, 23, 30, 41, 42, 55

[CYK09] Jiali Choy, Huihui Yap, and Khoongming Khoo. An analysis of the compact XSL attack on BES and embedded SMS4. In Juan A. Garay, Atsuko Miyaji, and Akira Otsuka, editors, *CANS 09*, volume 5888 of *LNCS*, pages 103–118, Kanazawa, Japan, December 12–14, 2009. Springer, Berlin, Germany. Cited on page 8

[Des77] Des. Data encryption standard. In *In FIPS PUB 46, Federal Information Processing Standards Publication*, pages 46–2, 1977. Cited on page 2

[DF04] D. S. Dummit and R. M. Foote. *Abstract Algebra*. Wiley, third edition, 2004. Cited on page 14, 77

[DH11] Jintai Ding and Timothy J. Hodges. Inverting HFE systems is quasi-polynomial for all fields. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 724–742, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Berlin, Germany. Cited on page 4

[DHN$^+$07] Jintai Ding, Lei Hu, Xuyun Nie, Jianyu Li, and John Wagner. High order linearization equation (HOLE) attack on multivariate public key cryptosystems. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007*, volume 4450 of *LNCS*, pages 233–248, Beijing, China, April 16–20, 2007. Springer, Berlin, Germany. Cited on page 4

[DPD12] Jintai Ding, Yanbin Pan, and Yingpu Deng. An algebraic broadcast attack against NTRU. In Willy Susilo, Yi Mu, and Jennifer Seberry, editors, *ACISP 12*, volume 7372 of *LNCS*, pages 124–137, Wollongong, NSW, Australia, July 9–11, 2012. Springer, Berlin, Germany. Cited on page 4

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002. Cited on page 3

[DS09a] Itai Dinur and Adi Shamir. Cube attacks on tweakable black box polynomials. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 278–299, Cologne, Germany, April 26–30, 2009. Springer, Berlin, Germany. Cited on page 4, 51, 54

[DS09b] Itai Dinur and Adi Shamir. Side channel cube attacks on block ciphers. Cryptology ePrint Archive, Report 2009/127, 2009. `http://eprint.iacr.org/2009/127` . Cited on page 54

[DS11] Itai Dinur and Adi Shamir. Breaking Grain-128 with dynamic cube attacks. In Antoine Joux, editor, *FSE 2011*, volume 6733 of *LNCS*, pages 167–187, Lyngby, Denmark, February 13–16, 2011. Springer, Berlin, Germany. Cited on page 4, 55

[DSW08] Jintai Ding, Dieter Schmidt, and Fabian Werner. Algebraic attack on HFE revisited. In Tzong-Chen Wu, Chin-Laung Lei, Vincent Rijmen, and Der-Tsai Lee, editors, *ISC 2008*, volume 5222 of *LNCS*, pages 215–227, Taipei, Taiwan, September 15–18, 2008. Springer, Berlin, Germany. Cited on page 4

[EDC09] Jeremy Erickson, Jintai Ding, and Chris Christensen. Algebraic cryptanalysis of SMS4: Gröbner basis attack and SAT attack compared. In Donghoon Lee and Seokhie Hong, editors, *ICISC 09*, volume 5984 of *LNCS*, pages 73–86, Seoul, Korea, December 2–4, 2009. Springer, Berlin, Germany. Cited on page 13, 14

[EP10] Christian Eder and John Perry. F5c: A variant of faugères F5 algorithm with reduced Gröbner bases. *Journal of Symbolic Computation*, 45(12):1442 – 1458, 2010. MEGA2009. Cited on page 10

[Fau99] Jean-Charles Faugère. A new efficient algorithm for computing Grobner bases (F4). *Journal of Pure and Applied Algebra*, 139(13):61 – 88, 1999. Cited on page 8, 10

[FIP01] Federal information processing standards publication (FIPS 197). Advanced Encryption Standard (AES), 2001. Cited on page 3

[FM13] Jean-Charles Faugère and Chenqi Mou. Sparse fglm algorithms. *CoRR*, abs/1304.1238, 2013. Cited on page 11

[FØPG10] Jean-Charles Faugère, Rune Steinsmo Ødegård, Ludovic Perret, and Danilo Gligoroski. Analysis of the MQQ public key cryptosystem. In Swee-Huay Heng, Rebecca N. Wright, and Bok-Min Goi, editors, *CANS 10*, volume 6467 of *LNCS*, pages 169–183, Kuala Lumpur, Malaysia, December 12–14, 2010. Springer, Berlin, Germany. Cited on page 4

[FP10] Jean-Charles Faugère and Ludovic Perret. Algebraic cryptanalysis of curry and flurry using correlated messages. In Feng Bao, Moti Yung, Dongdai Lin, and Jiwu Jing, editors, *Information Security and Cryptology*, volume 6151 of *Lecture Notes in Computer Science*, pages 266–277. Springer Berlin Heidelberg, 2010. Cited on page 54

[FSS14] Jean-Charles Faugère, Pierre-Jean Spaenlehauer, and Jules Svartz. Sparse gröbner bases: The unmixed case. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, ISSAC '14, pages 178–185, New York, NY, USA, 2014. ACM. Cited on page 3, 13, 15

[FV13] P.A. Fouque and T. Vannet. Improving Key Recovery to 784 and 799 rounds of Trivium using Optimized Cube Attacks. *FSE2013*, 2013. Cited on page 54, 55

[HJ12] Cheng-Shen Han and Jie-HongRoland Jiang. When boolean satisfiability meets gaussian elimination in a simplex way. In P. Madhusudan and SanjitA. Seshia, editors, *Computer Aided Verification*, volume 7358 of *Lecture Notes in Computer Science*, pages 410–426. Springer Berlin Heidelberg, 2012. Cited on page 14

[HJM07] Martin Hell, Thomas Johansson, and Willi Meier. Grain; a stream cipher for constrained environments. *Int. J. Wire. Mob. Comput.*, 2(1):86–93, May 2007. Cited on page 54

[Jea02] Jean-Charles Faugère. A New Efficient Algorithm for Computing Gröbner Bases Without Reduction to Zero (F5). In *In: ISSAC 02: Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*, pages 75–83, 2002. Cited on page 8, 10, 11

[JV11] Antoine Joux and Vanessa Vitse. A variant of the f4 algorithm. In Aggelos Kiayias, editor, *CT-RSA 2011*, volume 6558 of *LNCS*, pages 356–375, San Francisco, CA, USA, February 14–18, 2011. Springer, Berlin, Germany. Cited on page 10

[Kal93] Erich Kaltofen. Analysis of coppersmith's block wiedemann algorithm for the parallel solution of sparse linear systems. In Gérard Cohen, Teo Mora, and Oscar Moreno, editors, *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, volume 673 of *Lecture Notes in Computer Science*, pages 195–212. Springer Berlin Heidelberg, 1993. Cited on page 13

[Ker83] Auguste Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, IX:5–83, January 1883. Cited on page 1

[Knu94] Lars R. Knudsen. Truncated and higher order differentials. In Bart Preneel, editor, *FSE'94*, volume 1008 of *LNCS*, pages 196–211, Leuven, Belgium, December 14–16, 1994. Springer, Berlin, Germany. Cited on page 51

[KY10] Abdel Alim Kamal and Amr M. Youssef. Applications of SAT solvers to AES key recovery from decayed key schedule images. Cryptology ePrint Archive, Report 2010/324, 2010. `http://eprint.iacr.org/2010/324` . Cited on page 13

[LJN12] Tero Laitinen, Tommi A. Junttila, and Ilkka Niemelä. Extending clause learning SAT solvers with complete parity reasoning (extended version). *CoRR*, abs/1207.0988, 2012. Cited on page 15

[LK07] Chu-Wee Lim and Khoongming Khoo. An analysis of XSL applied to BES. In Alex Biryukov, editor, *FSE 2007*, volume 4593 of *LNCS*, pages 242–253, Luxembourg, Luxembourg, March 26–28, 2007. Springer, Berlin, Germany. Cited on page 8

[LV99] Richard J. Lipton and Anastasios Viglas. On the complexity of SAT. In *40th FOCS*, pages 459–464, New York, New York, USA, October 17–19, 1999. IEEE Computer Society Press. Cited on page 14

[Mat93] Mitsuru Matsui. Linear cryptoanalysis method for DES cipher. In Tor Helleseth, editor, *EUROCRYPT'93*, volume 765 of *LNCS*, pages 386–397, Lofthus, Norway, May 23–27, 1993. Springer, Berlin, Germany. Cited on page 2, 3

[MCD$^+$09] Mohamed Saied Emam Mohamed, Daniel Cabarcas, Jintai Ding, Johannes Buchmann, and Stanislav Bulygin. MXL3: An efficient algorithm for computing Gröbner bases of zero-dimensional ideals. In Donghoon Lee and Seokhie Hong, editors, *ICISC 09*, volume 5984 of *LNCS*, pages 87–100, Seoul, Korea, December 2–4, 2009. Springer, Berlin, Germany. Cited on page 8, 12

[MDB08] Mohamed Saied Emam Mohamed, Jintai Ding, and Johannes Buchmann. Algebraic cryptanalysis of MQQ public key cryptosystem by MutantXL. Cryptology ePrint Archive, Report 2008/451, 2008. `http://eprint.iacr.org/2008/451` . Cited on page 12

[MDB11] Mohamed Saied Emam Mohamed, Jintai Ding, and Johannes Buchmann. The complexity analysis of the MutantXL family. Cryptology ePrint Archive, Report 2011/036, 2011. `http://eprint.iacr.org/2011/036`. Cited on page 12, 13

[MDBW09] Mohamed Saied Emam Mohamed, Jintai Ding, Johannes Buchmann, and Fabian Werner. Algebraic attack on the MQQ public key cryptosystem. In Juan A. Garay, Atsuko Miyaji, and Akira Otsuka, editors, *CANS 09*, volume 5888 of *LNCS*, pages 392–401, Kanazawa, Japan, December 12–14, 2009. Springer, Berlin, Germany. Cited on page 4

[MM00] Fabio Massacci and Laura Marraro. Logical cryptanalysis as a sat-problem: Encoding and analysis. *In Journal of Automated Reasoning*, 24:165–203, 2000. Cited on page 3

[MMDB08] Mohamed Saied Mohamed, Wael Said Mohamed, Jintai Ding, and Johannes Buchmann. MXL2: Solving Polynomial Equations over GF(2) Using an Improved Mutant Strategy. In *Proceedings of the 2nd International Workshop on Post-Quantum Cryptography*, PQCrypto '08, pages 203–215, Berlin, Heidelberg, 2008. Springer-Verlag. Cited on page 8, 11, 12

[MY92] Mitsuru Matsui and Atsuhiro Yamagishi. A new method for known plaintext attack of FEAL cipher. In Rainer A. Rueppel, editor, *EUROCRYPT'92*, volume 658 of *LNCS*, pages 81–91, Balatonfüred, Hungary, May 24–28, 1992. Springer, Berlin, Germany. Cited on page 2

[MZ06] Ilya Mironov and Lintao Zhang. Applications of SAT solvers to cryptanalysis of hash functions. Cryptology ePrint Archive, Report 2006/254, 2006. `http://eprint.iacr.org/2006/254` . Cited on page 13

[NSZW09] J. Nakahara, P. Sepehrdad, B. Zhang, and M. Wang. Linear (Hull) and Algebraic Cryptanalysis of the Block Cipher PRESENT. In *CANS*, volume 5888, pages 58–75. Springer, 2009. Cited on page 24

[OA94] Kazuo Ohta and Kazumaro Aoki. Linear cryptanalysis of the fast data encipherment algorithm. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 12–16, Santa Barbara, CA, USA, August 21–25, 1994. Springer, Berlin, Germany. Cited on page 2

[QW14] Frank Quedenfeld and Christopher Wolf. Advanced algebraic attack on trivium. Cryptology ePrint Archive, Report 2014/893, 2014. `http://eprint.iacr.org/`. Cited on page 105

[RM12] Alexander Rostovtsev and Alexey Mizyukin. On boolean ideals and varieties with application to algebraic attacks. *IACR Cryptology ePrint Archive*, 2012:151, 2012. informal publication. Cited on page 72

[Sei02] Charles Seife. Computer security. crucial cipher flawed, cryptographers claim. *Science (New York, N.Y.)*, 297(5590):2193, September 2002. Cited on page 3

[Sep12] Pouyan Sepehrdad. *Statistical and Algebraic Cryptanalysis of Lightweight and Ultra-Lightweight Symmetric Primitives*. PhD thesis, IC, Lausanne, 2012. Cited on page 27

[Sha45] Claude Shannon. A mathematical theory of cryptography. Memorandum MM 45-110-02, Bell Laboratories, Murray Hill, NJ, USA, September 1945. Classified report. Superseded by [Sha49]. Cited on page 119

[Sha49] Claude E. Shannon. Communication theory of secrecy systems. *The Bell System Technical Journal*, 28(4):656–715, October 1949. A footnote on the initial page says: "The material in this paper appeared in a confidential report, 'A Mathematical Theory of Cryptography', dated Sept. 1, 1945 ([Sha45]), which has now been declassified.". Cited on page 1, 119

[SN12] Hadi Soleimany and Kaisa Nyberg. Zero-correlation linear cryptanalysis of reduced-round LBlock. Cryptology ePrint Archive, Report 2012/570, 2012. `http://eprint.iacr.org/2012/570`. Cited on page 56

[Son14] Guangyan Song. Simonspeck. `https://github.com/GSongHashrate/SimonSpeck`, 2014. Cited on page 63

[Soo10] Mate Soos. Cryptominisat 2.5.0. In *SAT Race competitive event booklet*, July 2010. Cited on page 14

[SSV11] Pouyan Sepehrdad, Petr Susil, and Serge Vaudenay. Fast Key Recovery Attack on ARMADILLO1 and Variants. In *Proceedings of CARDIS 2011*, LNCS. Springer, 2011. Cited on page 5

[SSV14] Petr Susil, Pouyan Sepehrdad, and Serge Vaudenay. On selection of samples in algebraic attacks and a new technique to find hidden low degree equations. In Willy Susilo and Yi Mu, editors, *ACISP 14*, volume 8544 of *LNCS*, pages 50–65, Wollongong, NSW, Australia, July 7–9, 2014. Springer, Berlin, Germany. Cited on page 6, 41, 70, 105

[SSVV13] Pouyan Sepehrdad, Petr Susil, Serge Vaudenay, and Martin Vuagnoux. Smashing WEP in a passive attack. In Shiho Moriai, editor, *FSE 2013*, volume 8424 of *LNCS*, pages 155–178, Singapore, March 11–13, 2013. Springer, Berlin, Germany. Cited on page 5

[Ste06] Till Stegers. Faugère's F5 Algorithm Revisited. Cryptology ePrint Archive, Report 2006/404, 2006. `http://eprint.iacr.org/`. Cited on page 10

[SV13] Petr Susil and Serge Vaudenay. Multipurpose cryptographic primitive armadillo3. In Stefan Mangard, editor, *Smart Card Research and Advanced Applications*, volume 7771 of *Lecture Notes in Computer Science*, pages 203–218. Springer Berlin Heidelberg, 2013. Cited on page 5

[SW12] Yu Sasaki and Lei Wang. Comprehensive study of integral analysis on 22-round LBlock. In *ICISC 12*, LNCS, pages 156–169, Seoul, Korea, November 28–30, 2012. Springer, Berlin, Germany. Cited on page 56

[TW10] Enrico Thomae and Christopher Wolf. Solving systems of multivariate quadratic equations over finite fields or: From relinearization to MutantXL. Cryptology ePrint Archive, Report 2010/596, 2010. `http://eprint.iacr.org/2010/596`. Cited on page 12

[Vie07] Michael Vielhaber. Breaking ONE.FIVIUM by AIDA an algebraic IV differential attack. Cryptology ePrint Archive, Report 2007/413, 2007. `http://eprint.iacr.org/2007/413`. Cited on page 4

[Vil97] Gilles Villard. A study of coppersmith's block wiedemann algorithm using matrix polynomials. Technical report, LMC-IMAG, REPORT # 975 IM, 1997. Cited on page 13

[WW14] Yanfeng Wang and Wenling Wu. Improved multidimensional zero-correlation linear cryptanalysis and applications to LBlock and TWINE.

In Willy Susilo and Yi Mu, editors, *ACISP 14*, volume 8544 of *LNCS*, pages 1–16, Wollongong, NSW, Australia, July 7–9, 2014. Springer, Berlin, Germany. Cited on page 56

[WZ11] Wenling Wu and Lei Zhang. LBlock: A lightweight block cipher. In Javier Lopez and Gene Tsudik, editors, *ACNS 11*, volume 6715 of *LNCS*, pages 327–344, Nerja, Spain, June 7–10, 2011. Springer, Berlin, Germany. Cited on page 42, 56

[YCC04] Bo-Yin Yang, Jiun-Ming Chen, and Nicolas Courtois. On asymptotic security estimates in XL and Gröbner bases-related algebraic cryptanalysis. In Javier López, Sihan Qing, and Eiji Okamoto, editors, *ICICS 04*, volume 3269 of *LNCS*, pages 401–413, Malaga, Spain, October 27–29, 2004. Springer, Berlin, Germany. Cited on page 14

[YCY13] JennyYuan-Chun Yeh, Chen-Mou Cheng, and Bo-Yin Yang. Operating degrees for xl vs. f4/f5 for generic mq with number of equations linear in that of variables. In Marc Fischlin and Stefan Katzenbeisser, editors, *Number Theory and Cryptography*, volume 8260 of *Lecture Notes in Computer Science*, pages 19–33. Springer Berlin Heidelberg, 2013. Cited on page 12

[ZGLC] Bo Zhu, Guang Gong, Xuejia Lai, and Kefei Chen. Another view on cube attack, cube tester, aida and higher order differential cryptanalysis. Cited on page 51

Curriculum Vitae

Sušil Petr

Email: susil.petr@gmail.com
Nationality: Czech Republic, EU
Date of birth: 24.10.1983

Education

PhD Candidate – Security and Cryptography Laboratory 2009–2015
Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland

Honours MSc, Mathematical Methods in Information Security 2006–2008
Charles University, Prague, Czech Republic

Study abroad, Information Security 06/2007–11/2007
University of Queensland, Brisbane, Australia

Honours BS, Computer science 2003–2006
Charles University, Prague, Czech Republic

Work Experience

OctopusNews, Prague, Czech Republic 07/2008–07/2009
SW development:
Development of UI and core systems in software for managing TV broadcasting.
SAS Automotive, Prague, Czech Republic 11/2006–11/2007
SW development:

Development of UI and core systems in software for managing serial production in car industry.
Moody's analytics, Brisbane, Australia 09/2007–11/2007
SW development:
Development of automated system for managing quotes in finantial applications.

IT Skills

Main: Java, SQL (MySQL, PostgreSQL), C, C++, HTML/CSS, Linux
Familiar: Java EE, PHP, Python, Haskell, JavaScript, Mac OS X, Windows
Administation: Linux, Kerberos, NFS

Projects

SNF (grant): Design and Analysis of Ultra-lightweight Cryptographic Primitives
Oridao (industry cooperation): Analysis of Multipurpose Cryptographic Primitive for smart cards

Language Skills

Czech (native), **English** (native-like fluency, C2 level), **French** (basic fluency, B1 level), **German** (basic fluency, A2/B1 level)

Teaching experience

Security and Cryptography, Probability and Statistics