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

Petr Sušil*, Pouyan Sepehrdad, and Serge Vaudenay

EPFL, Switzerland
{petr.susil,pouyan.sepehrdad,serge.vaudenay}@epfl.ch

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 and LBlock. In both cases, we present a *practical* attack 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, ElimLin, Gröbner basis, cube attack, universal proning.

1 Introduction

Algebraic attacks is 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 [1, 14–17, 25]). 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.

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

W. Susilo and Y. Mu (Eds.): ACISP 2014, LNCS 8544, pp. 50–65, 2014.

One approach in algebraic cryptanalysis is building a system of linear equations in the key variables using extensive preprocessing, such as cube attacks [5, 23, 25, 26]. Another approach is building a system of multivariate quadratic equations, and solving the system using Gröbner basis computation (F4/F5, XL/mXL), see [2, 28, 34, 39, 40, 43], using XSL algorithm, see [12, 13, 19, 37], or converting the multivariate system into algebraic normal form and running SAT solvers. such as in [42]. 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 [44] in Section 3.4 and 78 rounds of KATAN32 (out of 254) [10] without guessing any key bits in Section 6, while all previous approaches were guessing $32 - 45$ bits of the key. Therefore, the complexity of their attack is of order $2^{32}\mathsf{T}(\mathsf{SAT}) - 2^{45}\mathsf{T}(\mathsf{SAT})$. We also note that unlike SAT solvers, whenever ElimLin with our extensions, which we introduce in Section 5, 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 these 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 [45]. 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. In Section 5, 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 in Section 6 to attack reduced round KATAN32. It was previously analysed in [33, 35, 41]. 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 [41] 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]. We show the effectiveness of our approach on two well-known ciphers as an example and provide 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.

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 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 can solve the system and recover 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 step. Therefore, we focus on the selection of plaintexts which

allows ElimLin to solve the system or eliminate the highest possible number of variables.

3 On the 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 use* kln *to represent the key length. We use* sln *to represent the message length and the length of the state vector. We use* smpn *to represent the number of plaintext/ciphertext pairs. We use* rndn *to represent the number of rounds of the cipher.*

We represent state bits and key bits by variables. Each state variable $\mathsf{s}_{p,r}^{j}$ 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 $\mathsf{s}_{p,0}^{j}$ and ciphertext by $\mathsf{s}_{p,\mathsf{rndn}}^{j}$.

Notation 2. *We denote* $V = \bigcup_{t\in[1,\mathsf{kln}]} \{k_t\} \cup \bigcup_{p\in[1,\mathsf{smpn}]} \bigcup_{r\in[0,\mathsf{rndn}]} \bigcup_{j\in[1,\mathsf{sln}]} \{\mathsf{s}_{p,r}^{j}\}$ *the set of variables.*

The round function of the cipher is represented by a set of polynomials r_r^j which takes as input all state variables at round r and returns the j-th state variable at round $r+1$, i.e., $\mathsf{s}_{p,r+1}^{j}$ is given by polynomial $\mathsf{r}_r^j(\mathsf{s}_{p,r}^{1}, \ldots, \mathsf{s}_{p,r}^{\mathsf{sln}}, k_1, \ldots, k_{\mathsf{kln}})$. We denote corresponding equation[1] $\mathsf{Eq}_{j,r}^{p} = \mathsf{r}_r^j\left(\mathsf{s}_{p,r}^{1}, \ldots, \mathsf{s}_{p,r}^{\mathsf{sln}}, k_1, \ldots, k_{\mathsf{kln}}\right) - \mathsf{s}_{p,r+1}^{j}$.

Notation 3 (system). *We denote*

$$\mathcal{S} = \left(\bigcup_{p\in[1,\mathsf{smpn}]} \bigcup_{r\in[0,\mathsf{rndn}]} \bigcup_{j\in[1,\mathsf{sln}]} \{\mathsf{Eq}_{j,r}^{p}\} \right) \cup \left(\bigcup_{v\in V} \{v^2 - v\} \right)$$

where the first part represents equations between variables of round r *and* $r+1$ *and the second part represents equations which hold trivially over* $\mathbf{F}_2$*. We further denote* $\mathcal{S}_{\omega,\star,\star} = \mathcal{S} \cup \bigcup_{p\in[1,\mathsf{smpn}]} \bigcup_{j\in[1,\mathsf{sln}]} \left(\mathsf{s}_{p,0}^{j} - \omega_p^j\right)$, $\mathcal{S}_{\star,\star,\kappa} = \mathcal{S} \cup \bigcup_{i\in[1,\mathsf{kln}]} \{k_i - \kappa_i\}$

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

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

We use notation $\mathcal{S}_{\omega,\gamma,\kappa}$ to denote that we set plaintext to ω, ciphertext to γ and key to κ. The symbol $\star$ at any position means that the value is unset. Hence, $\mathcal{S}_{\omega,\star,\star}$ 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 $\mathcal{S}_{\omega,\gamma,\star}$ which represents, thus, the system in which we fix both the plaintext and the ciphertext.

Notation 4. *For a system $\mathcal{S}$, we denote $\mathcal{S}_{\omega,\star,\kappa} = \mathcal{S}_{\omega,\star,\star} \cup \mathcal{S}_{\star,\star,\kappa}$, $\mathcal{S}_{\star,\gamma,\kappa} = \mathcal{S}_{\star,\gamma,\star} \cup \mathcal{S}_{\star,\star,\kappa}$, and $\mathcal{S}_{\omega,\gamma,\star} = \mathcal{S}_{\omega,\star,\star} \cup \mathcal{S}_{\star,\gamma,\star}$*

Assumption 5. *We assume that the ideal $\langle \mathcal{S}_{\omega,\gamma,\star} \rangle$ 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. 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.

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.

Lemma 6. *Consider a system $\mathcal{S}$ such that ElimLin applied to $\mathcal{S}_{\omega,\gamma,\star}$ recovers the key bit k_j as value $c_j \in \mathbf{F}_2$. Let ElimLin$'$ be a variant of ElimLin which treats plaintext and ciphertext variables of the system $\mathcal{S}$ as if they had degree 0. Then, $\exists q \in$ **elspan'**$(\mathcal{S})$ which has the following form: $q = k_j + c_j + q'$ and q' evaluates to 0 when we set plaintext variables to ω and ciphertext variables to γ.*

Proof. *We perform the same substitution while running ElimLin$'$ and obtain the polynomial q'.*

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

3.2 A Selection Strategy for Plaintexts in ElimLin

Lemma 6 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 6. Since 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 [21]. These pairs are constructed using higher order and/or 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 6 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 ω 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., $\mathsf{s}^r_{1,1}, \ldots, \mathsf{s}^r_{\mathsf{smpn,sln}}$. Provided the $\mathsf{deg}_0(q') < d$, we can build a set of 2^D samples, i.e., find ω, such that q' evaluates to 0. This leads us to setting values ω 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. Throughout the paper, we represent this polynomial 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 $f(x, k) = \sum_{\substack{I \subseteq \{0,1\}^{\mathsf{sln}} \\ J \subseteq \{0,1\}^{\mathsf{kln}}}} a_{IJ} \prod_{i \in I} x_i \prod_{j \in J} k_j$.

Definition 7. *Let $m \in \{0,1\}^{\mathsf{sln}}$ and $t \in \{0,1\}^{\mathsf{sln}}$ 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 $|C_{m,t}| = 2^3$. $C_{m,t} = \{11110111, 11100111, 11110101, 11110011, 11100011, 11100001, 11110101, 11100001\}$.

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 $\sum_{\omega \in C_{m_i,t_i}} f(x,k) = \sum_{J \subseteq \{0,1\}^{\mathsf{kln}}} a_J^i \prod_{j \in J} k_j$ 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 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 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 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 1.

Table 1. Results on 10-round 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

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 ω and $\omega+\Delta$, where Δ has a low Hamming weight. Many statebits in the first rounds of computation $E_\kappa(\omega)$ and $E_\kappa(\omega + \Delta)$ 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 $\omega = (\omega_p : \omega_p \in C_{m,t})$. 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 $\sum_{\omega_p \in C_{m,t}} f(\omega_p, k) = \ell(k)$ for a linear function ℓ. We observe that the equation $\sum_{\omega_p \in C_{m,t}} f(\omega_p, k) = \ell(k)$ 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 TRIVIUM and GRAIN128, since there already exist cube attacks against a significant number of rounds [30, 25, 5].

4 Optimizing ElimLin

The 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 8. *Let $\mathcal{S}$ be the initial set for ElimLin. Let $\mathcal{S}_T, \mathcal{S}_L$ be the resulting sets of ElimLin. We call the linear span of $\mathcal{S}_T \cup \mathcal{S}_L$ ElimLin span and denote it by* $\boldsymbol{elspan}(\mathcal{S}) = \boldsymbol{linspan}(\mathcal{S}_T \cup \mathcal{S}_L)$.

Theorem 9 (ElimLin invariant [22]).
The span $\boldsymbol{elspan}(\mathcal{S})$ *is invariant with respect to the order of substitutions and Gauss elimination.*

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 5. 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.

5 Universal Proning: Recovering Linear Polynomials not found by ElimLin

We observe that most linear equations which ElimLin recovers are satisfied independent of the secret key, these are the linear equations in $\mathbf{elspan}\,(\mathcal{S}_{\omega,\star,\star})$ 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 $f \in \mathsf{R}$, such that $f \in \langle \mathcal{S}_{\omega,\star,\kappa} \rangle$ or $f \in \langle \mathcal{S}_{\star,\gamma,\kappa} \rangle$ for every key κ, hence, the name universal. Intuitively, we can see that a universal polynomial cannot help to recover the secret key but it helps to simplify the polynomial system. The concept of universal polynomials is closely related to concepts earlier studied in [20, slide 118-120]. Let us consider a polynomial $m \in \mathbf{F}_2[V]$ and a function which evaluates m under key κ.

Definition 10. *Let $\mathbf{F}_2[V]$ be the set of all polynomials in variables V over $\mathbf{F}_2$. Let us define the function $e_\omega : \mathbf{F}_2[V] \to \mathsf{Func}\left(\mathbf{F}_2^{kln}, \mathbf{F}_2\right)$, such that $e_\omega(m)$ is the function mapping κ in F_2^{kln} to the reduction of the polynomial m modulo $\langle \mathcal{S}_{\omega,\star,\kappa} \rangle$. Similarly, let us define the function $d_\gamma : \mathbf{F}_2[V] \to \mathsf{Func}\left(\mathbf{F}_2^{kln}, \mathbf{F}_2\right)$, such that $d_\gamma(m)$ is the function mapping κ in F_2^{kln} to the reduction of the polynomial m modulo $\langle \mathcal{S}_{\star,\gamma,\kappa} \rangle$.*

We recover universal polynomials from approximation of $\mathsf{ker}\,(e_\omega)$ and $\mathsf{ker}\,(d_\gamma)$.

6 Selection of Samples in KATAN32

We give the 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. The previous best algebraic attacks can be found in [6]. 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 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

7 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 5 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.

8 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 59-round cube of 2^{10} samples. The approach can be seen as an innovative 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 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 [20, 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 $\mathbf{linspan}\,(\mathbf{elspan}\,(\mathcal{S}_{\omega,\star,\star}) \cup \mathbf{elspan}\,(\mathcal{S}_{\star,\gamma,\star}))$ and the proper analysis of Universal Proning is a part of an ongoing research.

References

1. Al-Hinai, S.Z., Dawson, E., Henricksen, M., Simpson, L.R.: On the security of the LILI family of stream ciphers against algebraic attacks. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 11–28. Springer, Heidelberg (2007)
2. Albrecht, M.R., Cid, C., Faugère, J.-C., Perret, L.: On the Relation Between the Mutant Strategy and the Normal Selection Strategy in Gröbner Basis Algorithms. IACR Cryptology ePrint Archive 2011, 164 (2011)

3. Albrecht, M.R., Cid, C., Faugère, J.-C., Perret, L.: On the relation between the MXL family of algorithms and Gröbner basis algorithms. J. Symb. Comput. 47(8), 926–941 (2012)
4. Ars, G., Faugère, J.-C., Imai, H., Kawazoe, M., Sugita, M.: Comparison between XL and Gröbner basis algorithms. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 338–353. Springer, Heidelberg (2004)
5. Aumasson, J.-P., Dinur, I., Meier, W., Shamir, A.: Cube testers and key recovery attacks on reduced-round MD6 and Trivium. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 1–22. Springer, Heidelberg (2009)
6. Bard, G.V., Courtois, N.T., Nakahara Jr, J., Sepehrdad, P., Zhang, B.: Algebraic, aida/cube and side channel analysis of katan family of block ciphers. In: Gong, G., Gupta, K.C. (eds.) INDOCRYPT 2010. LNCS, vol. 6498, pp. 176–196. Springer, Heidelberg (2010)
7. Bardet, M., Faugère, J.-C., Salvy, B., Yang, B.-Y.: Asymptotic behaviour of the degree of regularity of semi-regular polynomial systems. In: Eighth International Symposium on Effective Methods in Algebraic Geometry, MEGA 2005, Porto Conte, Alghero, Sardinia, Italy, May 27-June 1 (2005)
8. Bardet, M., Faugère, J.-C., Salvy, B., Spaenlehauer, P.-J.: On the complexity of solving quadratic boolean systems. J. Complexity 29(1), 53–75 (2013)
9. De Cannière, C.: Trivium: A Stream Cipher Construction Inspired by Block Cipher Design Principles. In: Katsikas, S.K., López, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC 2006. LNCS, vol. 4176, pp. 171–186. Springer, Heidelberg (2006)
10. De Cannière, C., Dunkelman, O., Knežević, M.: KATAN and KTANTAN - a family of small and efficient hardware-oriented block ciphers. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 272–288. Springer, Heidelberg (2009)
11. Cheng, C.-M., Chou, T., Niederhagen, R., Yang, B.-Y.: Solving quadratic equations with XL on parallel architectures. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 356–373. Springer, Heidelberg (2012)
12. Choy, J., Yap, H., Khoo, K.: An analysis of the compact XSL attack on BES and embedded SMS4. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 103–118. Springer, Heidelberg (2009)
13. Cid, C., Leurent, G.: An analysis of the XSL algorithm. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 333–352. Springer, Heidelberg (2005)
14. Courtois, N.T.: Higher order correlation attacks, XL algorithm and cryptanalysis of toyocrypt. In: Lee, P.J., Lim, C.H. (eds.) ICISC 2002. LNCS, vol. 2587, pp. 182–199. Springer, Heidelberg (2003)
15. Courtois, N.T.: Algebraic attacks over GF(2^k), application to HFE challenge 2 and Sflash-v2. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 201–217. Springer, Heidelberg (2004)
16. Courtois, N.T., Bard, G.V.: Algebraic cryptanalysis of the data encryption standard. In: Galbraith, S.D. (ed.) Cryptography and Coding 2007. LNCS, vol. 4887, pp. 152–169. Springer, Heidelberg (2007)
17. Courtois, N.T., Bard, G.V., Wagner, D.: Algebraic and slide attacks on KeeLoq. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 97–115. Springer, Heidelberg (2008)
18. Courtois, N.T., Debraize, B.: Algebraic description and simultaneous linear approximations of addition in Snow 2.0. In: Chen, L., Ryan, M.D., Wang, G. (eds.) ICICS 2008. LNCS, vol. 5308, pp. 328–344. Springer, Heidelberg (2008)
19. Courtois, N.T., Pieprzyk, J.: Cryptanalysis of block ciphers with overdefined systems of equations. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 267–287. Springer, Heidelberg (2002)

20. Courtois, N.T.: A new frontier in symmetric cryptanalysis. Invited talk, Indocrypt (2008), http://www.nicolascourtois.com/papers/front_indocrypt08_2p.pdf
21. Courtois, N.T., Mourouzis, T., Song, G., Sepehrdad, P., Sušil, P.: Combined Algebraic and Truncated Differential Cryptanalysis on Reduced-Round Simon (April 2014) (Preprint)
22. Courtois, N.T., Sepehrdad, P., Sušil, P., Vaudenay, S.: ElimLin algorithm revisited. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 306–325. Springer, Heidelberg (2012)
23. Dinur, I., Shamir, A.: Cube attacks on tweakable black box polynomials. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 278–299. Springer, Heidelberg (2009)
24. Dinur, I., Shamir, A.: Side Channel Cube attacks on Block Ciphers. IACR Cryptology ePrint Archive 2009, 127 (2009)
25. Dinur, I., Shamir, A.: Breaking grain-128 with dynamic cube attacks. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 167–187. Springer, Heidelberg (2011)
26. Dinur, I., Shamir, A.: Applying cube attacks to stream ciphers in realistic scenarios. Cryptography and Communications 4(3-4), 217–232 (2012)
27. Erickson, J., Ding, J., Christensen, C.: Algebraic cryptanalysis of SMS4: Gröbner basis attack and SAT attack compared. In: Lee, D., Hong, S. (eds.) ICISC 2009. LNCS, vol. 5984, pp. 73–86. Springer, Heidelberg (2010)
28. Faugère, J.-C.: A new efficient algorithm for computing Grobner bases (F4). Journal of Pure and Applied Algebra 139(13), 61–88 (1999)
29. Faugère, J.-C., Perret, L.: Algebraic cryptanalysis of curry and flurry using correlated messages. In: Bao, F., Yung, M., Lin, D., Jing, J. (eds.) Inscrypt 2009. LNCS, vol. 6151, pp. 266–277. Springer, Heidelberg (2010)
30. Fouque, P.A., Vannet, T.: Improving Key Recovery to 784 and 799 rounds of Trivium using Optimized Cube Attacks. In: FSE 2013 (2013)
31. Hell, M., Johansson, T., Meier, W.: Grain; a stream cipher for constrained environments. Int. J. Wire. Mob. Comput. 2(1), 86–93 (2007)
32. Hodges, T., Petit, C., Schlather, J.: Degree of Regularity for Systems arising from Weil Descent. In: YAC 2012 - Yet Another Conference in Cryptography, p. 9 (2012)
33. Isobe, T., Sasaki, Y., Chen, J.: Related-key boomerang attacks on KATAN32/48/64. In: Boyd, C., Simpson, L. (eds.) ACISP. LNCS, vol. 7959, pp. 268–285. Springer, Heidelberg (2013)
34. Faugère, J.-C.: A New Efficient Algorithm for Computing Gröbner Bases Without Reduction to Zero (F5). In: ISSAC 2002: Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation, pp. 75–83 (2002)
35. Knellwolf, S., Meier, W., Naya-Plasencia, M.: Conditional differential cryptanalysis of Trivium and KATAN. In: Miri, A., Vaudenay, S. (eds.) SAC 2011. LNCS, vol. 7118, pp. 200–212. Springer, Heidelberg (2012)
36. Knudsen, L.R.: Truncated and higher order differentials. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 196–211. Springer, Heidelberg (1995)
37. Lim, C.-W., Khoo, K.: An analysis of XSL applied to BES. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 242–253. Springer, Heidelberg (2007)
38. Lipton, R.J., Viglas, A.: On the complexity of SAT. In: 40th FOCS, October 17-19, pp. 459–464. IEEE Computer Society Press, New York (1999)
39. Mohamed, M.S.E., Mohamed, W.S.A.E., Ding, J., Buchmann, J.: MXL2: Solving Polynomial Equations over GF(2) Using an Improved Mutant Strategy. In: Buchmann, J., Ding, J. (eds.) PQCrypto 2008. LNCS, vol. 5299, pp. 203–215. Springer, Heidelberg (2008)

40. Mohamed, M.S.E., Cabarcas, D., Ding, J., Buchmann, J., Bulygin, S.: MXL3: An efficient algorithm for computing Gröbner bases of zero-dimensional ideals. In: Lee, D., Hong, S. (eds.) ICISC 2009. LNCS, vol. 5984, pp. 87–100. Springer, Heidelberg (2010)
41. Song, L., Hu, L.: Improved algebraic and differential fault attacks on the katan block cipher. In: Deng, R.H., Feng, T. (eds.) ISPEC 2013. LNCS, vol. 7863, pp. 372–386. Springer, Heidelberg (2013)
42. Soos, M.: Cryptominisat 2.5.0. In: SAT Race competitive event booklet (July 2010)
43. Stegers, T.: Faugère's F5 Algorithm Revisited. Cryptology ePrint Archive, Report 2006/404 (2006), http://eprint.iacr.org/
44. Wu, W., Zhang, L.: LBlock: A lightweight block cipher. In: Lopez, J., Tsudik, G. (eds.) ACNS 2011. LNCS, vol. 6715, pp. 327–344. Springer, Heidelberg (2011)
45. Yang, B.-Y., Chen, J.-M., Courtois, N.T.: On asymptotic security estimates in XL and Gröbner bases-related algebraic cryptanalysis. In: López, J., Qing, S., Okamoto, E. (eds.) ICICS 2004. LNCS, vol. 3269, pp. 401–413. Springer, Heidelberg (2004)