Variable Elimination - a Tool for Algebraic Cryptanalysis

Bjørn Greve[1], Øyvind Ytrehus[2], and Håvard Raddum[2]

[1] *Norwegian Defence Research Establishment, Kjeller, Norway*
[2] *Simula UiB, Bergen, Norway*

Abstract

Techniques for eliminating variables from a system of nonlinear equations are used to find solutions of the system. We discuss how these methods can be used to attack certain types of symmetric block ciphers, by solving sets of equations arising from known plain text attacks. The systems of equations corresponding to these block ciphers have the characteristics that the solution is determined by a small subset of the variables (*i.e.*, the secret key), and also that it is known that there always exists at least one solution (again corresponding to the key which is actually used in the encryption). It turns out that some toy ciphers can be solved simpler than anticipated by this method, and that the method can take advantage of overdetermined systems.

1 Introduction

The goal of algebraic cryptanalysis is to exploit the algebraic structure of a cipher in order to retrieve the secret key. This is achieved by expressing the encryption transformation as a large system of nonlinear equations over a finite field, and then solving this system of equations. Algebraic cryptanalysis received renewed attention recently, and attacks such as the XL and XSL algorithms were introduced [13, 14, 16], although these attacks were shown to be infeasible in practice [21, 20, 22]. Approximately at the same time, direct constructions of Gröbner bases were considered [4]. We also note that the algebraic approach may be combined with other cryptanalytic methods [27].

As a consequence of these results, researchers instead started to consider reduced block ciphers with a behaviour resembling that of real block ciphers, in order to perform experiments in practice and to to gain insight into the behaviour and the effectiveness of algebraic attacks. In [8] the authors performed experiments for solving systems of equations arising from small-scale versions of AES by using Gröbner basis methods via the $F4$ algorithm [17]. The results showed that even for small-scale versions of AES, the computations became unfeasible at an early stage. The results on small scale variants of the AES cipher were improved in [5], where the authors conclude that Gröbner basis methods are unlikely to solve a full system of equations for AES.

The ideas presented in this paper are motivated by the recent papers [24, 25], where the authors develop efficient algorithms for eliminating variables systems of Boolean equations. Our aim is to eliminate the auxiliary variables introduced when describing reduced block ciphers as systems of equations over $\mathbb{F}_2$, while still limiting the degree of the polynomial equations that arise during the elimination process, in contrast to other related methods such as Gröbner bases. The downside of this approach is that during the elimination algorithm, higher degree polynomials have to be discarded, with the implication that false solutions (keys) will be introduced in the system. In order to deal with this problem, the concept of information loss is introduced. Using this concept, we generalize the elimination algorithm from [24] into a *dynamical* elimination algorithm. We perform experiments on several types of small scale block ciphers, and in particular demonstrate how the suggested algorithm can eliminate many of the auxiliary variables from block ciphers while

keeping the degrees at minimum without losing any information about the user selected key. In particular it is also demonstrated how the algorithm in some cases actually succeeds in eliminating all the auxiliary variables and find polynomials of low degree in only the key variables.

The paper is structured as follows: Section 2 introduces how to construct quadratic systems of Boolean equations representing block ciphers, as well as basic notation used in the paper. In Section 3 we describe the dynamical elimination algorithm. Section 4 connects the algorithm to information theory and introduces the information measure as well as the concept of information loss. Section 5 presents experimental results, while Section 6 contains a conclusion and sketches some issues for further research.

2 Background, notation and preliminaries

For reference, we start by listing up some of the notation that will be used throughout the paper.

Term	Meaning
k	Length of key, plain text and cipher text
n	Number of variables of the set of equations
r	Number of rounds in the cipher
m	Number of polynomials or equations
$\mathcal{P}$	Message space
$\mathcal{C}$	Cipher text space
$\mathcal{K}$	Key space
$x_0, \ldots, x_{n-1}$	The variables of the set of equations
$\mathbb{B}[x_j, \ldots, x_{n-1}]$	$\mathbb{F}_2[x_j, \ldots, x_{n-1}]/(x_i^2 + x_i \vert i = j, \ldots, n-1)$
t	Monomial term
f^i	Polynomial of degree i
F	Set of polynomials, or equations on the form $f = 0$
$F^i, F^{\leq i}$	Set of polynomials, or equations, all of degree $= i$, resp. $\leq i$
$L^i, L^{\leq i}$	Set of all monomials of degree $= i$, resp. $\leq i$
$\langle F \rangle$	The linear span of the polynomials in F
$F_{x_i}, F_{\overline{x_i}}$	Set of polynomials in which all (resp. no) polynomials depend on x_i
$Z(F)$	Set of zero points for all polynomials in F
$I(F)$	The ideal generated by F
π_i	Projection omitting coordinates $0, \ldots, i$, that is, $\pi_i : \mathbb{F}_2^n \to \mathbb{F}_2^{n-i-1}$
$M(F)$ or $M_d(F)$	Macaulay matrix corresponding to F

A block cipher is described by an invertible *encryption* function $E : \mathcal{K} \times \mathcal{P} \to \mathcal{C}$ that takes the plaintext $P \in \mathcal{P}$ and the key $K \in \mathcal{K}$ as input, and outputs $E(K, P) = C \in \mathcal{C}$, the resulting ciphertext. The *decryption* function $E^{-1}(K, C) = P \in \mathcal{P}$ yields the corresponding plaintext. Here the sets $\mathcal{P}$, $\mathcal{C}$ and $\mathcal{K}$ are the message space, cipher text space, and the key space respectively. In this paper, for convenience, we will focus on the (common) case where $\mathcal{C} = \mathcal{P} = \mathcal{K} = \mathbb{F}_2^k$.

2.1 Block ciphers and systems of polynomial equations

We consider modern block ciphers which are designed as a sequence of r *rounds*, exemplified in Figures 2 and 8. Each round consists of a *confusion* operation executed by an *S-box*, and a *diffusion* operation, which is typically a linear transformation acting on all current variables. S-boxes in practical ciphers are selected so that there are no linear relations between the input and the output symbols of the S-box. From an algorithmic point of view, given a key $K \in \mathcal{K}$, the encryption algorithm produces a unique ciphertext for a given plaintext input, and the decryption algorithm produces a unique plaintext for a given ciphertext input.

Mathematically, these block ciphers can be viewed in terms of a family of equations that relate the key $K \in \mathcal{K}$ to a plaintext and a corresponding ciphertext. That is, in a known plaintext attack, we may substitute the plaintext and the corresponding ciphertext produced for a specific key $K \in \mathcal{K}$ into this general family of equations, to derive a set F of nonlinear equations that restrict the set of possible keys used for the encryption. Thus solving this set of equations in practice breaks the cipher.

Constructing systems of equations for block ciphers over $\mathbb{F}_2$: The essential set of variables corresponds to the encryption key, denoted by $K = (x_0, \ldots, x_{k-1})$. In accordance with Figures 2 and 8, let X_0 be the bitwise XOR of K and P, then X_0 is the input to the first round of the cipher. For the ℓ'th round, for $\ell = 1, \ldots, r-1$, we introduce new *output variables*, $X_\ell = (x_{k\ell}, \ldots, x_{k(\ell+1)-1})$ representing the "internal state" of the cipher after execution of the ℓth round. Thus X_ℓ is the output of round ℓ and the input to round $\ell + 1$. The auxiliary variables produced at the output variables of the r'th round can be immediately eliminated since they can be described as linear combinations of the key and ciphertext. The significance of this structured way of designing the set of equations is that if the S-box can be represented by relations of degree 2, then the entire set of equations have degree 2. This restriction of degrees comes at the expense of expanding the set of variables by the introduction of the auxiliary variables $x_k, \ldots, x_{kr-1}$. The total number of variables is $n = kr$. (Alternatively, the set of variables could be confined to the original key variables, but this would lead to extremely complex equations in $\{x_0, \ldots, x_{k-1}\}$.)

Thus we can construct r disjoint sets $F_1, \ldots, F_r$ of quadratic equations over $\mathbb{F}_2$ as follows. Each polynomial $f \in F_\ell$ is a sum of some monomials of degree ≤ 2 monomials, relating variables in the set $V_\ell \cup V_{\ell+1}$.

For schematic demonstrations on how ciphers are constructed we refer to Section 5. To get a feeling on how the equations look like, the following equations are the first equations describing the toy cipher considered in Section 5.3.

$$\begin{aligned}
x_1 + x_2 + 1 + x_{16} + x_{19} + x_0x_1 + x_0x_3 + x_0x_{16} + x_0x_{17} + x_0x_{18} &= 0 \\
x_1 + x_2 + 1 + x_{16} + x_{19} + x_0x_1 + x_0x_2 + x_0 + x_0x_{16} + x_0x_{19} &= 0 \\
x_2 + x_{16} + x_{17} + x_{19} + x_0x_1 + x_0x_3 + x_0x_{16} + x_1x_2 + x_1 + x_1x_{16} &= 0 \\
x_2 + x_3 + x_{16} + x_{18} + x_{19} + x_0x_1 + x_0x_2 + x_0 + x_0x_{16} + x_1x_2 + x_1 + x_1x_{17} &= 0 \\
x_2 + x_3 + x_{16} + x_{18} + x_{19} + x_0x_1 + x_0x_2 + x_0 + x_0x_{16} + x_1x_3 + x_1x_{18} &= 0 \\
\vdots & \vdots
\end{aligned}$$

The resulting set of equations $F = \cup_{j=0}^{r} F_j$ describes the complete block cipher encryption. The set of equations forms the basis for an ideal of the multivariate polynomial ring $\mathbb{B}[x_0, \ldots, x_{n-1}]$, whose zero set is denoted by

$$Z(F) = \{\mathbf{a} \in \mathbb{F}_2^n | f(\mathbf{a}) = 0, \forall f \in F\}.$$

2.2 Algorithms for eliminating variables and solving systems of Boolean equations

The work in this paper is inspired by the algorithms and a framework for eliminating variables from a system of Boolean equations developed in [24, 25]. In the following we will give a brief introduction to the tools developed in these papers, starting with [24]. For more details, we refer the reader to the papers.

A standard technique for computing elimination ideals is to use Gröbner bases, which eliminate one monomial at the time. In fact, to compute elimination ideals in practice via Gröbner bases one has to compute the full Gröbner basis before performing elimination. This is because the orders with the elimination property are usually not the most optimal choice during computations [4, 26]. In addition Gröbner bases are computationally heavy because the degrees of the polynomials grow rapidly over the iterations.

In an attempt to avoid these problems, the authors of [24] propose an alternative algorithm for eliminating variables from systems of Boolean equations, which also restricts the degree of the polynomials. Roughly speaking, the idea is to produce only new polynomials of restricted degree, with the benefit that the elimination process has much lower complexity. Adapting the notation from [24], an ideal where the degree is restricted to some d is denoted by J^d, where J^∞ means that we allow all degrees.

The disadvantage of not using all polynomials is that one has to discard polynomials of degree $> d$, which gives an ideal J^d that is only contained in the elimination ideal $J^\infty = I(F) \cap \mathbb{B}[x_j, \ldots, x_{n-1}]$. It follows that $Z(J^d)$ of the eliminated system contains all the projected solutions of the original set of equations, but it may also contain "false" solutions which will not fit the ideal $I(F)$ when lifted back to the original vector space, regardless of which values we assign to the eliminated variables. Thus there is a trade-off between the maximum degree d allowed, and the proximity between the "practical" ideal J^d and the true elimination ideal J^∞.

Throughout the paper we focus on only eliminating the variable x_0 from F. In practice, all procedures are iterated to eliminate any number of variables we choose. As part of the variable elimination process it will be necessary to split a set of polynomials according to dependency on x_0 and possibly according to degree. The procedures for this can be implemented in terms of row reduction on the *Macaulay matrix* of the given set. This is a matrix in which the columns are indexed by the monomials according to the total order we use, and the rows are indexed by the polynomials in our set. Each matrix entry is the coefficient of the corresponding monomial in the corresponding polynomials. We shall need two types of total orders on the monomials in a Boolean ring.

1. An x_0-elimination order: If t_1 and t_2 are monomials where t_1 contains x_0 while t_2 does not, then $t_1 > t_2$. An example is the lexicographic order.

2. A degree order: If t_1 has degree larger than t_2 then $t_1 > t_2$.

In this paper we adopt the following non-standard notation from [24]: For a polynomial f^i, the superscript i indicates that f has degree i, and does not mean f raised to the power i. The reason for this choice of notation is that since the idea is to bound the degree of polynomials we work with, it is important to keep in mind which degree the various polynomials and monomials have. Similarly, F^i for a set of polynomials indicates that all polynomials in F have degree i. This means that we can split the initial system F into d sets according to degree:

$$F = F^d \cup F^{d-1} \cup \ldots \cup F^2 \cup F^1. \tag{1}$$

For any cipher worth studying, we may assume that in the initial system $F^1 = \emptyset$.

It will be convenient to use a formal procedure $SplitVariable(F, x_0)$. This procedure is essentially Gaussian elimination on the Macaulay matrix $M(F)$ on all the columns indexed by monomials containing x_0, using the x_0-elimination order. When $M(F)$ is in row-reduced echelon form, the set F is naturally split into polynomials that contain terms depending on x_0, and polynomials with no terms depending on x_0. The outputs of $SplitVariable(F, x_0)$ are sets of polynomials F_{x_0} consisting of the polynomials which contain x_0-terms (the upper rows of the resulting matrix), and $F_{\overline{x_0}}$ consisting of polynomials not containing x_0-terms (the remaining lower rows of the resulting matrix).

The tools used to construct the elimination framework from [24] are resultants, coefficient constraints, and syzygies, accompanied with the concept of normalization. Consider the projection which omits the first coordinate:

$$\pi_0 : \mathbb{F}_2^n \to \mathbb{F}_2^{n-1}$$

$$(a_0, a_1, \ldots, a_{n-1}) \mapsto (a_1, \ldots, a_{n-1}),$$

and denote by $\mathbb{B}[x_1, \ldots, x_{n-1}]$ the ring of Boolean polynomials where x_0 has been omitted. We may in a similar fashion consider a sequence of i projections

$$\mathbb{F}_2^n \to \mathbb{F}_2^{n-1} \to \cdots \to \mathbb{F}_2^{n-i},$$

that we denote as $\pi_{i-1} : \mathbb{F}_2^n \to \mathbb{F}_2^{n-i}$. When eliminating x_0 from an ideal $I \subset \mathbb{B}[x_0, \ldots, x_{n-1}]$, we compute the elimination ideal $I \cap \mathbb{B}[x_1, \ldots, x_{n-1}]$. The advantage of the algorithm from [24] is that the number of variables decreases in each iteration, and in particular it is shown to be more efficient than the Gröbner base approach.

The *resultant* eliminates x_0 from a pair of polynomials. Formally, let $f_1 = a_1 x_0 + b_1$ and $f_2 = a_2 x_0 + b_2$ be two polynomials in $\mathbb{B}[x_0, \ldots, x_{n-1}]$ where the variable x_0 has been factored out so the polynomials a_i and b_i are in $\mathbb{B}[x_1, \ldots, x_{n-1}]$. The resultant is then given by $\mathrm{Res}(f_1, f_2, x_0) = a_1 b_2 + a_2 b_1$, and hence lies in $\mathbb{B}[x_1, \ldots, x_{n-1}]$. This can of course be computed with respect to any variable, as well as in an iterative manner. When restricting the degree to $\leq d$, we can take the resultant of all polynomial pairs from the sets F^i, F^j, where $i + j \leq d + 1$ as

$$\mathrm{Res}(F^i, F^j) = \{\mathrm{Res}(f, g, x_0) | f \in F^i, g \in F^j, i + j \leq d + 1\}.$$

For a set F of polynomials of differing degree, we can split F into $F^d, \ldots, F^2$ and denote the set of all resultants by

$$Res(F) = \cup_{i+j \leq d+1} Res(F^i, F^j) \subseteq I(F) \cap \mathbb{B}[x_1, \ldots, x_{n-1}].$$

Note that the last inclusion is in general strict, where in order to close the gap the concept of *Coefficient constraints* was introduced in [24].

For coefficient constraints, again write f_i as $f_i = a_i x_0 + b_i$ where neither a_i nor b_i depends on x_0. The set of coefficient constraints $\mathrm{Co}(F)$ is then defined as

$$\mathrm{Co}_1(F) = \{b_1(a_1 + 1), b_2(a_2 + 1), \ldots, b_m(a_m + 1)\}.$$

An important fact from [24] is that the zero set of $Co(F)$ is in the projection of the zero set of $I(F)$ onto $\mathbb{F}_2^n$, and together with the resultants describes exactly the full elimination ideal $I(F) \cap \mathbb{B}[x_1, \ldots, x_{n-1}]$. This can be iterated by computing resultants and coefficient constraints iteratively with respect to any variable. Algorithm 1 restricts the degree to d, but allowing all degrees it will compute the full elimination ideal.

As demonstrated in [24], these objects link naturally to the notion of *syzygies* which are connected to Gröbner bases in the literature in relation to optimizations of Buchberger's algorithm. The most known approaches to Gröbner bases (as for example [17, 18]), reduce to computing the module of syzygies over the polynomial ring $K[x_0, \ldots, x_{n-1}]$ for some field K. The approach in [24] is not to compute Gröbner bases, but rather find ways to eliminate variables from the system of polynomials, without using Gröbner bases ([6, 7]). The idea from [24] is to find more efficient ways of computing the vector space

$$\langle F^d \cup L^{\leq 1} F^{d-1} \cup L^{\leq 2} F^{d-2} \cup \cdots \cup L^{\leq d-2} F^2 \rangle \cap \mathbb{B}[x_1, \ldots, x_{n-1}],$$

where $L^{\leq i}$ consists of all monomials of degree $\leq i$, and $L^{\leq j} F^i$ consists of all products $l^{\leq j} f^i$ where $l^{\leq j} \in L^{\leq j}$ and $f^i \in F^i$, $(i = 2, \ldots, d-1)$. Since we are bounding the maximal degree to be d, we can be certain to form any such product provided that $i + j \leq d$. In this regard the construction from [24] differs from Gröbner bases in the following sense.

1. In [24] the syzygies are computed directly in the Boolean ring $\mathbb{B}[x_1, \ldots, x_{n-1}]$, which means that the field equations are encoded into the computations in contrary to Gröbner bases which are computed over the polynomial ring $\mathbb{F}_2[x_1, \ldots, x_{n-1}]$ and thus needing the field equations to be added to the system.

Algorithm 1 ResAndCo($F = F^d \cup F^{d-1} \cup \ldots \cup F^2 \cup F^1, x_0$)

In: Set $F = F^d \cup F^{d-1} \cup \ldots \cup F^2 \cup F^1$ of polynomials in $\mathbb{B}[x_0, \ldots, x_{n-1}]$
Out: Set R of polynomials where $\pi_0(Z(F)) \subseteq Z(R)$ and no polynomial in R depends on x_0

for $1 \leq i \leq d$ **do**
  for f_k^i in F^i **do**
    $f_k^i = a_k x_0 + b_k$
  end for
end for
$R = \emptyset$
for $1 \leq i \leq d$ **do**
  for $1 \leq j \leq d + 1 - i$ **do**
    for $(f_k^i, f_l^j) \in F^i \times F^j$ **do**
      $R \leftarrow R \cup \{a_k b_l + a_l b_k\}$ ▷ Resultants
    end for
  end for
end for
for $1 \leq i \leq (d+1)/2$ **do**
  for $f_k^i \in F^i$ **do**
    $R \leftarrow R \cup \{b_k(a_k + 1)\}$ ▷ Coefficient constraints
  end for
end for
Return R

2. In the approach of [24] its only needed to compute syzygies on the a_j^{i-1} terms, and these have degree *one less* than the f^i's.

3. The approach is "straight to the point" in the sense that we eliminate variables throughout the algorithm, in contrast to first computing the full Gröbner basis and then doing the elimination. We also avoid precise term orderings apart from the elimination order $x_0 > x_1 > \ldots$, and we eliminate one variable in each iteration, instead of only one monomial.

In [24], the non-trivial syzygies are also treated, but for computational purposes we avoid those here. For more details we refer to [24]. The next fundamental tool used for elimination from [24], is *normalization*.

2.3 Normalization

The purpose of normalization is to eliminate many monomials containing x_0 from a given polynomial, using a set of lower-degree polynomials depending on x_0 as a basis. Suppose we have an x_0-elimination ordering. Let $f^i \in F^i_{x_0}$ be given, and let $G^{\leq i}_{x_0} = \{g_1, \ldots, g_s\}$ be a set of polynomials where each g_j depend on x_0. In general we say that f^i is normalized with respect to G_{x_0} if no term in f^i is divisible by any leading term of the polynomials in G_{x_0}, and we write $f^{i,norm}$ when we need to stress that f^i is normalized (with respect to some basis).

Without restriction on the degree, it is easy to create $f^{i,norm}$ from f^i and G_{x_0}. Simply run through the polynomials in G_{x_0} and check if some monomial m_f in f^i is divisible by an initial term of some $g_j \in G_{x_0}$. If $m_f = q * in(g_j)$, eliminate m_f by adding qg_j to f^i. Doing this successively for all polynomials in G_{x_0} will produce $f^{i,norm}$.

Since we need to restrict the degree of the polynomials we work with to be at most d, extra care has to be taken. The leading term $in(g_j)$ of some g_j may not be of the highest degree among the monomials in g_j, for instance if $g_j = x_0x_3 + x_1x_3x_4$. When this happens we can only eliminate

the m_f in f^i where $deg(q) + deg(g_j) \leq d$, since we forbid introducing terms of degree larger than d. We therefore give the following definition of what normalization means in this paper.

Definition 1. *Let $f^i \in F^i_{x_0}$ and $d \geq i$ be given, and let G_{x_0} be a set of polynomials all depending on x_0 with an x_0-elimination ordering. We say that f^i is* normalized *with respect to G_{x_0} if no term m_f in f^i can be written as $m_f = q * in(g)$ for any $g \in G_{x_0}$ where $deg(q) + deg(g) \leq d$. When writing $F^{i,norm}$ for a set of polynomials we mean that every polynomial in F^i is normalized with respect to G_{x_0}, and that all the polynomials have distinct initial terms.*

In our algorithm we combine normalization and Gaussian reduction to get polynomials with distinct initial terms. We start with $F^2_{x_0}$ and perform Gaussian reduction. Denote the new set of polynomials $F^{2,norm}_{x_0}$. The general procedure for normalizing all of F is as follows.

NORMALIZE($F^i_{x_0}$)

1. The polynomials of $F^i_{x_0}$ are put into the rows of a Macaulay matrix. Perform $SplitVariable(F^i_{x_0}, x_0)$ to get new set $F^i_{x_0}$ where the x_0-part is in row-reduced echelon form. If there are polynomials without x_0-terms, put these into $F^i_{\overline{x_0}}$.

2. For each f^i in $F^i_{x_0}$ normalize it with respect to $\cup_{j=2}^{i-1} F^{j,norm}_{x_0}$.

3. Again perform $SplitVariable(F^i_{x_0}, x_0)$ to get the x_0-part in row-reduced echelon form. The polynomials containing x_0 form the set $F^{i,norm}_{x_0}$. If any polynomials do not contain x_0, then add these polynomials to the set $F^i_{\overline{x_0}}$.

The output of the normalization procedure will be the two sets $F^{i,norm}_{x_0}$ and $F^i_{\overline{x_0}}$.

2.4 An algorithm for solving systems of Boolean equations [25]

In [25] it is studied how to solve Boolean equation systems by eliminating variables, building on the work in [24]. It is shown that for systems not containing any cross-terms between variables being eliminated, the solution space of new equation systems produced stays intact if we allow increasing the degree by one for every variable eliminated. This insight allows to set bounds on the number of monomials occurring in the systems as they evolve, and gives an accurate estimate on how many variables that need to be eliminated before the system can be solved by re-linearization. We note that it is also shown that if we allow the degree to run freely, the maximal degree of the polynomials when running the algorithm is upper bounded by $n/2$.

The estimate in [25] gives a surprisingly low number of needed elimination before a system can be solved. The complexity depends on how much the initial system is overdetermined, and decreases with the degree of overdeterminedness. This is also verified experimentally, although only for relatively small values of n.

3 Dynamical elimination algorithms

In this section we develop a "dynamical" elimination algorithm to be applied on systems of Boolean equations describing block ciphers, based on the algorithms and ideas from [24, 25]. Dynamical elimination in this setting means that we restrict the degree of the polynomials when eliminating variables, and only allow the degree to grow after a number of eliminations have already occurred. We select an initial degree bound d, and discard polynomials of degree $> d$ that arise during each elimination. Thus, at some point we obtain an ideal J^d that is only contained in the true elimination ideal $J^\infty = I(F) \cap \mathbb{B}[x_j, \ldots, x_{n-1}]$: That is, $Z(J^d)$ of the eliminated system contains all the projected solutions of the original set of equations, in addition to "false" solutions which will not fit the ideal $I(F)$ when lifted back to $\mathbb{F}_2^n$, regardless of which values we assign to the eliminated variables.

Hence, we need a device that can help us to detect when this expansion in solution space happens. In Section 4 we introduce the information theoretic concept of information measure and information loss, which is a tool used for detecting false solutions. The idea of the hybrid algorithm is to eliminate variables respecting the degree limit, until some given point, and then increase the degree in order to keep more information about the solutions in the system. Several strategies for when to increase the degree bound can be envisioned, and in Section 5 we try out several of them and see how they differ.

3.1 The main elimination algorithm

Restricting the degrees means that the objective of the algorithm is to find as many polynomials in the ideal $I(F)$ generated by $F^d, F^{d-1}, \ldots, F^2$ as possible *computing only with polynomials of degrees* $\leq d$. The overall goal and benefit of this algorithm is the limiting of both storage and computational complexity, and following the ideas from [24, 25] the goal is to find more efficient ways of computing the vector space

$$\langle F^d \cup L^1 F^{d-1} \cup L^2 F^{d-2} \cup \cdots \cup L^{d-2} F^2 \rangle \cap \mathbb{B}[x_k, \ldots, x_{n-1}].$$

The ideas and algorithms from [24] iteratively eliminate one variable at the time. However, when considering higher degree polynomials, one has to take the *non-trivial* syzygies into account. For simplicity and avoiding these difficulties, the algorithm proposed in this paper is a subversion of the setup from [24], not containing the extra component of computing non-trivial syzygies. This means that the algorithm referred to as *eliminate()*, is designed for eliminating variables from a system of Boolean equations by only considering the syzygies corresponding to resultants and coefficient constraints, along with normalization.

The main reason for proposing a subversion of the setup in [24] is that when performing experiments, we experienced that the simplified algorithm performed well and rarely needed to compute resultants and coefficient constraints. Based on this, we believe that the non-trivial syzygies also occur rarely, and the reason lies in the normalization, which seems to remove most monomials containing the variable to be eliminated. We therefore believe that our implementation is quite close to $\langle \cup_{j=1}^{d} L^{d-j} F^j \rangle \cap \mathbb{B}[x_k, \ldots, x_{n-1}]$, despite the fact that the algorithm theoretically only produces a subset of $\langle \cup_{j=1}^{d} L^{d-j} F^j \rangle \cap B[k+1, n]$. More precisely, *eliminate()* is explained as follows:

1. We start by using *SplitVariable*(F, x_0) to divide each of the sets $F^{d-1}, \ldots, F^2, F^1$ into subsets $F_{x_0}^{d-1}, \ldots, F_{x_0}^2, F_{x_0}^1$ containing x_0 and $F_{\overline{x_0}}^{d-1}, \ldots, F_{\overline{x_0}}^2, F_{\overline{x_0}}^1$ not containing x_0. This is performed by putting the polynomials of F into the rows of the Macaulay matrix $M(F)$ and doing Gauss elimination on terms depending on x_0.

2. We increase the input by adding for each $i = 1, 2, \ldots, d-1$, the sets $x_0 F_{\overline{x_0}}^{i-1}$, $(x_0+1)F_{x_0}^{i-1}$ to F^i. Note that we multiply each set F^{i-1} with only one variable (x_0, to be eliminated), and avoid multiplying with all of $L^{\leq i-1}$. The reason for this multiplication is to generate more polynomials of degree i, see [24].

3. For the new increased sets $F^d, F^{d-1}, \ldots, F^3, F^2, F^1$, we split each of them into $F_{x_0}^i$ containing x_0 and $F_{\overline{x_0}}^i$ not containing x_0, again by using *SplitVariable*(F^i, x_0).

4. For each $i = 2, \ldots, d$ we normalize each f^i in $F_{x_0}^i$, NORMALIZE$(F_{x_0}^i)$ with respect to $\cup_{j=2}^{i-1} F_{x_0}^{j,norm}$. Normalizing removes many monomials containing x_0 from F. We note that this will impose a chain reaction, removing monomials in an iterative manner from the sets $F_{x_0}^i$ as i increases. Again perform *SplitVariable*$(F_{x_0}^i, x_0)$ to get the x_0-part in row-reduced echelon form. The polynomials containing x_0 form the set $F_{x_0}^{i,norm}$. If any polynomials do not contain x_0, then add these polynomialst to the set $F_{\overline{x_0}}^i$.

5. From the current sets $F_{x_0}^{i,norm}$ we compute resultants and coefficient constraints, potentially producing more polynomials without x_0. We create sets R of polynomials in $\mathbb{B}[x_1, \ldots, x_{n-1}]$ by using Algorithm 1 satisfying the degree constraint $\leq d$. This produces as many polynomials without x_0 as possible for each degree $\leq d$. All of these are added to the sets $F_{\overline{x_0}}^i$, and we remove any linearly dependent polynomials in $F_{\overline{x_0}}^i$ before returning.

The new sets $F_{\overline{x_0}}^d, F_{\overline{x_0}}^{d-1}, \ldots, F_{\overline{x_0}}^2, F_{\overline{x_0}}^1$ are returned from *eliminate()*, as described in Algorithm 2. Note that when $F^1 = \emptyset$ (as is normally the case), the lowest set of equations considered would be F^2.

Algorithm 2 $eliminate(F^d, F^{d-1}, \ldots, F^2, F^1, x_0)$

In: Sets $F^d, F^{d-1}, \ldots, F^2, F^1$, where each $F^i = (f_{1_i}^i, \ldots, f_{m_i}^i)$ is a set of polynomials of degree i in $\mathbb{B}[x_0, \ldots, x_{n-1}]$ for $1 \leq i \leq d$ and x_0 the variable to be eliminated
Out: Sets $F_{\overline{x_0}}^d, F_{\overline{x_0}}^{d-1}, \ldots, F_{\overline{x_0}}^2, F_{\overline{x_0}}^1$ of polynomials where $x_0 \notin \cup_{i=1}^d F_{\overline{x_0}}^i$

for $1 \leq i < d$ **do**
$F_{x_0}^{i-1}, F_{\overline{x_0}}^{i-1} \leftarrow SplitVariable(F^{i-1}, x_0)$ ▷ $f_j^{i-1} \in F_{x_0}^{i-1}$ will have unique leading monomials containing x_0
$F^i \leftarrow (x_0 + 1)F_{x_0}^{i-1} \cup x_0 F_{\overline{x_0}}^{i-1} \cup F^i$
$F_{x_0}^i, F_{\overline{x_0}}^i \leftarrow SplitVariable(F^i, x_0)$
$F_{\overline{x_0}}^i, F_{x_0}^{i,norm} \leftarrow Normalize(F_{x_0}^i)$ with respect to $\cup_{j=1}^{i-1} F_{x_0}^j$
$F_{\overline{x_0}}^i \leftarrow ResAndCo(\cup_{j=1}^{i-1} F_{x_0}^{j,norm}, x_0) \cup F_{\overline{x_0}}^i$
end for
Return $F_{\overline{x_0}}^d, F_{\overline{x_0}}^{d-1}, \ldots, F_{\overline{x_0}}^2, F_{\overline{x_0}}^1$

3.2 Extensions of the elimination algorithm

We briefly explain how the general elimination algorithm can be improved by adding the following procedure:

It happens that the vector space $\langle \cup_{i=2}^d L^{\leq d-i} F^i \rangle$ contains more linearly independent polynomials in each respective degree $\leq d-1$, beyond the ones found initially in each of the sets $F^{d-1}, \ldots, F^2$. Hence we may search for new polynomials of degree $\leq d-1$, and exploit those also for elimination. This can be done in an iterative way by ordering the monomials according to degree. Formally, the procedure may be performed as follows.

Let $F^{i,(1)} = F^i$ and assume that the sets F^j for $j < i$ are already as large as possible. We then compute

$$\langle F^{i,(2)} \rangle = \langle \cup L^{d-i} F^{i,(1)} \cup (\cup_{j=1}^{i-1} L^{d-j} F^j) \rangle \cap \mathbb{B}[x_1, \ldots, x_{n-1}].$$

If now $\langle F^{i,(2)} \rangle$ is strictly larger than $\langle F^{i,(1)} \rangle$, we continue to compute the intersections

$$\langle F^{i,(k+1)} \rangle = \langle \cup L^{d-i} F^{i,(k)} \cup (\cup_{j=1}^{i-1} L^{d-j} F^j) \rangle \cap \mathbb{B}[x_1, \ldots, x_{n-1}],$$

until it stabilizes, and we get $\langle F^{i,(k+1)} \rangle = \langle F^{i,(k)} \rangle$ for some $k \geq 1$.

Following this procedure we can create more polynomials of allowed degree than just by multiplying monomials on the initial system once. This procedure allows us to work with polynomials that would be discarded because of the degree restriction if they were generated in the first multiplication process: After a Gaussian elimination with respect to degree produces new lower-degree polynomials, we can multiply those with a new set of monomials and still remain within the degree d. These new polynomials can not be reached after one run of multiplication of monomials on the

initial system, without violating the degree bound. This procedure can easily be implemented in *eliminate*(), by adding a while-loop searching for lower degree polynomials when normalizing. The reason for implementing the search for additional low degree polynomials in this way is the fact that it may increase the number of low degree polynomials in the basis for normalization, and thus improve the normalization procedure. In Section 5 we perform experiments with this improved version of the algorithm.

4 Information measure and information loss

The aim of this section is to connect the problem of solving systems of equations representing block ciphers to information theoretical aspects. In the setting of information theory, we can associate the *information* that F^2 contains in terms of the number of solutions of the system.

Let X be a discrete random variable that takes values $v_1, \ldots, v_M$ with probabilities $p_i = P(X = v_i), i = 1, \ldots, M$. The (binary) *entropy* of X is defined as

$$H(X) = -\sum_{i=1}^{M} p_i \log_2 p_i. \tag{2}$$

If X is uniformly distributed, i.e. $p_i = 1/M, i = 1, \ldots, M$, then $H(X) = \log_2 M$. Given X and another random variable Y that assumes values $y_1, \ldots, y_{M'}$, the conditional entropy of X given that we observe that Y takes a specific value y is

$$H(X|Y = y) = -\sum_{i=1}^{M} P(X = x_i|Y = y) \log_2 P(X = x_i|Y = y), \tag{3}$$

and the information that we get about X by observing $Y = y$ is $H(X) - H(X|Y = y)$.

The application of the concept of entropy in the context of solving systems of equations can then be described as follows.

1. **General equation system**: We wish to recover a solution $e = (e_1, \ldots, e_n) \in Z(F^2)$. We assume that the solution e is drawn uniformly from the space of all solutions $\mathbb{F}_2^n$. Then it follows that the entropy of the solution before working on the system is $H(e) = n$. The system F^2 that e must satisfy will reduce the entropy of e since not all possible solutions satisfy all equations in F^2. We can then think of F^2 as containing the *information* about the solution e and define
$$i(F^2) = n - \log_2(|Z(F^2)|) \tag{4}$$
For example, if F^2 has a unique solution we say that F^2 has n bits of information about the solution, and if half of the points in the space $\mathbb{F}_2^n$ satisfy the polynomials in F^2 we say the system contains 1 bit of information.

2. **Cipher equation system**: When F^2 is constructed from a plaintext/ciphertext pair of a particular block cipher, we are interested in finding the secret key, i.e. we are only interested in finding the values for $x_0, \ldots, x_{k-1}$ in a solution to the system. To be able to describe the cipher in terms of quadratic equations, several auxiliary variables $x_k \ldots, x_{n-1}$ are needed. In this case F^2 contains n variables where $n > k$. If we fix a key and assign values to $x_0, \ldots, x_{k-1}$, the known plaintext will dictate the values for all remaining variables. In this setting, the secret key is "controlling" the whole system of equations and the auxiliary variables are uniquely determined by the given plaintext pair along with the key, resulting in a ciphertext that may or may not be equal to the given one.

   For a given plaintext/ciphertext pair there will typically only be one or very few keys that fit the system. For equation systems created from ciphers it therefore makes more sense to say

that initially we have k bits of entropy on the secret key, and that the system F^2 contains

$$i(F^2) = k - \log_2(\text{Number of keys in } \mathbb{F}_2^k \text{ that satisfy } F^2), \tag{5}$$

bits of information on the secret key.

We note that this definition of information does not tell us anything about how to solve a system of equations. It only provides us with a measure of how much information about the secret key, or a solution, a system contains.

4.1 Information loss

Algorithm 2 eliminates one variable from a set of polynomials and creates a new set of polynomials not containing the eliminated variable. This is done without increasing the degree of the polynomials in the new set to more than d. Let $F_0 = F^2$, the initial system of equations. Repeatedly eliminating one variable using Algorithm 2 leads to a sequence of polynomial sets

$$F_0 \to F_1 \to \ldots \to F_j,$$

where F_j consists of polynomials in $B[x_j, \ldots, x_{n-1}]$.

In order to keep the degree bounded, we cannot compute the polynomials of too high degree that otherwise should have been in the F_j's, from some index j and onwards. Omitting these polynomials means that constraints given by some polynomials in F_{j-1} are not present in F_j. In terms of zero sets, we get the following chain of inclusions

$$Z(F_0) \subseteq Z(F_1) \subseteq \ldots \subseteq Z(F_j).$$

This sequence is in line with information theory's *data processing lemma*, that says that we can not get more information from processing some input than the input initially had. In other words, the sequence $i(F_0), i(F_1), i(F_2), \ldots$ is strictly non-increasing, since we can only lose information when we have to limit the degree in order to control the computational resources (complexity). At some point we can expect the sequence to start decreasing, which means that we start to lose information. In the case of systems created from a cipher this translates to having *false keys* as solutions, keys that fit in some F_j without being part of a solution for F_0.

If for some j the set F_j becomes empty, the whole space becomes the solution space and we have lost all information ($i(F_j) = 0$) about the original equation system. In the case of systems constructed from ciphers it is also possible to lose all information before F_j is empty. Even if there are polynomials left in some F_j, it may happen that all assignments of $x_0, \ldots, x_{k-1}$ will produce assignments of the auxiliary variables that fit the system.

As noted above, the notion of information loss does not tell us anything about *how* to solve a system of equations, it just enables us to detect when the degree restriction is too strict. If we can find an efficient algorithm to compute the function $i(F)$ for some arbitrary set of equations F, or an estimate of the function, the result could be applied towards guiding the dynamic adjustment of the degree bound in the elimination process. However, so far we do not know such an algorithm, and formally we may assume that an oracle can provide the evaluation of $i(F)$. We leave for future research the problem of computing $i(F)$ efficiently. For the purpose of testing the elimination approach in Section 5, we implement the evaluation of $i(F)$ by brute force.

5 Experimental results

We have implemented Algorithm 2 as the function *eliminate()* and done some experiments to see how it performs in practice. We have used reduced versions of some ciphers specified below. We

combined our algorithm with the information measure and information loss from Section 4. One expected finding that was confirmed by our experiments is that increasing the maximal allowed degree helps in keeping information in the system longer. In this section we report on these experiments.

5.1 Low degree polynomials in reduced LowMC cipher

LowMC is a family of block ciphers proposed by Martin Albrecht et al. [1]. The cipher family is designed to minimize the number of AND-gates in the critical path of an encryption, while still being secure. The cipher itself is a normal SPN network, with each round consisting of an S-box layer, an affine transformation of the cipher block and addition with a round key. All round keys are produced as affine transformations of the user-selected secret key.

Two features of the LowMC ciphers are interesting with respect to algebraic cryptanalysis. First, the S-box used is as small as possible without having linear relations among the input and output bits. LowMC uses a 3×3 S-box, where the ANF of each output bit only contains one multiplication of input bits, making the three output polynomials of the S-box quadratic. We can search for other quadratic relations in the six input/output variables, and we then find 14 linearly independent quadratic polynomials.

Second, the S-boxes in one round do not cover the whole state, so a part of the cipher block is not affected by the S-box layer. The number of S-boxes to use in each round is a parameter that varies within the cipher family, and some variants are proposed with only one S-box per round.

The cipher parameters we have used for the reduced LowMC version of our experiments are:

- Block size: 24 bits
- Key size: 32 bits
- 1 S-box per round
- 12, 13 or 14 rounds

As will become clear below, the number of rounds is degree dependent when *eliminate()* is successful in breaking the reduced cipher, and the degree is lower than anticipated.

5.1.1 Constructing equation system.

The attack is a known plaintext attack, where we assume we are given a plaintext/ciphertext pair and the task is to find the unknown key. We use the 14 quadratic polynomials describing the S-box as the base equations. The bits in the unknown key are assigned as the variables $x_0, \ldots, x_{31}$, and the output bits from each S-box used in the cipher are the variables $x_{32}, \ldots$. All other operations in LowMC are linear, so the input and output bits of every S-box can be written as a linear combination of the variables defined and the constants from the plaintext and ciphertext.

Inserting the actual linear combination for each input/output bit of the S-box in one round will produce $14r$ equations in total, where r is the number of rounds. These equations describe a complete LowMC encryption. The initial number of variables is $32 + 3r$, but this can be reduced by using the known ciphertext. The bits of the cipher block output from the last round are linear combinations of variables. These linear combinations are set to be equal to the known ciphertext bits, giving 24 linear equations that can be used to eliminate 24 variables by direct substitution. After this the final number of variables is $8 + 3r$. See Figure 1 for the equation setup.

5.1.2 Experimental results.

The goal of our experiment is to try to eliminate all the variables x_i for $i \geq 32$, and find some polynomials of degree at most 3, only in variables representing the unknown user-selected key. If we are able to find at least one polynomial only in $x_0, \ldots, x_{31}$ for one given plaintext/ciphertext

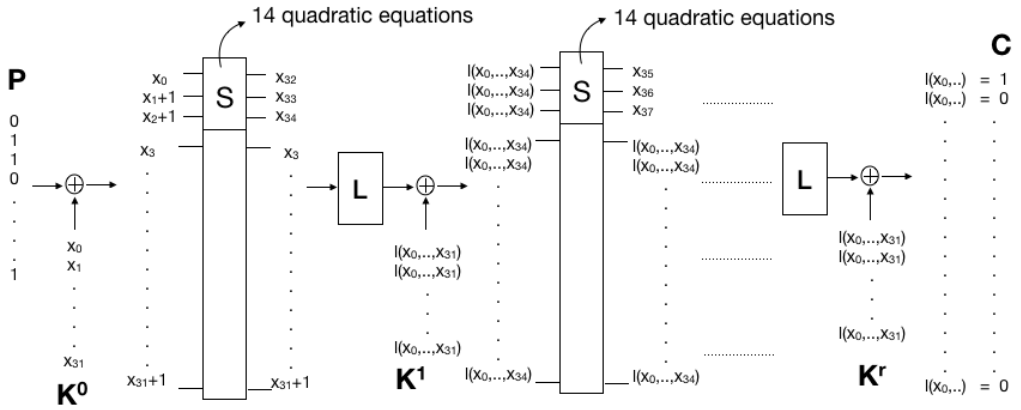


Figure 1: Setup of equation system representing reduced LowMC. All $l(.)$'s only indicate some linear combination, and are not equal.

pair, we can repeat for other known plaintext/ciphertext pairs and build up a set of equations that can be solved by re-linearization when the set has approximately $\binom{32}{3}$ independent polynomials.

12 rounds: The system initially contains 44 variables and 168 quadratic equations.

We applied the *eliminate()* algorithm on the system, limiting the degree to $d = 3$. As the algorithm proceeds, eliminating one variable at the time, the sizes of the output sets change. The number of degree 3 polynomials grows at first before starting to decrease before the last variables are eliminated. We note in particular that the size of the output was never above the 168 input polynomials which means that the computational complexity is low. In the end, after eliminating all the x_i for $i \geq 32$ we where left with 9 cubic polynomials in only key variables $x_0, \ldots, x_{31}$.

13 rounds: The initial system contains 47 variables and 182 quadratic equations.

When applying *eliminate()* algorithm on the system and limiting the degree to $d = 3$, we did not find any cubic polynomials in only $x_0, \ldots, x_{31}$ for any 13-round systems we tried. However, if we increase the allowed degree to 4 after eliminating all but the two last variables, we found two polynomials of degree 4 in only the key variables $x_0, \ldots, x_{31}$. So we see that increasing the degree at certain stages improves the result of the algorithm.

14 rounds: The initial system contains 50 variables and 196 quadratic equations.

When applying *eliminate()* algorithm on the system, and limiting the degree to either $d = 3$ or $d = 4$, we did not find any polynomials in only $x_0, \ldots, x_{31}$ for any 14-round systems we tried. However, if we applied *eliminate()* restricting the degree to $d = 4$ and then extended the degree to 5 after eliminating all but the last auxiliary variable x_{32}, we found 102 polynomials of degree 5 in only the key variables $x_0, \ldots, x_{31}$. This verifies that the effect of extending the degree at certain stages improves the result of the algorithm, and also improves the number of rounds we expect can be attacked by our algorithm.

The above results can also be described in terms of information loss, since the information measure function $i(F_j)$ is strictly non-increasing for increasing j. This means that all information is lost when we get the empty set of polynomials since $i(\emptyset) = 0$. In the cases studied above we obtain *some* polynomials in only key variables, and our interpretation of this is that the resulting system of equations contains information about the user selected key. However, because of the key length of 32 bits we were not able to do the necessary exhaustive search to perform information loss experiments on this cipher with the current tools available.

For the various systems we have repeated the experiments using different plaintext/ciphertext pairs. Different independent polynomials were produced for different p/c-pairs, as expected. However, we also found that in the 12-round cases we looked at, much fewer than $\binom{32}{3}$ polynomials were

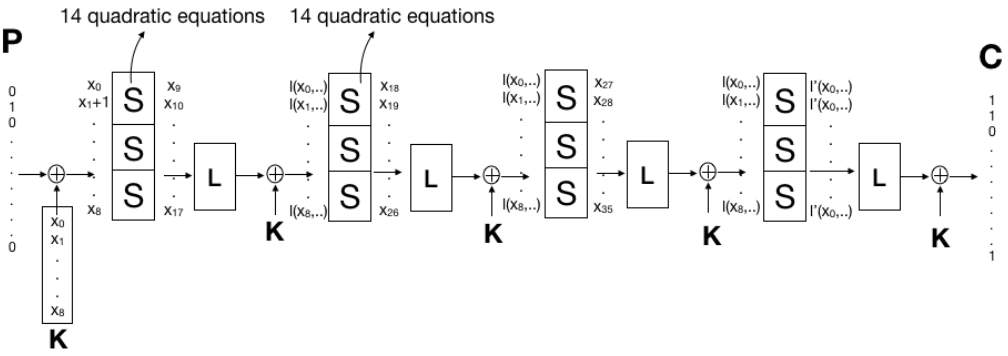


Figure 2: Cipher variant with 3 S-boxes per round

needed in order to solve for the unknown key. After collecting polynomials from approximately 20 different p/c-pairs, Gauss elimination on the resulting set started to produce linear polynomials in the key bits. We also verified that the true key actually satisfied this polynomial set.

5.2 Small scale ciphers with LowMC S-box

Next we defined some small block ciphers and constructed equation systems representing the ciphers with some given plaintext/ciphertext pairs. The ciphers were made as ordinary SPN networks, based on the 3×3 S-box found in LowMC [1] and linear transformations of the whole block. See Figure 2 for details of one variant with 3 S-boxes per round and four rounds, including the labelling of variables.

We have run experiments on 3-, 4- and 5-round variants of this cipher, with 3 or 4 S-boxes per round for testing the performance of *eliminate()* together with measurements of the information loss on the key. When running *eliminate()*, we did two types of experiments in order to get a better comparison of the results:

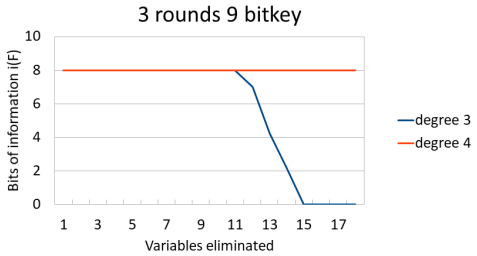
1. The first type of experiments we limit the degree first to $d = 3$, then right before eliminations where we know information loss occurs we extend the degree to $d = 4$ and so on. That is, we try to keep the degree low for as long as possible.

2. In the second type of experiments we set the maximal allowed degree in the polynomials during elimination be $d = 3$ or $d = 4$ from the start and never change it, and measure the information loss.

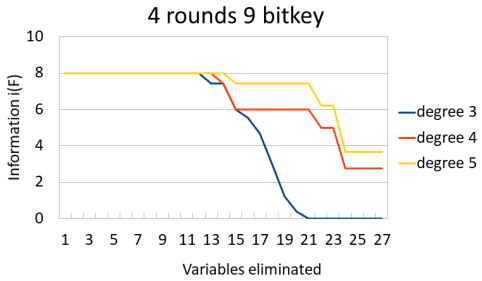
5.2.1 3, 4 and 5 rounds with 9-bit key

These cipher variants consists of 27, 36 and 45 variables together with 126, 168 and 210 quadratic polynomials for the 3, 4 and 5 round versions, respectively. Here $x_0, \ldots, x_8$ represent the unknown key. When the other variables were eliminated we measured how quickly the information on the key-variables were lost in the resulting systems. All these experiments are of type 1. Figures 3a and 3b show the results for 3 and 4 rounds, and Figure 4a shows the results for 5 rounds.

As can be seen in Figures 3 and 4a, all information is kept in the systems for many eliminations. For the case $d = 3$, all information gets lost while eliminating the eleventh to fifteenth variables. When we increase the allowed degree to $d = 4$ or $d = 5$ at the point where we start to lose information, we see that no information is lost in the 3-round case and that some, but not all, information gets lost in the 4-round case. For the 5-round case all information is eventually lost, even when increasing to $d = 5$.

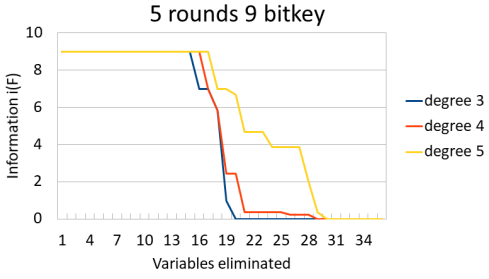


(a) Information loss for 3-round system

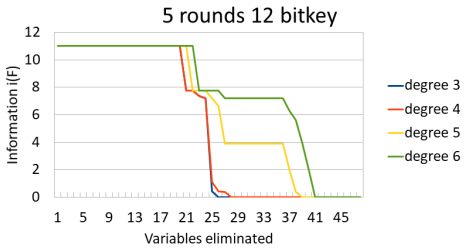


(b) Information loss for 4-round system

Figure 3: Information loss with 9-bit key. Degree only increased when information loss is about to occur.

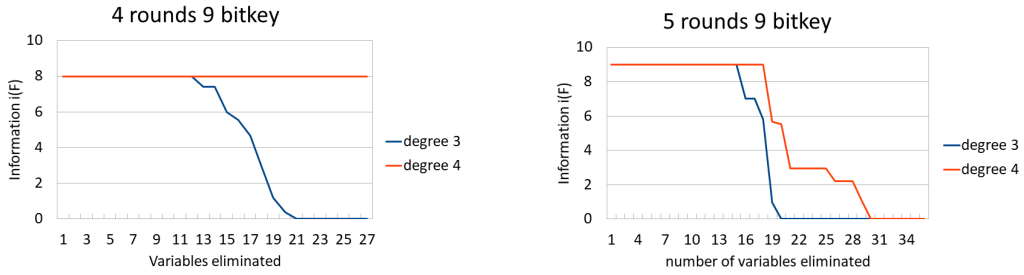


(a) 9-bit key.



(b) 12-bit key.

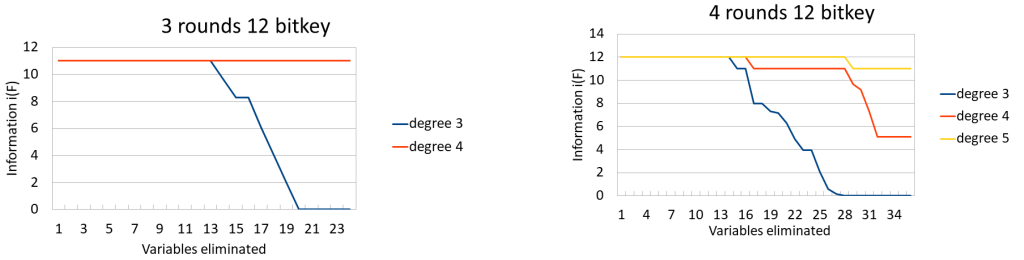
Figure 4: Information loss for 5-round systems. Degree only increased when information loss is about to occur.



(a) Information loss for 4-round system.

(b) Information loss for 5-round system.

Figure 5: Information loss with $d = 3$ and $d = 4$ from start



(a) Information loss for 3-round system.

(b) Information loss for 4-round system.

Figure 6: Information loss with 12-bit key. Degree only increased when information loss is about to occur.

We also did experiments of type 2, and measured how quickly the information on the key-variables are lost in the resulting systems when we start with $d = 4$ and keep the degree bound fixed throughout. These information plots are shown in Figure 5. We observe that information is kept longer than for $d = 4$ in Figures 3b and 4a.

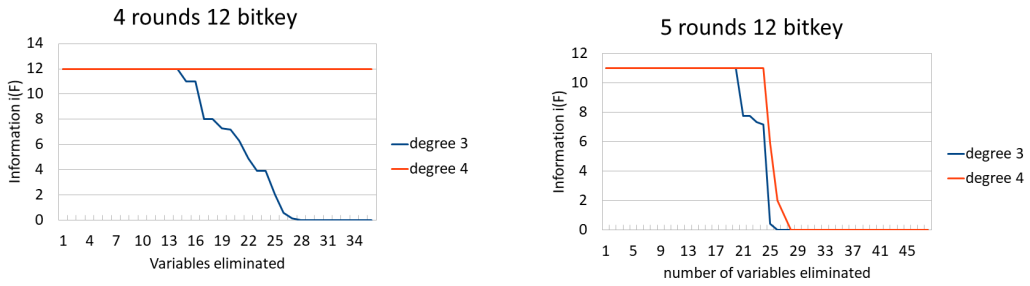
We note a few observations from this. First, increasing the degree certainly helps in keeping the information in the systems longer. Secondly, increasing the degree bound earlier also helps in keeping the information in a system longer. Finally, there is a degree bound for which no information is lost even when eliminating all auxiliary variables. These plots are the ones shown as straight lines. This indicates there exists some degree bound allowing all auxiliary variables in a system to be safely eliminated, without losing any information. In all cases where we have *some* information left after eliminating all auxiliary variables, we can break the associated cipher by collecting more polynomials in only key variables using other plaintext/ciphertext pairs.

5.2.2 3, 4 and 5 rounds with 12-bit key

These cipher variants consists of $36, 48$ and 60 variables together with $168, 224$ and 280 quadratic polynomials for the 3-, 4- and 5-round versions, respectively. Here the 12 variables $x_0, \ldots, x_{11}$ are the unknown key. We did the same experiments as in the case for 9-bit keys, measuring when and how quickly information on the key-variables are lost in the resulting systems.

Figures 6 and 4b show the results for Experiment 1 for $3, 4$ and 5 rounds, respectively. Figure 7 shows how the information on key-variables are lost in Experiment 2 during elimination for degree bounds $d = 3$ and $d = 4$.

We see the same pattern as in the 9-bit key case. Increasing the allowed degree has a significant



(a) Information loss for 4-round system.

(b) Information loss for 5-round system.

Figure 7: Information loss with $d = 3$ and $d = 4$ from start.

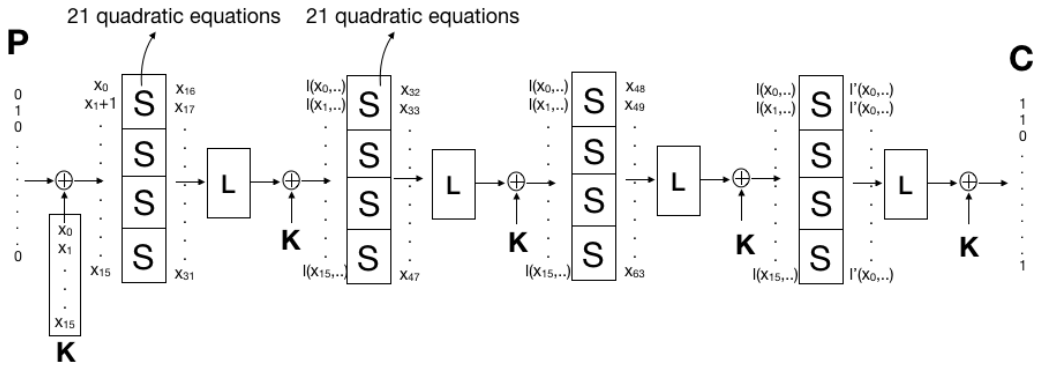


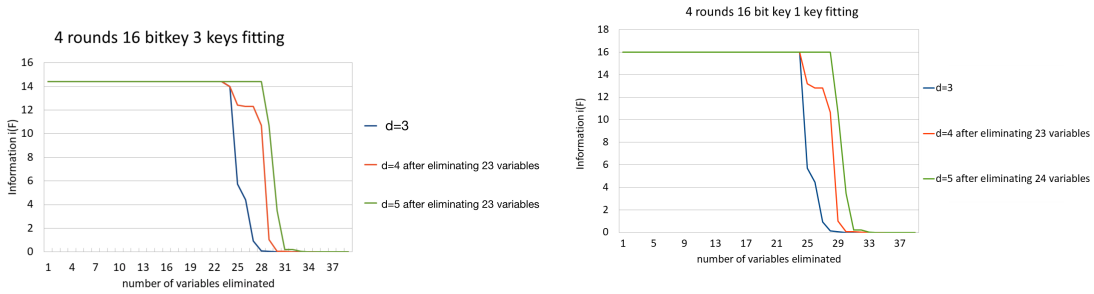
Figure 8: Setup of equation system representing 4-round toy cipher. All $l(.)$'s and $l'(.)$'s indicate some linear combination of variables.

impact on keeping information in the systems longer, but more rounds make it harder to eliminate all variables without losing all information on the key. We also see in these experiments that it matters exactly when we increase the allowed degree to $d = 4$. Having the allowed degree set to 4 from the beginning keeps information in the systems longer than if we increase the degree bound to $d = 4$ right before $d = 3$ starts to lose information.

5.3 Toy cipher based on PRINCE

For these experiments we made a small 4-round toy cipher for testing, where we have increased the size of the S-box. The toy cipher has a 16-bit block and a 16-bit key, and is built as a normal SPN network. Each round consists of an S-box layer with four 4×4 S-boxes (the same S-box as used in PRINCE), followed by a linear transformation and a key addition. The same key is used in every round, and the equation system representing the toy cipher is constructed similarly to the reduced LowMC systems. Each output bit of the PRINCE S-box has degree 3 when written as a polynomial of the input bits, but there exist 21 quadratic relations in input/output variables describing the S-box. The number of quadratic equations in the 4-round toy cipher is therefore 336, in 64 variables, where $x_0, \ldots, x_{15}$ represent the secret key. See Figure 8 for the setup of the equations.

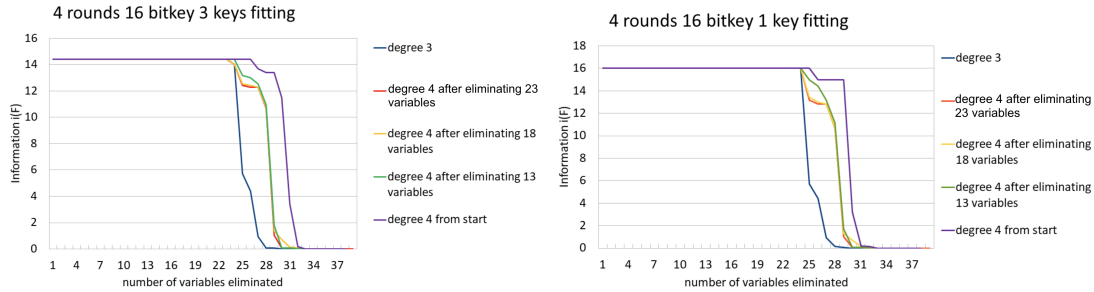
We have run experiments on this cipher with two different plaintext/ciphertext pairs. For the first p/c-pair there are 3 distinct keys that fit the initial system, and for the second there is 1 unique key that fits the initial system. Because of computational restrictions, we limit ourselves to



(a) 3 keys satisfying initial system.

(b) Unique key satisfying initial system.

Figure 9: Information loss for toy cipher, where we extend the degree to $d = 4$ and $d = 5$ after eliminating 23 (figure 9a) and 24 variables (figure 9b) respectively.



(a) 3 keys satisfying initial system.

(b) Unique key satisfying initial system.

Figure 10: Information loss for toy cipher, where we extend the degree to $d = 4$ after eliminating $0, 13, 18$ or 23 variables.

degrees varying between $d = 3$ and $d = 5$. When testing the performance of *eliminate()* together with information loss, we did the following two types of experiments:

1. This is the same experiment as in type 1 above, where we only increase the degree bound at the point we are about to start losing information.

2. In the second type of experiment we increase the maximal allowed degree from $d = 3$ to $d = 4$ after eliminating some number of variables, to see how much it helps to increase the degree bound early.

5.3.1 Experimental results and information measure

Figure 9 shows the plot for information loss in Experiment 1. We see that increasing the degree to $d = 4$ and $d = 5$ only for the latest eliminations helps a little in retaining the information, but not for very long.

The information plots for Experiment 2 are shown in Figure 10. These plots are interesting as they show that increasing the degree earlier helps in keeping the information in the system longer. The earlier we increase to $d = 4$, the longer we can keep information in the system.

6 Conclusion

In [25] it was shown that surprisingly few eliminations may be sufficient to solve a system of Boolean equations if we allow the degree of the polynomials to grow with one for each variable

eliminated. However, even the modest linear growth in the degree may lead to too heavy solving complexity, as several variables still need to be eliminated before a solution is found. In this paper we restrict the degree even more, by keeping it fixed up to some number of eliminations, and only increasing it at given points.

This approach has two advantages. First, keeping the degree fixed means that we have full control over the complexity, and can attempt to solve bigger systems in practice without running out of memory. Second, once a number of variables have been eliminated, increasing the degree by one will not lead to more monomials in the Macaulay matrix than what it had with the full variable set to begin with. In other words, whenever $\binom{n}{d} \geq \binom{n-k}{d+1}$, we can increase the allowed degree after k eliminations without increasing the overall complexity.

When eliminating variables from an equation system and keeping a bound on the allowed degree, we may introduce false solutions into the new systems, that do not correspond to solutions in the initial system. This leads to the question of how much information a system contains, measured in terms of how large part of the solution space that fits a given system. When eliminating variables, the systems may lose information due to the restriction on the allowed degree.

We measured this information loss for systems representing block ciphers. The different systems all show the same general behavior: All information is kept in the system for rather many eliminations, but is then lost rather abruptly at some point when eliminating more variables. As could be expected, increasing the degree helps, and increasing it sooner rather than later also helps in retaining the information in a system longer. We showed that the simplest ciphers we looked at could be broken by eliminating all non-key variables, and still having information left in the system.

One can say that the findings in this paper opens up more questions than it answers. First and foremost of these is how much and how early the bound of allowed degrees must be increased to solve a system representing a block cipher. A second question is how the number of equations influences solving complexity using our approach. As shown in [25], the solving approach we take here benefits significantly from overdetermined equation systems. Finally, a third question is whether it is possible to estimate when the information loss in a system is about to occur, without measuring it via exhaustive search. All of this leaves a lot nice questions to investigate in this direction.

References

[1] M. Albrecht, C. Rechberger, T. Schneider, T. Tiessen, M. Zohner. *Ciphers for MPC and FHE,* Eurocrypt 2015, LNCS 9056, pp. 430 – 454, Springer, 2015.

[2] Bard, G. V. *Algebraic Cryptanalysis*. Springer. ISBN 978-1-4419-1019-6. (2009).

[3] C. Bouillaguet, H.-C. Chen, C.-M. Cheng, T. Chou, R. Niederhagen, A. Shamir and B.-Y. Yang. *Fast exhaustive search for polynomial systems in* $\mathbb{F}_2$. In Cryptographic hardware and embedded systems – CHES 2010. 12th international workshop, Santa Barbara, USA, August 17–20, 2010. Proceedings, pages 203–218. Berlin: Springer, 2010.

[4] J. Buchmann, A. Pyshkin, and R. P. Weinmann, *A zero-dimensional Gröbner basis for AES-128*, Proc. of FSE 2006, LNCS, vol. 4047, Springer, Berlin, 2006a, pp. 78–88.

[5] M. Brickenstein and A. Dreyer, *PolyBoRi: A framework for Gröbner basis computations with Boolean polynomials*, Elec. Proc. of MEGA 2007, http://www.ricam.oeaw.ac.at/mega2007/electronic/26.pdf.

[6] D.Cox, J.Little, D.O'Shea, *Ideals, varieties and algorithms*, Third edition, 2007 Springer Science and Business Media.

[7] D.Cox, J.Little, D.O'Shea *Using Algebraic Geometry* GTM 185, Springer Science and Business Media 2005.

[8] C. Cid, S. Murphy, and M.J.B. Robshaw, *Small scale variants of the AES*, Proc. of FSE 2005, LNCS, vol. 3557, Springer Berlin, 2005b, pp. 145-162.

[9] Cover, Thomas M. and Thomas, Joy A., *Elements of Information Theory, 2nd Edition*, Wiley, 2006.

[10] N. T. Courtois and G. V. Bard, *Algebraic cryptanalysis of the data encryption standard*, Cryptography and Coding, LNCS, vol. 4887, Springer, Berlin, 2007, pp. 152–169.

[11] D. Lokshtanov, R. Paturi, S. Tamaki, R. Williams, and H. Yu. *Beating brute force for systems of polynomial equations over finite fields.* The 27th ACM-SIAM Symposium on Discrete Algorithms (SODA 2017).

[12] Kipnis A., Shamir A. *Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization.* Advances in Cryptology — CRYPTO' 99. CRYPTO 1999. Lecture Notes in Computer Science, vol 1666, pp. 19 – 30. Springer, Berlin, Heidelberg 1999.

[13] A. Shamir, J. Patarin, N. Courtois, A. Klimov, *Efficient Algorithms for solving Overdefined Systems of Multivariate Polynomial Equations*, Eurocrypt'2000, LNCS 1807, pp. 392 — 407, Springer 2000.

[14] Courtois N.T., Pieprzyk J. *Cryptanalysis of Block Ciphers with Overdefined Systems of Equations*, Advances in Cryptology — ASIACRYPT 2002. ASIACRYPT 2002. Lecture Notes in Computer Science, vol 2501, pp. 267 – 287. Springer, Berlin, Heidelberg 2002

[15] Murphy S., Robshaw M.J. *Essential Algebraic Structure within the AES.* Advances in Cryptology — CRYPTO 2002. CRYPTO 2002. Lecture Notes in Computer Science, vol 2442, pp. 1 – 16. Springer, Berlin, Heidelberg 2002

[16] Biryukov A., De Cannière C. *Block Ciphers and Systems of Quadratic Equations*, Fast Software Encryption, FSE 2003. Lecture Notes in Computer Science, vol 2887, pp. 274 – 289. Springer, Berlin, Heidelberg 2003

[17] J-C. Faugere. *A new efficient algorithm for computing Gröbner bases (F4). Effective methods in algebraic geometry* (Saint-Malo, 1998), J. Pure Appl. Algebra 139 (1999)

[18] J-C. Faugere. *A new efficient algorithm for computing Gröbner bases without reduction to zero (F5)*, Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation, 75–83, ACM, New York, 2002.

[19] M. Sala, T. Mora, L. Perret, S. Sakata, C. Traverso, *Gröbner Bases, Coding and Cryptography*, Springer, 2009

[20] G. Ars, J. C. Faugère, H. Imai, M. Kawazoe, and M. Sugita, *Comparison between XL and Gröbner basis algorithms*, Proc. of Asiacrypt 2004 (P. J. Lee, ed.), LNCS, vol. 3329, Springer, Berlin, 2004, pp. 338–353.

[21] C. Cid and G. Leurent, *An analysis of the XSL algorithm*, Proc. of ASIACRYPT 2005, LNCS, vol. 3788, Springer, Berlin, 2005, pp. 333–352.

[22] C. W. Lim and K. Khoo, *Detailed analysis on XSL applied to BES*, Proc. of FSE 2007, LNCS, vol.4593, Springer, Berlin, 2007, pp. 242–253.

[23] A. Joux and V. Vitse, *A crossbred algorithm for solving Boolean polynomial systems*, Cryptology ePrint Archive, Report 2017/372, 2017, https://eprint.iacr.org/2017/372.

[24] B. Greve, H. Raddum, G.Fløystad and Ø. Ytrehus, *Variable Elimination from non-linear Boolean equation systems*, Submitted to Journal of symbolic computation, may 2018

[25] B. Greve, H. Raddum G.Fløystad and Ø. Ytrehus, *Solving non-linear boolean equation systems by variable elimination*. To be submitted.

[26] Bayer, D., Stillman, M.: *A theorem on refining division orders by the reverse lexicographic order*. Duke Math. J. 55(2), 321–328 (1987)

[27] T. Shimoyama and T. Kaneko, *Quadratic relation of S-box and its application to the linear attack of full round DES*, Proc. of CRYPTO 1998, LNCS, vol. 1462, Springer, Berlin, 1998, pp. 200–211.

[28] P.Zajac, *Upper bounds on the complexity of algebraic cryptanalysis of ciphers with a low multiplicative complexity*, Designs, Codes and Cryptography January 2017, Volume 82, Issue 1–2, pp 43–56.