Noname manuscript No.
(will be inserted by the editor)

On Selection of Samples in Algebraic Attacks and a New Technique to Find Hidden Low Degree Equations

Petr Sušil · Pouyan Sepehrdad · Serge Vaudenay · Nicolas Courtois

Received: 1.Nov 2014

Abstract The best way of selecting samples in algebraic attacks against block ciphers is not well explored and understood. We introduce a simple strategy for selecting the plaintexts and demonstrate its strength by breaking reduced-round KATAN32, LBlock and SIMON. For each case, we present a *practical* attack on reduced round version which outperforms previous attempts of algebraic cryptanalysis whose complexities were close to exhaustive search. The attack is based on the selection of samples using cube attack and ElimLin which was presented at FSE'12, and a new technique called Universal Proning. In the case of LBlock, we break 10 out of 32 rounds. In KATAN32, we break 78 out of 254 rounds. Unlike previous attempts which break smaller number of rounds, we do not guess any bit of the key and we only use structural properties of the cipher to be able to break a higher number of rounds with much lower complexity. We show that cube attacks owe their success to the same properties and therefore, can be used as a heuristic for selecting the samples in an algebraic attack. The performance of ElimLin is further enhanced by the new Universal Proning technique, which allows to discover linear equations that are not found by ElimLin.

Keywords: algebraic attacks, LBlock, KATAN32, SIMON, ElimLin, cube attack, universal proning, extended proning

Supported by a grant of the Swiss National Science Foundation, 200021_134860/1.

EPFL, Switzerland
E-mail: {petr.susil, pouyan.sepehrdad , serge.vaudenay}@epfl.ch

1 Introduction

Algebraic attacks are a very powerful method for breaking ciphers in low data complexity attacks. This scenario is the most usual in practice. Algebraic cryptanalysis has brought about several important results (see [14,15,16,17,25, 1]). An algebraic attack can be divided into several steps: building a system of equations and finding the solution to the system using an appropriate algorithm. The methods for finding the solution are, however, not sufficiently adapted for algebraic cryptanalysis, which shed a skeptical light on the entire discipline. The attacks mostly report breaking several rounds of a target cipher, but fail to explore scalable strategies for improvements. In this paper, we start filling this gap.

One approach in algebraic cryptanalysis is building a system of linear equations in the key variables using extensive preprocessing, such as cube attacks [5,25,23,26]. Another approach is building a system of multivariate quadratic equations, and solving the system using Gröbner basis computation (F4/F5, XL/mXL), see [28,34, 2,40,39,44], using XSL algorithm, see [20,13, 37,12], or converting the multivariate system into algebraic normal form and running SAT solvers such as in [43]. All these methods usually implement a common step called ElimLin [22].

ElimLin is a simple algorithm which uses linear equations from the linear span of a system for elimination of variables by substitution. It works iteratively until no new linear equation can be found. Using this method we can, in some cases, linearize a large multivariate polynomial system. Since this technique is used as the first step by all advanced techniques a proper understanding of ElimLin algorithm is crucial for further advances in algebraic cryptanalysis.

In this paper, we present evidence that the success of SAT solvers in algebraic attacks depends on the performance of ElimLin algorithm and we expect similar phenomena to occur in the

case of F4 and mXL. We show that the selection of samples based on a cube attack on R round ciphers performs well when breaking $R + \epsilon$ rounds cipher for a small ϵ. We demonstrate this by breaking 10 rounds (out of 32) of LBlock [45] in Section 3.4 and 78 rounds of KATAN32 (out of 254) [10] without guessing any key bits in Section 5, while all previous approaches were guessing 32 resp. 45 bits of the key. Then in Section 7, we mount the attack against the cipher called SIMON. Therefore, the complexity of their attack is of order $2^{32}\mathsf{T}(\mathsf{SAT})$ resp. $2^{45}\mathsf{T}(\mathsf{SAT})$ where $\mathsf{T}(\mathsf{SAT})$ denotes the average time necessary to break KATAN32. We also note that unlike SAT solvers, whenever ElimLin with our extensions was successful to recover one key, it was successful to recover the key in all cases we tested. The running time of our attack was several hours for smaller sets of samples, and up to 10 days for the largest sets of samples. Finally, we introduce a technique called Universal Proning which allows to find additional linear equations of the system which are satisfied for a random key with high probability.

The relation between algebraic methods have been extensively studied. ElimLin is a basic algorithm which is part of every algebraic tool. XSL is an ad-hoc version of XL which was analyzed in [13]. The XL algorithm computes the Gröbner basis in a similar way as F4, but it performs additional unnecessary computations [4]. The mXL variant of XL [40] is equivalent to F4 [3]. The comparison between Gröbner basis computation and performance of SAT solver was shown in [27]. The complexity of SAT was further studied in [38]. The asymptotic estimates of the complexity of XL and Gröbner basis were given in [46]. The multivariate equations representing the cipher are difficult to solve in general. The most general solving technique is to find the Gröbner basis of the ideal generated by the system using algorithms such as F4. Using this technique, the degree of equations in the system is iteratively increased until the first fall appears [32, Section 4.6], and the system is fully solved, when a so-called degree of regularity is reached [8, Definition 3]. This degree is usually high [7] and therefore such computation is often infeasible due to memory requirements. The SAT solving techniques also do not perform very well for complicated systems. The XL algorithm is a variant of the F4 algorithm [3] and therefore, suffers from the same problems. ElimLin algorithm can be seen as iterations of a Gauss elimination and a substitution. It does not increase the degree of the system in any intermediate step, and hence it finds no solution in many cases. We observe that the running time of all the techniques above depends on the selection of plaintext-ciphertext pairs.

In this paper, we introduce a technique for the selection of samples which significantly improves the running time for selected ciphers. In Section 2, we recall ElimLin algorithm then, in Section 3, we introduce our method for selecting samples in an algebraic attack and show its performance using reduced round LBlock. In Section 4, we discuss implementation improvements of ElimLin, which allow to parallelize the attack and reduce memory requirements. We apply our optimized ElimLin and the cube selection of samples against KATAN32 in Section 5. In Section 6, we introduce a new technique called Universal Proning for recovering linear equations which cannot be found by ElimLin, but which are satisfied for a random key with high probability. We use this technique together with ElimLin to attack reduced round KATAN32 which was previously analysed in [33, 35, 42]. We compare our results to state-of-the-art algebraic attacks on KATAN32 and show that our technique of selecting samples and recovering hidden linear equations outperform previous results. The recent attack against KATAN32 in [42] achieves similar number of rounds as we do but the authors guess 45 statebits before running the SAT. Hence, the complexity of their attack is $2^{45}\mathsf{T}(\mathsf{SAT})$ which is comparable to a previous attack in [6]. In Section 7, we demonstrate the selection of samples against another cipher called SIMON. The results show improvement against selection of samples based on truncated differentials from [19]. In Section 8, we explain the impact of our selection strategy to other algebraic attacks and we conclude in Section 9.

We show the effectiveness of our approach on three well-known ciphers as an example and provide an evidence to support the hypothesis that this would be the case for other ciphers as well. Our sample selection technique can also be used in attacks based on F4/mXL and SAT solvers. The trade-off between increasing number of samples for ElimLin and increasing degree in F4/mXL still remains an open problem.

Algorithm 1: ElimLin

Input : A system of polynomial equations $\mathcal{Q} = \{\mathsf{Eq}_1, \ldots, \mathsf{Eq}_{m_0}\}$ over $\mathbf{F}_2$.
Output : An updated system of equations $\mathcal{Q}_T$ and a system of linear equations $\mathcal{Q}_L$.
Set $\mathcal{Q}_L \leftarrow \emptyset$ and $\mathcal{Q}_T \leftarrow \mathcal{Q}$ and $k \leftarrow 1$.
repeat
    Perform Gauss elimination Gauss(.) on $\mathcal{Q}_T$ with an arbitrary ordering of equations and monomials to eliminate non-linear monomials.
    Set $\mathcal{Q}_{L'} \leftarrow$ Linear equations from Gauss($\mathcal{Q}_T$).
    Set $\mathcal{Q}_T \leftarrow \mathsf{Gauss}(\mathcal{Q}_T) \setminus \mathcal{Q}_{L'}$.
    Set flag.
    for all $\ell \in \mathcal{Q}_{L'}$ in an arbitrary order **do**
        if ℓ is a trivial equation (no variable expressed in ℓ) **then**
            if ℓ is unsolvable **then**
                Terminate and output "No Solution".
            end if
        else
            Unset flag.
            Let x_{t_k} be a monomial from ℓ.
            Substitute x_{t_k} in $\mathcal{Q}_T$ and $\mathcal{Q}_{L'}$ using ℓ.
            Insert ℓ in $\mathcal{Q}_L$.
            $k \leftarrow k + 1$
        end if
    end for
until flag is set.
Output $\mathcal{Q}_T$ and $\mathcal{Q}_L$.

2 The ElimLin algorithm

The ElimLin algorithm is a very simple tool for solving systems of multivariate equations. It is based on iterations of a Gauss elimination and a substitution of variables by linear equations. It is used as a preprocessing tool in most computer algebra systems, e.g., F4/F5 algorithm, XL, or even in cryptominisat. Since this algorithm is a common base of all solvers, it is important to carefully investigate its properties and capabilities. We recall ElimLin algorithm in Algorithm 1 and we refer the reader to [22] for additional details. Later in the paper, we discuss a strategy to improve the running time of ElimLin when we consider many samples. It was already shown in [22] that increasing the number of samples helps to find the secret key using ElimLin. We now show that selecting the plaintexts carefully can significantly improve the performance of ElimLin and even outperforms state-of-the-art attacks based on SAT solvers. Since ElimLin performs only substitution by linear terms, the degree of the system cannot increase. Therefore, ElimLin solves the system and recovers the secret key only in very special cases. ElimLin is performed as the first step of Gröbner basis computation and even some SAT solvers, such as cryptominisat, run ElimLin as a preprocessing. Therefore, we focus on the selection of plaintexts which allows ElimLin to solve the system or eliminate the highest possible number of variables.

3 Selection of samples

In this section, we define our system of equations and give necessary definitions. In part 3.1, we give a new characterization of the system when ElimLin succeeds. In part 3.2, we find a strategy for selection of samples, which allows to satisfy this condition. This selection strategy is based on cube attacks which we recall in part 3.3. In part 3.4, we show the performance of such a technique on LBlock, and compare our results to previous algebraic attacks based on ElimLin. In part 3.5, we give further insight into our method and directions for future testing and improvements.

Notation 1 *We denote* kln *the key length. We denote* sln *the message length and the length of the state vector. We denote* smpn *the number of plaintext/ciphertext pairs. We denote* rndn *the number of rounds of the cipher.*

We represent state bits and key bits by variables. Each state variable $\mathsf{s}^j_{p,r}$ corresponds to a plaintext of index p, a round r, and an index j in the state vector. The key is represented by key variables $k_1, \ldots, k_{\mathsf{kln}}$. The plaintext p is represented by $\mathsf{s}^j_{p,0}$ and ciphertext by $\mathsf{s}^j_{p,\mathsf{rndn}}$.

Definition 2 (Boolean polynomial) *Let $b \in \mathbf{F}_2[V]$ be a polynomial such that*

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

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 $\mathbf{F}_2[V]$.

Notation 3 *We denote the set of variables as* $V = \bigcup_{t \in [1, \mathit{kln}]} \{k_t\} \cup \bigcup_{p \in [1, \mathit{smpn}]} \bigcup_{r \in [0, \mathit{rndn}]} \bigcup_{j \in [1, \mathit{sln}]} \{\mathsf{s}^j_{p,r}\}$.

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

$$\mathsf{r}^j_r(\mathsf{s}^1_{p,r}, \ldots, \mathsf{s}^{\mathsf{sln}}_{p,r}, k_1, \ldots, k_{\mathsf{kln}})$$

We denote the corresponding equation $\mathsf{Eq}^j_{p,r} = \mathsf{r}^j_r\left(\mathsf{s}^1_{p,r}, \ldots, \mathsf{s}^{\mathsf{sln}}_{p,r}, k^r_1, \ldots, k^r_{\mathsf{kln}}\right) - \mathsf{s}^j_{p,r+1}$ where $k^r_j = \mathsf{rk}^r_j(k_1, \ldots, k_{\mathsf{kln}})$.

Notation 4 (system) *We denote*

$$\mathcal{S} = \left(\bigcup_{p \in [1, \mathit{smpn}]} \bigcup_{r \in [0, \mathit{rndn}]} \bigcup_{j \in [1, \mathit{sln}]} \{\mathit{Eq}^j_{p,r}\} \right)$$

The equations are taken over boolean ring, i.e $\mathcal{S} \subseteq \mathbb{B}[V]$, and they represent relations between variables of round r and $r+1$. We further denote

$$\mathcal{S}_{\chi,\star,\star} = \mathcal{S} \cup \bigcup_{p \in [1, \mathit{smpn}]} \bigcup_{j \in [1, \mathit{sln}]} \left(\mathsf{s}^j_{p,0} - \chi^j_p\right),$$

$$\mathcal{S}_{\star,\gamma,\star} = \mathcal{S} \cup \bigcup_{p \in [1, \mathit{smpn}]} \bigcup_{j \in [1, \mathit{sln}]} \left(\mathsf{s}^j_{p,\mathit{rndn}} - \gamma^j_p\right),$$

$$\mathcal{S}_{\star,\star,\kappa} = \mathcal{S} \cup \bigcup_{i \in [1, \mathit{kln}]} \{k_i - \kappa_i\}$$

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. 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 represents, thus, the system in which we fix both the plaintext and the ciphertext.

Notation 5 *For a system*[1] *$\mathcal{S}$, we denote:*

$$\mathcal{S}_{\chi,\star,\kappa} = \mathcal{S}_{\chi,\star,\star} \cup \mathcal{S}_{\star,\star,\kappa}$$

$$\mathcal{S}_{\star,\gamma,\kappa} = \mathcal{S}_{\star,\gamma,\star} \cup \mathcal{S}_{\star,\star,\kappa}$$

$$\mathcal{S}_{\chi,\gamma,\star} = \mathcal{S}_{\chi,\star,\star} \cup \mathcal{S}_{\star,\gamma,\star}$$

We say that $\mathcal{S}_{\chi,\star,\star}$ and $\mathcal{S}_{\star,\gamma,\star}$ are open-ended. Furthermore, we define a ring homomorphism $()_{\chi,\gamma,\star}$ which assigns an element of $\mathbf{F}_2$ to each variable.

Observation 6 *The ideal $\langle \mathcal{S}_{\star,\gamma,\kappa} \rangle$ and $\langle \mathcal{S}_{\chi,\star,\kappa} \rangle$ is 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 7 *We assume that the ideal $\langle \mathcal{S}_{\chi,\gamma,\star} \rangle$ is a maximal ideal.*

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.

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

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

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

Definition 9 (degree of polynomial) *Let $q \in \mathbf{F}_2[V]$ such that $q = \sum_{W \in \mathcal{P}(V)} a_W \prod_{v \in W} v^{e_W}$. We define the degree of multivariate polynomial*

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

Hence based on Definition 9, we consider degree of a polynomial to be the degree of the equivalent boolean polynomial, i.e, for $q \in \mathbf{F}_2[V]$ and $b \in \mathbb{B}[V]$ such that $q = b \bmod \mathsf{Triv}[V]$ we have $\deg q = \deg b$.

Notation 10 *For a set $\mathcal{S} \subset \mathbf{F}_2[V]$ and $D \in \mathbb{N}$ we denote a set $\mathcal{S}^D = \{f : f \in \mathcal{S}, \mathit{deg}(f) \leq D\}$.*

In our experiments, the equations for KATAN32 are build as in [6] and the equations for LBlock as in [22]. This allows for more accurate comparison of our the method of selection of samples.

[1] We assume that our equations are sound in the sense being fully "Describing" equations [18] for each component of the encryption process.

3.1 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 3.2 to derive a selection strategy for plaintexts.
We now reformulate ElimLin (Algo. 1) based on matrix operations. We consider matrices over $\mathbf{F}_2[V]$. The original polynomial system $\mathcal{Q} = \{\mathsf{Eq}_1, \ldots, \mathsf{Eq}_{m_0}\}$ is represented as $\begin{pmatrix} \mathsf{Eq}_1 \\ \vdots \\ \mathsf{Eq}_{m_0} \end{pmatrix}$.
ElimLin performs Gauss elimination and substitution. Gauss elimination (step 6 of Algorithm 1), corresponds to 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 monomial. Let assume substitution of x_1 using linear polynomial $\mathsf{Eq}_1 = x_1 + \ell(\vec{x})$. Assume that $\mathsf{Eq}_2 = x_1 p + q$ where $x_1$does not appear in p and q. Let $x_1 p$ be the substituted term. After substitution, we obtain $\ell(\vec{x})p + q$. Hence, the substitution of x_1 in Eq_2 can be expressed as matrix multiplication

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

We use asociativity and express $\mathsf{ElimLin}(\mathcal{Q})$ by a single matrix multiplication, say $\mathsf{E} \cdot \mathcal{Q}$.

Lemma 11 *Consider a system $\mathcal{S}$ such that 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 ElimLin transformation, i.e, one line of matrix $\mathsf{E} \cdot \mathcal{S}_{\chi,\gamma,\star}$ corresponds to equation $k_j + c_j$. Then, the same line of matrix $\mathsf{E} \cdot \mathcal{S}$ can be written as $k_j + c_j + q'$ and $(q')_{\chi,\gamma,\star} = 0$.*

Proof *We have $\mathsf{E} \cdot (\mathcal{S}_{\chi,\gamma,\star}) = (\mathsf{E} \cdot \mathcal{S})_{\chi,\gamma,\star}$ since $()_{\chi,\gamma,\star}$ is a homomorphism. In the first case, we obtained $k_j + c_j$. Hence in the second case, we have to obtain a polynomial q such that $q_{\chi,\gamma,\star} = k_j + c_j$. Let us consider a polynomial $q = q_{\chi,\gamma,\star} + q'$. Then, q' belongs to kernel of the homomorphism $()_{\chi,\gamma,\star}$, i.e, $(q')_{\chi,\gamma,\star} = 0$.* □

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

3.2 A Selection Strategy for Plaintexts in ElimLin

Lemma 11 characterizes the span of ElimLin when it recovers the value of the key k_j. We now discuss the strategy to ensure that this condition is satisfied. We now consider the polynomial q' from Lemma 11. Since we cannot choose simultaneously the plaintext and the ciphertext for a single sample, we consider several different scenarios: selecting only plaintexts, only ciphertexts, selecting partly plaintexts and partly ciphertexts. The selection of related plaintexts such that corresponding ciphertexts are also somehow related is left as open problem. Such pairs might be constructed using high-order and truncated differential cryptanalysis [36]. 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 11 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 since we cannot control both the plaintexts and the ciphertexts. Hence, we are looking for a set of samples that lead to 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, $\mathsf{s}^r_{1,1}, \ldots, \mathsf{s}^r_{\mathsf{smpn},\mathsf{sln}}$. 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.

3.3 Cube Attack

The cube attack [23] can be seen as a tool to analyze a black-box polynomial which we represent by $f(x,k)$. The aim is to derive a set of equations which is easy to solve and which is satisfied for all keys, i.e, for all values of k. The attacker does this in the offline phase. Afterwards, in the online phase, the attacker finds the evaluation for each equation 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, since 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 $f(x,k) = \sum_{\substack{I\subseteq\{0,1\}^{s\ell n}\\ J\subseteq\{0,1\}^{k\ell n}}} a_{IJ}\prod_{i\in I} x_i \prod_{j\in J} k_j$.

Definition 12 *Let $m,t \in \{0,1\}^{s\ell n}$ 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
$C_{m,t} = \{\begin{smallmatrix}11110111,11100111,11110101,11110011,\\ 11100011,11100001,11100101,11110001\end{smallmatrix}\}$.

Theorem 13 *Let $C_{m,t}$ be a cube and $f(x,k) = \sum_{\substack{I\subseteq\{0,1\}^{s\ell n}\\ J\subseteq\{0,1\}^{k\ell n}}} a_{IJ}\prod_{\substack{i\in I\\ j\in J}} x_i k_j$. Then,*

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

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

Proof *The proof can be found in the Appendix.*

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 $\sum_{x\in C_{m_i,t_i}} f(x,k) = \sum_{J\subseteq\{0,1\}^{k\ell n}} a^i_J \prod_{j\in J} k_j$ are linearly independent low degree equations. Even though cube attack may be a powerful tool in algebraic cryptanalysis, it has been successful against only very few ciphers. The reduced round TRIVIUM [9] can be attacked for 784 and 799 rounds [30], and can be distinguished with 2^{30} samples up to 885 rounds [5]. The full round TRIVIUM has 1152 rounds, which means that 70% of the cipher can be broken by this simple algebraic technique. GRAIN128 [31] was broken using so called dynamic cube attack in [25]. KATAN32 was attacked in [6] using so called side-channel cube attack first introduced in [24]. While cube attacks celebrate success in only few cases, we show that they can be used for selection of samples in other algebraic attacks.

3.4 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 3.1. We give an attack against 10 rounds of LBlock. This attack outperforms the previous attempts of algebraic cryptanalysis [22]. We compare our strategy of using samples for cube attack to the strategy of selecting a random cube or a random set of samples. The strategy of selecting a random cube was previously explored in [29]. The authors were choosing correlated messages based on a algebraic-high order differential.

Breaking 8 rounds of LBlock. The previous result on breaking 8 rounds of LBlock using ElimLin required 8 random plaintexts, and guessing 32 bits of the key (out of 80bits). We found that if we select 8 plaintexts based on cube $C_{m,t}$ for $\begin{smallmatrix}\text{m=0x0000000000000007}\\ \text{t=0x0e84fa78338cd9fb0}\end{smallmatrix}$, we break 8 rounds of LBlock without guessing any key bits. We verified this result for 100 random keys and we were able to recover each of the 100 secret keys we tried using ElimLin.

Breaking 10 rounds of LBlock. We found that if we select 16 plaintexts based on cube $C_{m,t}$ for $\substack{\text{m=0x0000000000003600}\\ \text{t=0x0e84fa78338cd89b6}}$, 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 the 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 1.

3.5 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 give additional observations about behavior of ElimLin with our selection of samples.

Structure of the cube. Let E_κ denote the encryption under the key κ, and let 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 since 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 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 k, a plaintext x, a statebit j and r rounds. 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 find 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.

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, since 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.

Comparison of number of attacked rounds by Cube Attacks and ElimLin with 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

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 1: Results on 10-round LBlock

TRIVIUM and GRAIN128, since there already exist cube attacks against a significant number of rounds [30, 25, 5].

4 Optimizing ElimLin

An efficient implementation of ElimLin faces several challenges. For ElimLin to be successful it is necessary to consider a lot of samples. However, a high number of samples leads to an increase in memory requirements. We remind the Theorem 13 from [22] and use the result to split the system into small subsystems corresponding to different plaintext samples and recover most linear equations with small memory requirements.

Definition 14 *Let $\mathcal{Q}$ be the initial set for ElimLin. Let $\mathcal{Q}_T, \mathcal{Q}_L$ be the resulting sets of ElimLin. We call the linear span of $\mathcal{Q}_T \cup \mathcal{Q}_L$ ElimLin span and denote* $\boldsymbol{elspan}(\mathcal{Q}) = \boldsymbol{linspan}(\mathcal{Q}_T \cup \mathcal{Q}_L)$.

Theorem 15 (ElimLin invariant [22]) *The span* $\boldsymbol{elspan}(\mathcal{Q})$ *is invariant with respect to the order of substitutions and Gauss elimination.*

4.1 Algorithm

Since the result of ElimLin does not depend on the order of substitutions, we propose a divide and conquer strategy for handling a large amount of samples. We divide the system $\mathcal{S}$ which consists of multiple samples into small mutually disjoint subsystems $\mathcal{S}_i$ of smaller number of samples and we let for $a, b \in [1, \mathsf{smpn}]$ denote $\mathcal{S}_{[a,b]} = \bigcup_{i \in [a,b]} \mathcal{S}_i$. Therefore, we have $\mathbf{elspan}(\mathcal{S}) = \mathbf{elspan}\left(\mathcal{S}_{[1,\mathsf{smpn}]}\right)$. We compute recursively $A = \mathbf{elspan}\left(\mathcal{S}_{\left[x, \frac{x+y}{2}\right]}\right)$ and $B = \mathbf{elspan}\left(\mathcal{S}_{\left[\frac{x+y}{2}+1, y\right]}\right)$

$$\mathbf{elspan}\left(\mathcal{S}_{[x,y]}\right) = \mathbf{elspan}\left(A \cup B\right)$$

The analysis of intermediate outputs of ElimLin reveals that most linear equations are discovered already in the union of these small systems, i.e, in $\mathbf{elspan}(\mathcal{S}_i \cup \mathcal{S}_j)$. Since the running time and the memory requirements of ElimLin are proportional to the size of the system, we propose an additional tweak of the algorithm. In this tweak, we compute ElimLin for all pairs of subsystems, i.e, for $\mathbf{elspan}(\mathcal{S}_i \cup \mathcal{S}_j)$, and, before performing a merge in the inner node of the tree containing leaves $[a, b]$, we also include all linear equations recovered by ElimLin on pairs of leaf nodes, i.e, $\bigcup_{i,j \in [a,b]} (\mathcal{S}_i \cup \mathcal{S}_j)_L$. Therefore, for n the number of systems S_i, we perform ElimLin $\binom{n}{2}$-times more, but we substantially decrease the memory requirements of ElimLin in each node $[a, b]$. The graphical representation is given in Figure 1.

In the next section, we show the performance of our new version of ElimLin algorithm and give examples of reduced round KATAN32 and sets of plaintexts that allow us to derive the key using ElimLin. All our attacks outperform the best known attacks and they can be performed using a standard computer with sufficient RAM. In our case, the limitation was 40GB of RAM memory. We expect that our results can be improved both in terms of time, memory and data. This requires better implementation of ElimLin and finding a better cube for selection of samples. Therefore we mainly concentrate on successes and failures of ElimLin to recover the secret key. Additionally, we use a method called Universal Proning which we describe in Section 6. This method allows to recover equations among state variables corresponding to different plaintexts which are valid for every key. These additional equations further speed up ElimLin and allow to break more rounds in some cases.

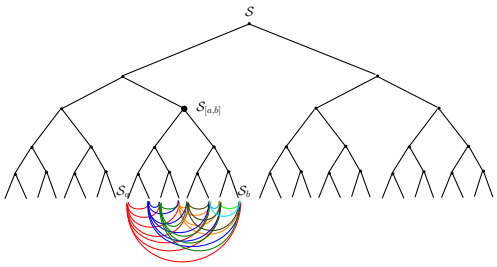

Fig. 1: Divide-Conquer with leaves processing

5 Selection of samples in KATAN32

We give results of the attack against KATAN32 in Table 3. The previous best algebraic attack is given by Bard et al. [6]. The authors attack:

- 79 rounds of KATAN32 using SAT solver, 20 chosen plaintexts and guessing 45 key bits.
- 71 and 75 rounds of KATAN32, and guessing 35-bits of the key.

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. Since 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 2, 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 is successful to recover the key. However, when we use the same mask but a fixed template, ElimLin cannot recover any key bit. We can also see that 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 should also note that the number of linear equations we recover for 70 round KATAN32 in the Universal Proning phase varies for different cubes. In the first case we recover less linear equations by Universal Proning compared to 69 round case, because some linear equations were already recovered by ElimLin. In the second case, ElimLin was unable to recover the new equations appearing in the additional round, but they exist in the ideal, and therefore they can be found by the Universal Proning technique. 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 3, we give successful results of attack by ElimLin applied on reduced round KATAN32 for various number of rounds. Hence, our selection of samples improves state-of-the-art attacks ok KATAN32. We attack 78 rounds of KATAN32 with 1024 samples without guessing any key bit (this corresponds to 79 rounds and guessing two key bits) with running time of 9 days while in [6], the authors need to guess 45 key bits to achieve complexity which is close to brute force.

6 Universal Proning

In this section, we explain how we can recover linear polynomials which are not found by ElimLin and how we can recover those that are recovered faster. We observe that most linear equations (polynomials of degree 1 over $\mathbf{F}_2[V]$) which ElimLin recovers are satisfied independently of the secret key, these are the linear equations in $\mathbf{elspan}\,(\mathcal{S}_{\chi,\star,\star})$ and $\mathbf{elspan}\,(\mathcal{S}_{\star,\gamma,\star})$. Therefore we introduce a new method called Universal Proning for finding all linear equations which are satisfied independently of the value of the key. In this section, we introduce universal polynomials. A universal polynomial is a polynomial $f \in \mathbf{F}_2[V]$, such that $f \in \langle \mathcal{S}_{\chi,\star,\kappa} \rangle$ or $f \in \langle \mathcal{S}_{\star,\gamma,\kappa} \rangle$ for every key κ, hence, the name

Table 2: Attack on KATAN32 using ElimLin: rounds vs. masks

rnd	cube rnd	mask	template	samples	proned lin	success	time
69	55	m=0x00007104	t=0x39d88a02	32	29	10/10	<1 hour
69	55	m=0x00007104	t=0x65f30240	32	29	10/10	<1 hour
69	n.a	m=0x00007104	t=0x00000000	32	35	no	2 hours
69	n.a	m=0x00007104	t=0xf0000000	32	29	no	2 hours
69	n.a	m=0x00007104	t=0x0f000000	32	29	no	2 hours
69	n.a	m=0x00007104	t=0x00f00000	32	29	no	2 hours
70	55	m=0x00007104	t=0x39d88a02	32	27	no	3 hours
70	55	m=0x00007104	t=0x65f30240	32	30	no	3 hours
71	55	m=0x00007105	t=0x23148a40	64	61	10/10	3 hours
71	55	m=0x00007904	t=0x20128242	64	56	no	7 hours
76	59	m=0x0004730c	t=0x21638040	256	572	3/3	3 days

Table 3: Attack on KATAN32 using ElimLin

rnd	cube rnd	mask	template	samples	proned lin	success	time
71	55	m=0x0002700c	t=0xf2b50080	64	116	5/5	<1 hour
70	55	m=0x0c007104	t=0xa2d88a61	128	235	5/5	<1 hour
70	55	m=0x00a07104	t=0x50570043	128	213	5/5	<1 hour
71	55	m=0x00007105	t=0x23148a40	64	61	10/10	3 hours
72	55	m=0x00a07104	t=0x50570043	128	245	20/20	7 hours
72	55	m=0x0c007104	t=0xa2d88a61	128	238	60/60	7 hours
73	55	m=0x0c007104	t=0xa2d88a61	128	217	5/5	7 hours
73	55	m=0x0002d150	t=0x20452820	128	226	20/20	8 hours
73	55	m=0x0002d150	t=0xffd40821	128	231	20/20	8 hours
74	56	m=0x10826048	t=0xca458604	128	212	5/5	9 hours
75	56	m=0x80214630	t=0x76942040	256	538	5/5	23 hours
75	56	m=0x1802d050	t=0x267129a8	256	563	5/5	23 hours
75	56	m=0x908a1840	t=0x6b05c0bd	256	544	5/5	23 hours
75	56	m=0x08030866	t=0x8620f000	256	592	5/5	23 hours
75	56	m=0x52824041	t=0x0d288d08	256	516	5/5	23 hours
75	56	m=0x10027848	t=0xcf758200	256	588	5/5	23 hours
76	59	m=0x0004730c	t=0x21638040	256	572	3/3	3 days
77	59	m=0x03057118	t=0x2cb20001	1024	2376	3/3	8 days
78	59	m=0x03057118	t=0x2cb20001	1024	2381	2/2	9 days

universal. Intuitively, we can see that a universal polynomial cannot help to recover the secret key.

6.1 Universal Proning: Motivation

We demonstrate the method in Table 4 on recovering an algebraic expression of an S-Box which is 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 4 and we find kernel of the matrix M.

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

We interpret the product

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

as equations which hold for any input $x_2x_1x_0$ and therefore they describe the S-box in Algebraic Normal 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 4. The Universal Proning Technique in

	$0 \to 7$	$1 \to 0$	$2 \to 4$	$4 \to 6$		$3 \to 2$	$6 \to 1$	$5 \to 3$	$7 \to 5$
	$000_2 \to 111_2$	$001_2 \to 000_2$	$010_2 \to 100_2$	$100_2 \to 110_2$		$011_2 \to 010_2$	$110_2 \to 001_2$	$101_2 \to 011_2$	$111_2 \to 101_2$
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	M	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 4: Recovering Algebraic Description of an S-Box (07532461).

Section 6.2 aims to recover the ANF of the encryption and the decryption functions using the same approach.

6.2 Universal Proning: Definitions

We now give definitions which allow to extend the previous method to recover ANF of an S-Box. In what follows we consider an extension to a deterministic cipher. In the case of recovering ANF for S-Box, we were not sufficiently formal. We now fill this gap for Universal Proning. Informally, universal polynomials are polynomials of the ring $\mathbf{F}_2[W]$ which are "satisfied" for all values of the secret key. We consider a fixed set of plaintexts χ and a fixed set of ciphertexts $\gamma = E_\kappa(\chi)$ where κ is the unknown secret key. Now we aim to find universal polynomials in variables W. We consider a polynomial ring $\mathbf{F}_2[W]$. Then for each polynomial, we consider the corresponding boolean function from the vector space $\mathsf{Func}\left(\mathbf{F}_2^{\mathsf{kln}}, \mathbf{F}_2\right)$. Such function is obtained through the mapping e_χ, d_γ or $f_{\chi,\gamma}$ as defined in Definition 17. Then using Theorem 20, we obtain that universal polynomials correspond to the zero function from the set $\mathsf{Func}\left(\mathbf{F}_2^{\mathsf{kln}}, \mathbf{F}_2\right)$. Hence, we will recover universal polynomials from the kernel of mapping $f_{\chi,\gamma}$. When we set both plaintext and ciphertext, we have to distinguish variables which are computed by the encryption process and by the decryption process. Hence, we introduce a duplicate set of variables $\mathsf{Dup}(V)$ in Definition 16.

Definition 16 *Let V' be a set of variables such that $V \cap V' = \{k_1, \ldots, k_{kln}\}$ and $|V| = |V'|$. Let us consider a bijective function $\mathsf{Dup} : V \to V'$ where $\mathsf{Dup}(k) = k$ for each $k \in \{k_1, \ldots, k_{kln}\}$ which is homomorphically extended from $\mathbf{F}_2[V]$ to $\mathbf{F}_2[V, \mathsf{Dup}(V)]$.*

Definition 17 *Let us define the mapping $e_\chi : \mathbf{F}_2[V] \to \mathsf{Func}\left(\mathbf{F}_2^{kln}, \mathbf{F}_2\right)$, such that $e_\chi(m)$ maps κ in F_2^{kln} to the reduction of the polynomial m modulo[3] $\langle \mathcal{S}_{\chi,\star,\kappa} \rangle$. We further denote for $\mathcal{K} \subseteq \mathbf{F}_2^{kln}$ the mapping $e_\chi|_\mathcal{K} : \mathbf{F}_2[V] \to \mathsf{Func}(\mathcal{K}, \mathbf{F}_2)$ so that $e_\chi|_\mathcal{K}(q) = e_\chi(q)|_\mathcal{K}$. Similarly, let us define the mapping $d_\gamma : \mathbf{F}_2[\mathsf{Dup}(V)] \to \mathsf{Func}\left(\mathbf{F}_2^{kln}, \mathbf{F}_2\right)$, such that $d_\gamma(m)$ maps κ in F_2^{kln} to the reduction of the polynomial m modulo $\mathsf{Dup}(\langle \mathcal{S}_{\star,\gamma,\kappa} \rangle)$ and we denote $d_\gamma|_\mathcal{K} : \mathbf{F}_2[\mathsf{Dup}(V)] \to \mathsf{Func}(\mathcal{K}, \mathbf{F}_2)$ so that $d_\gamma|_\mathcal{K}(q) = d_\gamma(q)|_\mathcal{K}$. Moreover, let us define $f_{\chi,\gamma} : \mathbf{F}_2[V, \mathsf{Dup}(V)] \to \mathsf{Func}\left(\mathbf{F}_2^{kln}, \mathbf{F}_2\right)$, such that $f_{\chi,\gamma}(m)$ maps κ in F_2^{kln} to the reduction of the polynomial m modulo the ideal $\langle \mathcal{S}_{\chi,\star,\kappa}, \mathsf{Dup}(\mathcal{S}_{\star,\gamma,\kappa}) \rangle_{\mathbf{F}_2[V,\mathsf{Dup}(V)]}$. We denote $f_{\chi,\gamma}|_\mathcal{K} : \mathbf{F}_2[V] \to \mathsf{Func}(\mathcal{K}, \mathbf{F}_2)$ so that $f_{\chi,\gamma}|_\mathcal{K}(q) = f_{\chi,\gamma}(q)|_\mathcal{K}$.*

We select a random subset $\mathcal{K} \subseteq \mathbf{F}_2^{\mathsf{kln}}$ and recover universal polynomials as $\ker\left(e_\chi|_\mathcal{K}\right)$ and $\ker\left(d_\gamma|_\mathcal{K}\right)$. The concept of universal polynomials is closely related to concepts earlier studied in [21, slide 118-120]. We define an ideal which is spanned by two open-ended systems where the relation between plaintext and ciphertext is discarded.

Definition 18 *For every $\kappa \in \mathbf{F}_2^{kln}$, we consider $\mathcal{S}_{\chi,\star,\kappa} \subset \mathbf{F}_2[V]$ and $\mathsf{Dup}(\mathcal{S}_{\star,\gamma,\kappa}) \subset \mathbf{F}_2[V']$*

[3] since $\mathcal{S}_{\chi,\star,\kappa}$ 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.

(with renamed variables). We consider the ring $\mathbf{F}_2[V, V']$ *and define* $\mathcal{B}_{\chi,\gamma} \subset \mathbf{F}_2[V, V']$ *as*

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

We say q *is universal iff* $q \in \mathcal{B}_{\chi,\gamma}$*. Otherwise, we say* q *is nonuniversal. For* $\mathcal{K} \subseteq \mathbf{F}_2^{kln}$*, we define*

$$\mathcal{B}^{\mathcal{K}}_{\chi,\gamma} = \bigcap_{\kappa \in \mathcal{K}} \left(\langle \mathcal{S}_{\chi,\star,\kappa} \rangle + \langle \mathsf{Dup}\left(\mathcal{S}_{\star,\gamma,\kappa}\right) \rangle \right)$$

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

The idea for duplication of variables and building a system such as $\mathcal{B}_{\chi,\gamma}$ was independently developed in [41].

Notation 19 *Let* $q \in \mathbf{F}_2[V, \mathsf{Dup}(V)]$*. Let us consider the unique polynomial* $q' \in \mathbf{F}_2[V]$ *such that* $q = q' \pmod{\langle v + \mathsf{Dup}(v) : v \in V \rangle}$*. We denote the polynomial* q' *as* $[\![q]\!]_V$*.*

Theorem 20 *Let us have plaintext-ciphertext pair* χ, γ *and* $\mathcal{K} \subset \mathbf{F}_2^{kln}$ *such that for the correct key* κ *(such that* $E_\kappa(\chi) = \gamma$*) we have* $\kappa \in \mathcal{K}$*. Then,* $\langle \mathcal{S}_{\chi,\gamma,\star} \rangle = \left[\!\left[\mathcal{B}^{\mathcal{K}}_{\chi,\gamma} \right]\!\right]_V$*. Specifically, we have* $\mathcal{B}_{\chi,\gamma} = \mathsf{ker}(f_{\chi,\gamma})$*.*

Proof *We have* $\langle \mathcal{S}_{\chi,\gamma,\star} \rangle = \left[\!\left[\mathcal{B}^{\{\kappa\}}_{\chi,\gamma} \right]\!\right]_V$*. For the correct key, we have* $\langle \mathcal{S}_{\chi,\gamma,\star} \rangle = \langle \mathcal{S}_{\chi,\gamma,\kappa} \rangle$ *and for the incorrect key, we obtain contradiction, i.e,* $1 \in \langle \mathcal{S}_{\chi,\gamma,\kappa} \rangle$ *and hence,* $\langle \mathcal{S}_{\chi,\gamma,\star} \rangle = \mathbf{F}_2[V]$*. Therefore, the intersection over* $\mathcal{K}$ *which contains the correct key is* $\langle \mathcal{S}_{\chi,\gamma,\star} \rangle$*.* □

Due to Theorem 20, we can compute $\mathcal{B}_{\chi,\gamma}$ resp. $\mathcal{B}^{\mathcal{K}}_{\chi,\gamma}$ as $\mathsf{ker}(f_{\chi,\gamma})$ resp. $\mathsf{ker}(f_{\chi,\gamma|\mathcal{K}})$. I.e, similarly to the approach in Table 4, we build a matrix $M(V, \mathcal{K})$ where the rows represent a variable $v \in V$ (each corresponds to some plaintext) and the columns represent a key $k \in \mathcal{K}$ (the input). The element (i, j) of the matrix is the value of variable (state bit) i for the key j. Afterwards, we find the nullspace of the matrix M. The vectors in the nullspace represent linear equations which hold among the state bits for selected keys.

6.3 ElimLin and Universal Proning

In this section, we reconsider the algorithm from Section 4.1. We consider a split of the system $\mathcal{S}_{\chi,\gamma,\star}$ into $\mathcal{S}_{\chi,\star,\star}$ and $\mathcal{S}_{\star,\gamma,\star}$ and we compute $\mathbf{elspan}\left(\mathcal{S}_{\chi,\star,\star}\right)$ and $\mathbf{elspan}\left(\mathcal{S}_{\star,\gamma,\star}\right)$ using Universal Proning. We can find hidden linear equations of system $\mathcal{S}_{\chi,\star,\star}$ resp. $\mathcal{S}_{\star,\gamma,\star}$ using ElimLin. However,we can also compute these equations using Universal Proning. Actually, we have

$$\mathbf{elspan}\left(\mathcal{S}_{\chi,\star,\star}\right) \subseteq \mathsf{ker}\left(e_\chi\right) \text{ and}$$
$$\mathsf{Dup}\left(\mathbf{elspan}\left(\mathcal{S}_{\star,\gamma,\star}\right)\right) \subseteq \mathsf{ker}\left(d_\gamma\right).$$

We perform these two computations together by computing $\mathsf{ker}\left(f_{\chi,\gamma}\right)$. Then, we perform the back-substitution and we obtain

$$\mathbf{elspan}\left(\mathcal{S}_{\chi,\star,\star}\right) \subseteq [\![\mathsf{ker}\left(f_{\chi,\gamma}\right)]\!]_V .$$

Due to Theorem 20, we have $[\![\mathcal{B}_{\chi,\gamma}]\!]_V = \langle \mathcal{S}_{\chi,\gamma,\star} \rangle$ and hence, we can use equations from the set $[\![\mathcal{B}_{\chi,\gamma}]\!]_V$ to speed-up ElimLin. Furthermore, we reduce the computational complexity of finding $\mathsf{ker}\left(f_{\chi,\gamma}\right)$ as follows. We consider $\mathcal{K} \subseteq \mathbf{F}_2^{kln}$ such that $|\mathcal{K}| \ll |V|$. Then, we want to compute $\mathsf{ker}\left(f_{\chi,\gamma|\mathcal{K}}\right) = \mathcal{B}^{\mathcal{K}}_{\chi,\gamma}$ and obtain new linear equations in the set $\left[\!\left[\mathcal{B}^{\mathcal{K}}_{\chi,\gamma}\right]\!\right]_V$. However, this straightforward approach is not possible; actually, it is easy to show that $1 \in \left[\!\left[\mathcal{B}^{\mathcal{K}}_{\chi,\gamma}\right]\!\right]_V$, i.e, every secret key would be a solution.
Hence, we restrict the ideal $\mathcal{B}^{\mathcal{K}}_{\chi,\gamma}$ to vector space of polynomials of degree at most 1 and denote this $\left(\mathcal{B}^{\mathcal{K}}_{\chi,\gamma}\right)_1$. Then, we select a set of random keys $\mathcal{K}$ such that $|\mathcal{K}| \gg |V|$. This ensures that with a high probability, we have $1 \notin \left[\!\left[\left(\mathcal{B}^{\mathcal{K}}_{\chi,\gamma}\right)_1\right]\!\right]_V$. At the same time, the computation of $\left(\mathcal{B}^{\mathcal{K}}_{\chi,\gamma}\right)_1$ by computation of a kernel of an appropriate matrix is actually less expensive than ElimLin. In case of 71-round KATAN32 and a cube of 64 samples, the Universal Proning required approximatelly $350s$ while the ElimLin required $600s$. We tried such attack with cube selection of samples when ElimLin itself was unable to recover the secret key. We run

$$\mathsf{ElimLin}\left(\mathcal{S}_{\chi,\gamma,\star} + \left[\!\left[\left(\mathcal{B}^{\mathcal{K}}_{\chi,\gamma}\right)_1\right]\!\right]_V\right)$$

and in 60% of cases, we could recover the secret key.

7 Selection of samples in SIMON

We deploy 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 3.3, 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 rank the cube C by the number of black-box polynomials f for which the cubesum is linear for the cube C.

In our experiment, we fixed a secret key and considered 10 round cubes of rank 3 and 4. Then, we build the polynomial system and run ElimLin. Table 5 and Table 6 shows how many polynomials in the key variables was recovered for each cube.

7.1 Limitations of Cube Selection

In the next step, we found 20 cubes of 2^{21} plaintexts of rank 1. We give these cubes in Table 7. Since our implementation of ElimLin is not suitable for systems of 2^{21} plaintext/ciphertext pairs, we selected subcubes of 2^5 and test the preformance of ElimLin against 13 rounds as in Section 5. Even though these subcubes had rank 64, we did not recover any polynomial in key variables. This phenomena is still an open problem.

8 Final remarks on ElimLin

On increasing the degree in F4 and increasing the number of samples in ElimLin
The F4/mXL keeps increasing the degree until the solution is found in the linear span. ElimLin on the other hand requires more plaintext-ciphertext pairs to recover the key. We show that a better selection strategy improves the success of ElimLin, but the question whether the cipher can be broken for a large enough set of well selected samples remains opened. Similarly, we can consider the increase of the number of samples as an alternative to linearization step of F4/mXL. The open problem is whether these strategies are equivalent or if one or the other performs better. However, we believe there is an advantage of considering multiple samples and using a method introduced in Section 6 over increasing the degree and linearization.

Implications for F4/mXL/SAT solvers
Table 2 show that selection of samples influences the degree of regularity of the system. This claim is based on the fact that for some choices of samples (choices of cubes m, t) ElimLin can solve the system. Therefore, the degree of regularity is at most 2. While for other choices it cannot recover the secret key and hence, the degree of regularity is in these cases greater than 2. We compare several strategies for selection of 16 samples for attacking 10-round LBlock. In the first case we select the samples based on a cube attack of 6 rounds. Then, we run ElimLin which successfully recovers a secret key only for subset of these cubes. Subsequently, whenever ElimLin succeeds to recover the secret key for a cube, we perform additional tests with 100 random secret keys and were able to recover the secret key in all cases. In the second case we select samples based on a random cube and obtain a system of 700 variables after ElimLin. In the third case we select samples randomly and obtain a system of 2000 variables after ElimLin. This example shows the importance of selection of samples. The running time of F4/mXL is proportional to the degree of regularity and the number of variables in the system and, therefore, the proper selection of samples is a crucial step. In the case of SAT solvers, the running time depends on the number of restarts performed by the solver and the number of restarts depends on the number of high degree relations.

9 Conclusion

We showed that the offline phase of the cube attack can be used for the selection of samples in other algebraic techniques and that such selection significantly outperforms the random selection of samples. We used this method against reduced round KATAN32, and showed that 78 rounds can be broken only using ElimLin and cube of 2^{10} samples. The approach can be seen as a method of turning a single cube from cube attack into a key recovery technique. Our results highlight several open problems. The strategy of selecting more samples can be seen as an alternative to increasing the degree as it is done by F4/mXL. Using more samples leads to more variables in the system, yet the same goal is achieved by increasing the degree and linearization. Hence, the comparison of our selection of

$C_{m,t}$ system of 2^5 samples		key polynomials recovered
M=0000001400000051	T=91E961A895DDFFAA	8
M=00000028000000A2	T=8F7049C053D5CE00	0
M=0000005000000144	T=4AEC0722CA7CD632	8
M=0000028000000A20	T=5DF80042CD90648F	9
M=0000028000000A20	T=C022AC2273E1818B	9
M=0000050000001440	T=1BB44000FFA88283	4
M=0000050000001440	T=E09C20551A6F0BB6	7
M=00000A0000002880	T=29F2E0A84802D018	8
M=00000A0000002880	T=6CBA814A4D784111	8
M=0000140000005100	T=AEE108C463EDA072	7
M=0000140000005100	T=F8C140111876A869	8
M=000028000000A200	T=92A0520276DD08EE	7
M=000028000000A200	T=ACC006A4FB4E15E0	9
M=000028000000A200	T=F4491689436808E3	6
M=0000A00000028800	T=25A64EA2686516B0	8
M=0000A00000028800	T=A20456140D1077B4	8
M=0001400000051000	T=E91830236128AA78	8
M=00028000000A2000	T=AA192A4B24B483AF	7
M=0005000000144000	T=8D0A65161C88280F	7
M=000A000000288000	T=0A150A7266176D7B	7
M=000A000000288000	T=10558985C513531E	8
M=000A000000288000	T=12152B9F06875130	7
M=000A000000288000	T=4A41CA6F9B5173E7	8
M=000A000000288000	T=8021827B80554735	8
M=0014000000510000	T=8461C1640A087257	9
M=0014000000510000	T=A400D49ABE8A0E33	7
M=0014000000510000	T=B4629121E684C6F6	8
M=0028000000A20000	T=621124BAB25CF6A5	9
M=0050000001440000	T=058F5B37E2915BCF	8
M=0050000001440000	T=12AE464784A89D89	7
M=0050000001440000	T=C5A74459282A67DA	9
M=0500000014400000	T=683089C8CA1E1FD8	9
M=1400000051000000	T=6245E094A44B67EE	7
M=28000000A2000000	T=5286BB0911464FF6	8
M=4000000110000005	T=A04D0812444FC0DA	7
M=5000000044000001	T=285CEDC7A0CD7C14	8
M=5000000044000001	T=2C4C76C6B01A63D2	7
M=800000022000000A	T=70E79C3D8B02C534	7

Table 5: results for rank 3 cubes for 10 round SIMON, attacked 13 rounds

samples for ElimLin and state of the art implementations of XL such as [11,40] is crucial for future directions for algebraic cryptanalysis. During our work we have discovered the existence of exploitable internal low degree relations inside open-ended systems of equations which depend on the plaintext and depend neither on the ciphertext nor the key [21, slide 118]. These additional equations are not always found by ElimLin and we show that our attacks can be enhanced by finding such equations first, which process we call Universal Proning. The fact that the solution is usually found in $\mathbf{elspan}\,(\mathcal{S}_{\chi,\star,\star}) + \mathbf{elspan}\,(\mathcal{S}_{\star,\gamma,\star})$ and the full analysis of Extended Proning is a part of an ongoing research.

References

1. 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.
2. 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.
3. Martin R. Albrecht, Carlos Cid, Jean-Charles Faugère, and Ludovic Perret. On the relation between the MXL family of algorithms and Gröbner basis algorithms. *J. Symb. Comput.*, 47(8):926–941, 2012.

$C_{m,t}$ system of 2^5 samples		key polynomials recovered
M=0000001400000054	T=846870006BF32D29	8
M=00000028000000A8	T=071C214742C05A06	7
M=0000005000000150	T=1EED04016076E803	8
M=0000005000000150	T=5EF38680EF07120B	8
M=0000005000000150	T=5EF6C68922707428	6
M=0000014000000540	T=42EC563DAF599011	6
M=000028000000A800	T=C7A18482AAE8160F	10
M=0000500000015000	T=1A040F0501788339	10
M=0000500000015000	T=7105054FA0348A1F	8
M=0000A0000002A000	T=4A8E1419E3D002F5	6
M=0005000000150000	T=C1426136AEE2397A	10
M=000A0000002A0000	T=42C18178B0857A68	9
M=0014000000540000	T=C041400FD3838E7C	0
M=0028000000A80000	T=885721591251F364	9
M=0050000001500000	T=51AF1118C02D8689	11
M=0050000001500000	T=82A60186AA8E321E	11
M=0050000001500000	T=D0A00F2FDE80FEC4	9
M=0140000005400000	T=5C8909B508B02190	10
M=028000000A800000	T=AC104EF0604D3456	7
M=0500000015000000	T=3A8159DDEA8B307E	9
M=28000000A8000000	T=552E6520033B1F98	11
M=5000000050000001	T=234ED3D6A7DDF6E4	7
M=800000028000000A	T=10A948BC1D9FF684	7
M=800000028000000A	T=7A3C3E184CD06DE0	11
M=A0000000A0000002	T=0AAC7AA5101927F8	10

Table 6: results for rank 4 cubes 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 7: Cubes of 2^{21} samples against 11 round SIMON

4. 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.
5. 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.
6. 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.
7. M. Bardet, J.-C. Faugère, B. Salvy, and B.-Y. Yang. Asymptotic behaviour of the degree of regularity of semi-regular polynomial systems. In *MEGA'05*, 2005. Eighth International Symposium on Effective Methods in Algebraic Geometry, Porto Conte, Alghero, Sardinia (Italy), May 27th – June 1st.
8. Magali Bardet, Jean-Charles Faugère, Bruno Salvy, and Pierre-Jean Spaenlehauer. On the complexity of solving quadratic boolean systems. *J. Complexity*, 29(1):53–75, 2013.
9. 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.
10. Christophe De Cannière, Orr Dunkelman, and Miroslav Knežević. KATAN and KTANTAN - a family of small and efficient hardware-oriented block ciphers. In Christophe Clavier and Kris Gaj, editors, *CHES 2009*, volume 5747 of *LNCS*, pages 272–288, Lausanne, Switzerland, September 6–9, 2009. Springer, Berlin, Germany.
11. Chen-Mou Cheng, Tung Chou, Ruben Niederhagen, and Bo-Yin Yang. Solving quadratic equations with XL on parallel architectures. In Emmanuel Prouff and Patrick Schaumont, editors, *CHES 2012*, volume 7428 of *LNCS*, pages 356–373, Leuven, Belgium, September 9–12, 2012. Springer, Berlin, Germany.
12. 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.
13. 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.
14. Nicolas Courtois. Higher order correlation attacks, XL algorithm and cryptanalysis of toyocrypt. In Pil Joong Lee and Chae Hoon Lim, editors, *ICISC 02*, volume 2587 of *LNCS*, pages 182–199, Seoul, Korea, November 28–29, 2002. Springer, Berlin, Germany.
15. 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.
16. 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.
17. 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.
18. 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.
19. 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.
20. 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.
21. Nicolas T. Courtois. A new frontier in symmetric cryptanalysis. Invited talk, Indocrypt, 2008. `http://www.nicolascourtois.com/papers/front_indocrypt08_2p.p`
22. 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.
23. 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.
24. Itai Dinur and Adi Shamir. Side Channel Cube attacks on Block Ciphers. *IACR Cryptology ePrint Archive*, 2009:127, 2009.
25. 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.
26. Itai Dinur and Adi Shamir. Applying cube attacks to stream ciphers in realistic scenarios. *Cryptography and Communications*, 4(3-4):217–232, 2012.
27. 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.
28. Jean-Charles Faugère. A new efficient algorithm for computing Grobner bases (F4). *Journal of Pure and Applied Algebra*, 139(13):61 – 88, 1999.
29. 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.
30. P.A. Fouque and T. Vannet. Improving Key Recovery to 784 and 799 rounds of Trivium using Optimized Cube Attacks. *FSE2013*.
31. 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.
32. Timothy Hodges, Christophe Petit, and Jacob Schlather. Degree of Regularity for Systems arising from Weil Descent. In *YAC2012 - Yet Another Conference in Cryptography*, 9 2012.
33. Takanori Isobe, Yu Sasaki, and Jiageng Chen. Related-key boomerang attacks on KATAN32/48/64. In Colin Boyd and Leonie Simpson, editors, *ACISP 13*, volume 7959 of *LNCS*, pages 268–285, Brisbane, Australia, July 1–3, 2013. Springer, Berlin, Germany.
34. 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.
35. Simon Knellwolf, Willi Meier, and María Naya-Plasencia. Conditional differential cryptanalysis of Trivium and KATAN. In Ali Miri and Serge Vaudenay, editors, *SAC 2011*, volume 7118 of *LNCS*, pages 200–212, Toronto, Ontario, Canada, August 11–12, 2011. Springer, Berlin, Germany.
36. 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.
37. 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.
38. 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.
39. 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.
40. 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.
41. Alexander Rostovtsev and Alexey Mizyukin. On boolean ideals and varieties with application to algebraic attacks. *IACR Cryptology ePrint Archive*, 2012:151, 2012. informal publication.
42. Ling Song and Lei Hu. Improved algebraic and differential fault attacks on the katan block cipher. In RobertH. Deng and Tao Feng, editors, *Information Security Practice and Experience*, volume 7863 of *Lecture Notes in Computer Science*, pages 372–386. Springer Berlin Heidelberg, 2013.
43. Mate Soos. Cryptominisat 2.5.0. In *SAT Race competitive event booklet*, July 2010.
44. Till Stegers. Faugère's F5 Algorithm Revisited. Cryptology ePrint Archive, Report 2006/404, 2006. `http://eprint.iacr.org/`.
45. 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.
46. 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.

A Additional proofs

Proof *of Theorem 13.*

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

The equality $\star$ is satisfied, since

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

since $\prod$ appears twice for every $i \in I \setminus I_m$. □