IST-2002-507932

ECRYPT

European Network of Excellence in Cryptology

Network of Excellence

Information Society Technologies

D.STVL.7
Algebraic Cryptanalysis of Symmetric Primitives

Due date of deliverable: 31. July 2008
Actual submission date: 18. July 2008

Start date of project: 1. February 2004 Duration: 4 years

Lead contractor: Royal Holloway, University of London (RHUL)

Revision 1.0

Project co-funded by the European Commission within the 6th Framework Programme		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission services)	
RE	Restricted to a group specified by the consortium (including the Commission services)	
CO	Confidential, only for members of the consortium (including the Commission services)	

Algebraic Cryptanalysis of Symmetric Primitives

Editor
Carlos Cid (RHUL)
Contributors
Martin Albrecht (RHUL), Daniel Augot (INRIA),
Anne Canteaut (INRIA), Ralf-Philipp Weinmann (TU Darmstadt)

18. July 2008
Revision 1.0

The work described in this report has in part been supported by the Commission of the European Communities through the IST program under contract IST-2002-507932.

Contents

Chapter 1

Introduction

The recent development of algebraic attacks can be considered an important breakthrough in the analysis of symmetric primitives; these are powerful techniques that apply to both block and stream ciphers (and potentially hash functions). The basic principle of these techniques goes back to Shannon's [74] work: they consist in expressing the whole cryptographic algorithm as a large system of multivariate algebraic equations (typically over $\mathbb{F}_2$), which can be solved to recover the secret key. Efficient algorithms for solving such algebraic systems are therefore the essential ingredients of algebraic attacks.

Algebraic cryptanalysis against symmetric primitives has recently received much attention from the cryptographic community, particularly after it was proposed against some LFSR-based stream ciphers [40] and against the AES and Serpent block ciphers [43]. This is currently a very active area of research. In this report we discuss the basic principles of algebraic cryptanalysis of stream ciphers and block ciphers, and review the latest developments in the field. We give an overview of the construction of such attacks against both types of primitives, and recall the main algorithms for solving algebraic systems. Finally we discuss future research directions.

Chapter 2

Algebraic Cryptanalysis of Symmetric Primitives

Algebraic attacks represent a new approach to cryptanalysis. In contrast to conventional methods of cryptanalysis, these new techniques are primarily algebraic rather than statistical; they exploit the intrinsic algebraic structure of the cipher. Algebraic attacks are in principle applicable to both block ciphers and stream ciphers.

Block Ciphers. In the most common form of algebraic attack, the attacker expresses the encryption transformation as a large set of multivariate polynomial equations, and subsequently attempts to solve the system to recover information about the encryption key. Algebraic attacks represent an exciting new development in cryptology, as it opens new perspectives in block cipher cryptanalysis. For example, only a handful of plaintext–ciphertext pairs is usually required in algebraic cryptanalysis. Furthermore, it is expected that if an algebraic attack proves to be successful against a particular cipher, it might not be easily avoided by simply increasing the number of rounds. We note however that thus far algebraic attacks have had limited success against block ciphers. We discuss algebraic cryptanalysis of block ciphers in more detail in Chapter 3 (the article [31] gives an accessible overview of algebraic cryptanalysis on block ciphers and related algebraic methods).

Stream Ciphers. In contrast, algebraic attacks have been much more effective in the analysis of several LFSR-based stream ciphers [35]. The attack exploits the fact that each new bit of the key stream gives a new equation on the secret state bits. By collecting a large number of bits from the key stream, one can construct a system of equations that can be solved using one of the methods discussed in this report. We discuss algebraic cryptanalysis of stream ciphers in more detail in Chapter 4.

In algebraic attacks, a cryptanalyst describes the encryption operation as a large set of multivariate polynomial equations, which once solved can be used to recover the secret key. Thus the difficulty of solving systems of equations arising from a cipher is directly related to its security, and as a result computational algebra is becoming an important tool for the cryptanalysis of symmetric-key cryptographic algorithms. In the following sections, we give a brief overview of the main techniques used for solving systems of multivariate polynomial equations, with special focus to methods used in cryptology.

2.1 General Techniques for Solving Polynomial Systems

Solving multivariate polynomial systems is a typical problem studied in Algebraic Geometry and Commutative Algebra. In this section, we focus on the main algorithms for solving algebraic systems, in the context of cryptology. Our discussion will go from the simplest to the most efficient algorithms, that is from the linearisation principle to F_4 and F_5, through XL and Buchberger algorithms, although this does not respect the chronological order of discovery of these algorithms. We will also discuss some more recently proposed algorithms and strategies, some of which have been specifically proposed in the context of cryptology. We also include some discussion on the complexity estimates of some of these algorithms.

The problem. Let k be a field and $f_1, \ldots, f_m$ be polynomials in n variables with coefficients in k, i.e. $f_i \in k[X_1, \ldots, X_n]$, for $i = 1, \ldots, m$. Let K be an algebraic extension of k. The problem is to find $(x_1, \ldots, x_n) \in K^n$ such that $f_i(x_1, \ldots, x_n) = 0$, for $i = 1, \ldots, m$. Note that the problem may have no solution (inconsistency of the equations), a finite number of solutions, or an infinite number of solutions (when the system is underdefined and K is the algebraic closure of k).

This problem is most often studied in the context of abstract algebra. More precisely, let $I \subseteq k[X_1, \ldots, X_n]$ be the *ideal* generated by $f_1, \ldots, f_m$ and

$$V_K(I) = \{(x_1, \ldots, x_n) \in K^n; \quad f_i(x_1, \ldots, x_n) = 0, \text{ for } i = 1 \ldots m\}$$

be the *variety* over K associated to I. The problem is then to find $V_K(I)$.

When k is a finite field of order q, one can always add to the existing set of equations the so-called field equations $X_i^q = X_i$, for $i = 1, \ldots, n$, and obtain $m + n$ equations. For most cryptographic applications, the case of interest is when $k = K = \mathbb{F}_2$. In this case, the field equations are $X_i^2 = X_i$. This preprocessing step has the following consequences: the space of solutions is 0-dimensional (or empty), including at "infinity", and the ideal becomes radical (i.e. the solutions are of multiplicity one).

2.1.1 Linearisation

The method of *linearisation* is a well-known technique for solving large systems of multivariate polynomial equations. In this method, one considers all monomials in the system as independent variables and tries to solve the system using linear algebra techniques. More precisely, let A be the set of multi-indices $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$, which represent the exponents of the monomials of $k[X_1, \ldots, X_n]$. Then any polynomial f can be written as $f = \sum_{\alpha \in A} c_\alpha X^\alpha$, where the sum involves only a finite number of monomials $X^\alpha = X_1^{\alpha_1} \cdots X_n^{\alpha_n}$. Using this notation, we can write the following matrix M_{L}:

$$\begin{array}{c} \\ f_1 \\ \vdots \\ f_m \end{array} \begin{array}{c} \begin{array}{ccc} \ldots & X^\alpha & \ldots \end{array} \\ \begin{pmatrix} \ldots & c_\alpha^1 & \ldots \\ & & \\ \ldots & c_\alpha^j & \ldots \end{pmatrix} \end{array} = M_{\mathrm{L}},$$

where $f_i = \sum_\alpha c_\alpha^i X^\alpha$. Note that the columns of the matrix can be arranged in different ways, depending on the order chosen to sort the multi-indices α.

To apply linearisation, one now considers each (non-constant) monomial X^α as an indeterminate and attempts to solve the corresponding system of linear equations using linear algebra techniques.

The effectiveness of the method clearly depends of the number of linearly independent polynomials in the system. For example, in the case of polynomials belong to the Boolean ring, the total number of monomials of degree less than or equal to 2 (excluding the constant) is $\binom{n}{2} + n$. Thus if the system consists of m polynomials of degree 2, it can be solved if the matrix M_{L} has this rank. Note that the method also tolerates a smaller rank: it is possible to perform an exhaustive search on the affine space of solutions when the dimension of the kernel of the matrix is not too large.

Concerning the complexity, we observe that the cost of the linear algebra operations is $O(N^3)$, N being the size of the matrix M_{L}. We may theoretically write $O(N^\omega)$, ω being the exponent of linear algebra, and sometimes even optimistically use $\omega \approx 2 + \epsilon$ in the case of sparse matrices.

Linearisation has been considered mostly in the cryptanalysis of LFSR-based, filtered, stream ciphers. As stated earlier, each new bit of the key stream gives rise to a new equation on the key bits, and by using a large number of bits from the key stream, one should have in theory enough equations to directly apply linearisation. We note that only a few (if any) *practical* attacks have been reported to have been implemented using linearisation.

2.1.2 The XL algorithm and variants

In order to apply the linearisation method, the number of *linearly independent* equations in the system needs to be approximately the same as the number of terms in the system. When this is not the case, a number of techniques have been proposed that attempt to generate enough linearly independent equations. The most prominent is the XL algorithm (standing for *eXtended Linearisation*), which was introduced in [39]. The XL algorithm aims at introducing new rows to the matrix M_{L}, by multiplication of the original equations by monomials of prescribed degree. More specifically, the following matrix M_{XL} is constructed:

$$\begin{array}{c} \\ \vdots \\ X^\beta f_1 \\ \vdots \\ X^{\beta'} f_m \\ \vdots \end{array} \begin{array}{c} \begin{array}{ccc} \dots & X^\alpha & \dots \end{array} \\ \begin{pmatrix} & \vdots & \\ \dots & c_\alpha^{1,\beta} & \dots \\ & \vdots & \\ \dots & c_\alpha^{j,\beta'} & \dots \\ & \vdots & \end{pmatrix} \end{array} = M_{\mathrm{XL}},$$

where the set of the rows is constructed from all products $X^\beta f_j = \sum_\alpha c_\alpha^{j,\beta} X^\alpha$, where β and f_j are such that $\deg(X^\beta f_j) \leq D$, D being a parameter of the algorithm. The hope is that at least one univariate equation (say in X_1) will appear after the Gaussian elimination on M_{XL}. This equation should be easily solved over the finite field, the found values substituted in the equations, and the process repeated for the other indeterminates X_i, $i \geq 2$. One expects that after a few iterations, the algorithm will yield a solution for the system. We note that, since the matrix M_{XL} is quite likely very sparse, another approach is to compute a random vector in the nullspace of this matrix via sparse matrix techniques. This vector can then be tested to check if it indeed yields the solution.

To estimate the complexity of the XL algorithm, the problem is to find D such that XL succeeds with parameter D. Since the number of monomials of total degree $\leq D$ in $k[X_1, \ldots, X_n]$ is equal to $\binom{n+D}{D}$, there is an exponential dependence on D.

Since the introduction of the XL method, a number of variants have been proposed attempting to exploit some specific properties of the polynomial system [42]. Very prominent is the method proposed in [44, 43]. The *XSL* method is based on the XL algorithm, but attempts to use the sparsity and specific structure of the equations; instead of multiplying the equations by all monomials of degree $\leq D - 2$ (supposing that the original equations were quadratic), in the XSL algorithm the equations are multiplied only by "carefully selected monomials" [43]. This has the intention to create less new terms when generating the new equations. Different versions of the XSL algorithm are found in literature, where the description of the method often leaves some room for interpretation and the analysis of the algorithm is not straightforward. However, it is now known [28] that, as presented in [43], the algorithm cannot solve the system arising from the AES, and some doubts are cast on whether the algorithm in its current form can provide an efficient method for solving the AES-like systems of equations. Furthermore, it has been recently shown in [60] that the XSL version proposed in [44] has a much higher complexity than expected and raised questions on whether the algorithm can work at all.

The XL algorithm has been generalised to an algorithm named *GeometricXL* in [66]. The key idea is the fact that when solving polynomial systems, both the problem formulation and the solution to the problem are geometric invariant, i.e. they are invariant under a linear coordinate transformation. This implies that there must be a geometric invariant algorithm to solve this problem. It was shown in [66] that the XL algorithm is a special case of an algorithm finding intersections of hyperplanes which the authors call *GeometricXL*. This generalisation allows a better understanding of XL and may offer some advantage in certain situations when compared to XL. Essentially the maximal degree D reached during a *GeometricXL* execution is the least possible degree reachable by XL under any linear coordinate transformation. As an illustration, if we consider the equation system

$$\begin{aligned} f_1 &= 15x_0^2 + x_1^2 + 5x_1x_2, \\ f_2 &= 23x_0^2 + x_2^2 + 9x_1x_2 \end{aligned}$$

over GF(37), an XL style algorithm will not need to increase the degree during the computation. However, if we apply a linear coordinate transform

$$\begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} \mapsto \begin{pmatrix} 2 & 26 & 10 \\ 26 & 4 & 13 \\ 33 & 21 & 2 \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix},$$

to obtain

$$\begin{aligned} f_1 &= 6x_0^2 + 2x_0x_1 + 3x_0x_2 + x_1^2 + 16x_1x_2 + 3x_2^2, \\ f_2 &= 18x_0^2 + 35x_0x_1 + 15x_0x_2 + 26x_1^2 + 12x_1x_2 + x_2^2, \end{aligned}$$

the system is only solveable by an XL style algorithm for $D = 4$.

For the example given above, *GeometricXL* solves the system with $D = 2$. Instead of searching for an univariate or bivariate polynomial in the linear span of the generated polynomials, it searches for any polynomial with an appropriate factorisation. However, a technical requirement of this algorithm is that the characteristic of the finite field k is larger than the maximal degree D. Thus, in particular for the case of GF(2), the algorithm is not applicable as described in [66].

Another proposed algorithm which can be considered as a variant of XL is the so-called *ElimLin* algorithm. The algorithm was first referred to in [37] and later described in [38]. The algorithm uses linear equations in the system to derive substitution rules. The algorithm essentially consists of two steps.

1. Substitute the leading term $\mathrm{LT}(l_i)$ of each linear polynomial l_i $(1 \leq i \leq o)$ by $l_i - \mathrm{LT}(l_i)$ in all polynomials p_j containing $\mathrm{LM}(l_i)$. This is equivalent to computing the remainder r_j by the routine $\texttt{division}(p_j, l_1, \ldots, l_o)$ discussed later in this section.
2. Perform Gaussian reduction on the polynomials p_j. If new linear polynomials arise, repeat step 1.

The performance of the *ElimLin* algorithm is related to the sparse selection strategy employed to choose the replacement variables; this strategy remains however unpublished.

2.1.3 Gröbner Basis Algorithms

Gröbner bases algorithms are perhaps the best known technique for solving polynomial systems. These algorithms return a basis for the ideal derived from the set of equations, which can then be used to obtain the solutions of the system. The most accessible historical reference is [21], while the book [45] presents a gentle introduction to the topic together with the basics of algebraic geometry (it does not however cover more recent algorithms, such as F_4 and F_5).

We now give a definition of a Gröbner basis of an ideal. Let $\preceq$ be a monomial order, i.e. a total order on the set of monomials X^α, $\alpha \in \mathbb{N}^n$, which is compatible with multiplication. Then the set of terms $c_\alpha X^\alpha$ of a polynomial $f = \sum_\alpha c_\alpha X^\alpha \in k[X_1, \ldots, X_n]$ can be ordered with respect to $\preceq$, and the notion of leading term $\mathrm{LT}(f)$, leading monomial $\mathrm{LM}(f)$ and leading coefficient $\mathrm{LC}(f)$ of the polynomial f are all well defined.

Let $I \subseteq k[X_1, \ldots, X_n]$ be an ideal and $\mathrm{LM}(I) = \{\mathrm{LM}(f) \ : f \in I\}$ the set of leading monomials of polynomials in I. A Gröbner basis of the ideal I is a set $G = \{g_1, \ldots, g_l\} \subset I$ such that

$$\langle \mathrm{LM}(I) \rangle = \langle \mathrm{LM}(g_i), \ldots, \mathrm{LM}(g_l) \rangle.$$

In other words, G is a Gröbner basis of I if the leading monomial of any polynomial in I is divisible by the leading monomial of some polynomial of G. One can show that every non-empty ideal $I \subseteq k[X_1, \ldots, X_n]$ has a Gröbner basis (which however is not unique). It is to be stressed that the notion of Gröbner basis is a mathematical one, independently of any algorithm computing Gröbner bases.

There is also the notion of a *Gröbner basis of degree* D of an ideal I (denoted by G_D), which has the property that the leading monomial of every polynomial in I of degree $\leq D$ is divisible by the leading monomial of a polynomial of G_D. It can be shown that there exists D large enough such that G_D is a Gröbner basis of I.

Gröbner bases algorithms are powerful tools for solving systems of polynomial equations. In most cases, when the Gröbner basis is found, the solution is also found. For most cryptographic applications, we will have a system with unique solution, say $(a_1, \ldots, a_n) \in \mathbb{F}_2^n$, and the ideal is radical. Then the *reduced* Gröbner basis of I is $\{X_1 - a_1, \ldots, X_n - a_n\}$.

The Buchberger algorithm

The Buchberger algorithm is the classical algorithm for computing the Gröbner basis of an ideal I. It works based on a generalisation of the Euclidean division of polynomials in one variable to the multivariate case. More precisely, given a monomial order, there exists an algorithm $\texttt{division}(f, f_1, \ldots, f_l) = (g_1, \ldots, g_l, r)$ with the following properties: $f = f_1 g_1 + \ldots f_l g_l + r$, and no leading monomial of the g_i divides r. Then a Gröbner basis of an ideal generated by $f_1, \ldots, f_l$ can be computed by the following algorithm (Buchberger algorithm):

Initialise: $G = \{f_1, \ldots, f_l\}$

Loop

1. Combine every pair f_i, f_j by cancelling leading terms, to get $S(f_i, f_j)$ (the S-polynomials);
2. Compute the remainders of the polynomials $S(f_i, f_j)$ by G;
3. Augment G with the non-zero remainders.

Until all remainders are zero.

Return G.

One can show that this algorithm terminates and computes a Gröbner basis of the ideal generated by $f_1, \ldots, f_l$. It is a fact that most S-polynomials generated in step 1 will reduce to zero, and therefore many useless computations leading to zero remainder are performed. The algorithm can be modified to include *Buchberger's criteria* [20], which are a priori conditions on the pairs (f_i, f_j) to detect the ones whose S-polynomial will have a remainder equal to zero, and therefore discard them from steps 1, 2 of the algorithm. While a great proportion of pairs will be discarded by the criteria, still many S-polynomials constructed will reduce to zero, as experienced in reported implementations.

The complexity of the Buchberger algorithm is closely related to the total degree of the intermediate polynomials that are generated during the running of algorithm. In the worst case, it is known to run in double exponential time. Regarding implementation, there are number of optimisations that can be made to improve the performance of the algorithm. For example, the main loop of the algorithm can be sliced into loops of finer grain, and instead of combining every possible pair, pairs can be successively selected with respect to some strategy; steps 2 and 3 can then be performed with this selection. For instance, the most recent pairs from G can be chosen; alternatively, the pairs of smallest total degree may also be chosen.

The F_4 algorithm

The F_4 algorithm [52] can be roughly sketched as a matricial version of the Buchberger algorithm. To introduce the idea, we first depict the Euclidean division for univariate polynomials

$f = f_d X^d + \cdots + f_1 X + f_0$ and $g = g_{d'} X^{d'} + \cdots + g_1 X + g_0$, with $d' \leq d$, as a matrix reduction algorithm. Consider the following:

$$\begin{array}{c} \\ f \\ X^{d-d'}g \\ X^{d-d'-1}g \\ X^{d-d'-2}g \\ \vdots \\ g \end{array} \begin{array}{c} \begin{array}{ccccccccc} X^d & X^{d-1} & & & & & & & X^0 \end{array} \\ \begin{pmatrix} f_d & f_{d-1} & \cdots & & & & & & f_0 \\ g_{d'} & g_{d'-1} & \cdots & g_0 & & & & & \\ 0 & g_{d'} & g_{d'-1} & \cdots & g_0 & & & & \\ 0 & 0 & g_{d'} & g_{d'-1} & \cdots & g_0 & & & \\ \vdots & & & \ddots & & & \ddots & & \\ 0 & 0 & 0 & 0 & g_{d'} & g_{d'-1} & \cdots & g_0 \end{pmatrix} \end{array} \tag{2.1}$$

Then successive reductions of the first row by the remaining rows (row echelon reduction by elementary row operations) give the remainder of f by g. Similarly the multivariate division algorithm can be written in a matrix fashion.

At each iteration of the F_4 algorithm, corresponding to each iteration in the Buchberger algorithm, and subject to the selection strategy, the two parts f_i and f_j of the selected pairs (f_1, f_j) are written in a global matrix M_{F_4}. Now the crucial point of the algorithm F_4 it to write, at a given step of the algorithm, all the considered f_i and f_j into this global matrix, together with all already known polynomials of the current basis G, multiplied by fitting monomials (in the same way as the polynomial g is shifted in the matrix in (2.1)).

Then, in a single step corresponding to one iteration in the Buchberger algorithm, a huge matrix reduction operation (computation of the row echelon form) is done on the matrix M_{F_4}. In contrast to the Buchberger algorithm, where each remainder is computed separately and sequentially, this global reduction operates all reductions of all polynomials by all multiples of polynomials in the current basis G. It turns out that, properly implemented, this is a big win [52, 75]. Additionally, the algorithm can also benefit from any optimisation technique from linear algebra algorithms that can be applied.

The F_5 algorithm

The F_5 algorithm [53] is essentially an improved criterion if the S-polynomial for two polynomials f_i and f_j reduces to zero. The main idea is to only consider trivial combinations that reduce to zero, i.e. to consider $f_i f_j - f_j f_i$. F_5 proceeds by tracking the construction of new polynomials to detect these sort of trivial anhiliators by iteratively computing the Gröbner bases for the systems $\{f_m\}, \{f_{m-1}, f_m\}, \ldots, \{f_1, \ldots, f_m\}$. In some well-defined cases this criterion detects all reductions to zero since they are all based on these trivial combinations. Several variants of F_5 exist among them is $F_5/Matrix$ which combines the advantages of F_4 with the F_5 criterion, In this case all matrices constructed during the course of the algorithm have full rank, if the input system is a *regular sequence*. A gentle introduction to F_5 can be found in [76].

Relationship between these algorithms

Recent research has shown that some of the algorithms introduced above are related. In fact, let M_∞ denote the Macaulay matrix with an infinite number of rows and columns, defined as

$$\begin{array}{c} \\ \vdots \\ X^\beta f_i \\ \vdots \\ X^{\beta'} f_j \\ \vdots \end{array} \begin{array}{c} \begin{array}{ccc} \cdots & X^\alpha & \cdots \end{array} \\ \begin{pmatrix} & \vdots & \\ \cdots & c_\alpha^{1,\beta} & \cdots \\ & \vdots & \\ \cdots & c_\alpha^{j,\beta'} & \cdots \\ & \vdots & \end{pmatrix} \end{array} = M_\infty,$$

for all monomials X^β, $X^{\beta'}$ of unbound degree. The M_{XL} matrix of the XL algorithm in degree D is therefore just a finite submatrix of the Macaulay matrix, corresponding to all monomials of degree less than or equal to D. Performing a Gaussian elimination on the Macaulay matrix is equivalent to running the Buchberger algorithm [58]. This fact is closely related to the behaviour of the XL algorithm, and it is shown in [8] that the XL algorithm terminates for a degree D if and only if it terminates in degree D for the lexicographical ordering.

Concerning F_4, we can see that the matrix

$$\begin{array}{c} \\ \vdots \\ X^\beta f_i \\ \vdots \\ X^{\beta'} f_j \\ \vdots \end{array} \begin{array}{c} \begin{array}{ccc} \cdots & X^\alpha & \cdots \end{array} \\ \begin{pmatrix} & \vdots & \\ \cdots & c_\alpha^{1,\beta} & \cdots \\ & \vdots & \\ \cdots & c_\alpha^{j,\beta'} & \cdots \\ & \vdots & \end{pmatrix} \end{array} = M_{F_4}$$

is constructed only from pairs (f_i, f_j) originating from the previous iterations of the algorithm, and which are not discarded by the Buchberger criteria. This shows that M_{F_4} is a very small submatrix of the matrix M_{XL} constructed by XL. Using an XL description as an F_4 algorithm, it is shown in [8] that *a slightly modified XL computes a Gröbner basis.*

We note that things are pushed further in the same vein, when one considers the F_5 algorithm, which constructs a matrix M_{F_5} with even less rows than M_{F_4}. In [8], an example is given for the case of 130 equations in 128 variables, where the number of rows in the matrix M_{XL} will be more than 10 thousands times the number of rows in the matrix M_{F_5}.

2.1.4 Recent Proposals and Strategies

We discuss below some recently proposed algorithms and strategies for solving system of equations, some of which have been specifically proposed in the context of cryptology.

SlimGB algorithm

SlimGB [18] is also an algorithm for computing Gröbner basis which is inspired by F_4. Similar to F_4, SlimGB also reduces several polynomials at once; however it does not rely on linear algebra techniques for the reduction step. The key concept of SlimGB is that a strategy can be employed during the reduction step to keep the polynomials "slim" by some metric.

Possible metrics are used to avoid coefficient growth or to keep the degree of the polynomials low. This algorithm is implemented in the computer algebra system SINGULAR [56] and the library POLYBORI [19].

PolyBoRi Library

System of equations arising from block ciphers and stream ciphers are usually described in the ring of Boolean functions, and are in general of significant relevance for algebraic cryptanalysis. It therefore makes sense to optimize algorithms to perform computations over such rings. For these systems, alternative representations are possible, for instance by representing Boolean functions as binary decision diagrams (BDD). A binary decision diagram is a rooted, directed, acyclic graph representing a Boolean function [59, 1]. The POLYBORI library [19] has been recently proposed, and uses BDDs for representing Boolean functions in Gröbner basis computations. More specifically, by using a more compact specialized version of BDDs called zero-suppressed binary decision diagram (ZDD) [64], the fact that the Boolean functions dealt with have a sparse polynomial representation is used. One of the main ideas behind POLYBORI is to preserve the sparsity of the representation throughout the computation; this allows to keep the memory consumption low. This is achieved by using an adapted version of the SlimGB algorithm [18].

The size of binary decision diagrams is very sensitive to the variable ordering. A change of the variable ordering can make the size of a ZDD grow exponentially in the worst case. Unfortunately, determining the optimal variable ordering – meaning one that minimizes the size of the representation – is an NP-complete problem [16]. At the same time, the natural order on ZDDs is lexicographical. This means that extracting the leading monomial with regards to the lexicographical term ordering is computationally cheap. Usually, a total degree ordering of the terms is employed during the Gröbner basis computation, making the naïve extraction of the leading term computationally costly. To work around this problem, the library authors devised caching tricks in the implementation that seem to work very well in practice. Both addition and multiplication of the Boolean functions are defined as recursive operations on the diagrams. This allows caching of expressions already computed, and common subexpressions to be reused.

Effectively, in the case of POLYBORI, the actual Gröbner basis computation is carried out in a polynomial ring over GF(2), the representation of the elements however is done in the ring of Boolean functions. This means that during the computation, the field equations have still to be taken into account. To this end, a new criterion for reducing the number of critical pairs during the computation was proposed.

Theorem 1 ([19])**.** *Let $f, g, l \in GF(2)[X_1, \ldots, X_n]$ such that $f = l \cdot g$ and furthermore let l have a linear leading term X_i. Then spoly$(f, X_i^2 + X_i)$ has a nontrivial t-representation against the system consisting of f and the field equations.*

From this follows that all pairs of a Boolean polynomial f and a field polynomial $X_i^2 + X_i$, in which X_i occurs in the irreducible nonlinear factor of f need not be considered during the computation.

The Raddum-Semaev Algorithms

In [68] a different approach to equation system solving over finite fields is presented; the approach entirely focuses on the solution set – the variety – rather than the polynomial

system. The key idea is to consider the varieties for each equation f_i restricted to the variables in the equation independently. The algorithm *Agreeing* considers pairs of restricted varieties for equations f_i and f_j and removes all those solutions – called configurations – from each restricted variety which are conflicting. These conflicting configurations cannot lead to a common solution. For reduced round block ciphers with very few rounds this algorithm is sufficient to produce the unique common solution. However, for a larger number of rounds the system can be brought into a conflict-free state without being reduced to a very small set of common solutions.

For this situation the *Glueing* and the *Splitting* algorithms were also presented. The former "glues" two varieties together, i.e. it considers all possible combinations of solutions for f_i with solutions for f_j. The later basically guesses one bit of information by considering only half of the possible solutions to a given equation f_i. If this guess leads to an empty solution set – i.e. two configurations contradict completely – the algorithm backtracks. Later the same authors introduced a generalised technique which does not consider multivariate polynomials over finite fields and their solution sets, but multiple right hand side linear (MRHS) equations [69].

SAT-Solvers

A new development in algebraic cryptanalysis of symmetric primitives is the use of SAT-solvers [11, 9] to solve systems of equations over GF(2). Here the cryptanalyst converts the equation system (which can be viewed as an expression of the system in Algebraic Normal Form) to the Conjunctive Normal Form (CNF) of Boolean expressions. An off-the-shelf SAT-solver software can then be used to solve the resulting SAT problem.

In Conjunctive Normal Form, literals (variables) and their negates are combined in clauses via logical OR ($\vee$). These clauses can be combined using logical AND ($\wedge$). To convert from ANF to CNF, every monomial (including the constant 1) is first renamed as a new variable. This results in a linear system in these new variables. Now, since logical XOR results in very long conjugations, all sums are split into subsums of a parameterised length by introducing new intermediate variables (for example, $a + b + c + e + d + f$ splits into $a + b + c + x$ and $x + d + e + f$). These equations are then converted to CNF.

The dominant family of SAT solvers in use today is the Chaff family [9]. The main idea is to use simplification rules, guessing and backtracking until a contradiction or a solution is found. Specifically, variables can have three possible values: true, false and not-yet-known. Any clause containing a true variable can be discarded, since it does not encode any further information. Any clause that has all variables set to false will trigger a backtrack. Now if we assume a clause has n variables and $n - 1$ are set to false while one is still not-yet-known, then this variable must be set to true. This assignment will affect other clauses and might trigger an avalanche effect. This rule is called "unit propagation" rule. If no further such simplifications can be made, then an assignment is guessed. If a contradiction is found, i.e. all variables in a clause are set to false, then the algorithm either backtracks and adds the negative of the last guess to the list of clauses or – if the algorithm cannot backtrack – then it will just return "unsatisfiable".

The 3-SAT problem is a well known NP-hard problem, so the average runtime of a SAT solving algorithm is expected to be exponential. However, due to the great demand for SAT solvers many good implementations exist with good heuristics. Yet, no tight complexity bounds exist and since the algorithms are randomised, assessing the runtime for a particular

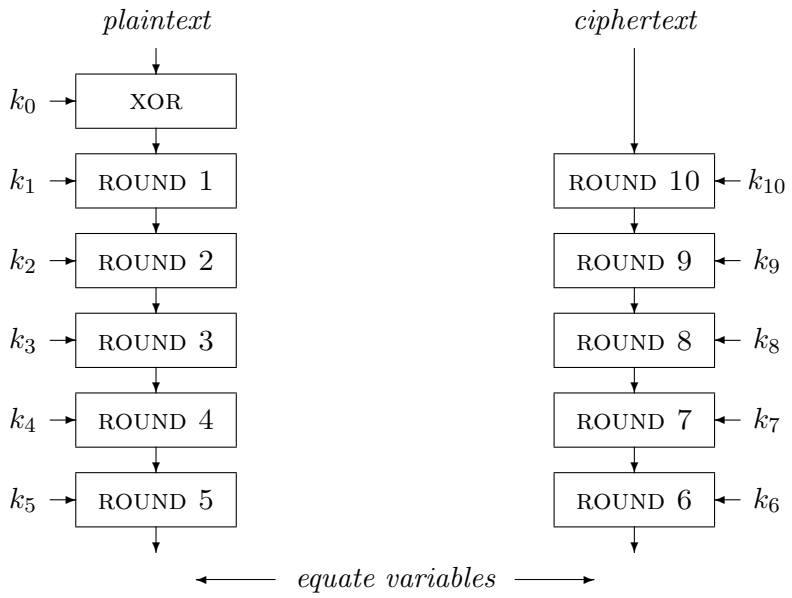

Figure 2.1: An illustration of the meet–in–the–middle technique for the AES-128 [30].

problem instance is hard. An introduction to SAT solvers from a cryptographic perspective can be found in [9].

Meet–in–the–middle Techniques

The iterative nature of modern block ciphers means that the associated systems of equations are typically structured in blocks, with each block containing the equations for one round. Variables in one block only occur in neighbouring blocks or within the relevant part of the key schedule.

A promising technique to find the solution for systems with such structure is to employ a *meet-in-the-middle* approach [29, 30]. The system consisting of r blocks (i.e. rounds) is divided into two subsystems for $\frac{r}{2}$ rounds[1]. We regard the output variables of the first equation subsystem as the input variables of the second equation subsystem. We can then compute the Gröbner bases of the two corresponding subsystems, using an appropriate elimination ordering. We then eliminate variables that do not appear in rounds $\frac{r}{2}$ and $\frac{r}{2}+1$. This gives two small systems of equations in variables from the two systems that are simply related by the round keys. These two equation systems can then be combined with some additional equations from the key schedule and solved to obtain the key. (Figure 2.1)

Experimental results using this approach on AES variants were presented in [29] and seem to confirm that a meet–in–the–middle technique may be more efficient than directly solving the full system of equations arising from a block cipher.

One possible drawback to this approach is that computations using elimination orderings (such as lexicographical) are known to be less efficient than those with degree orderings, and we might expect that using the lexicographical ordering in both subsystems would give

[1] We assume without loss of generality that r is even.

only limited advantages over using the degree reverse lexicographical ordering for the full system. An alternative approach would be to simply compute the Gröbner bases for the two subsystems using the most efficient ordering and then to combine both results to compute the solution of the full set equations. Some experimental results on this approach presented in [29] indicate that this approach can in fact be more efficient for larger examples of the small scale variants of the AES. This suggests the applicability of a more general *divide–and–conquer* approach to the problem of solving the equation system deriving from iterated block ciphers. This was pursued in [2], where a technique known as *Gröbner Surfing* was introduced.

Gröbner Surfing

Gröbner Surfing is the name of a technique proposed in [2] for incrementally computing a Gröbner basis of a key-recovery ideal. This technique works round by round: let $\mathcal{P}_r$ be the equations for the r-th round of the cipher and $\mathtt{GB}$ the Gröbner basis algorithm with the plaintext variables fixed in the first round and the ciphertext variables fixed in the last, the N_r-th round. The idea then is to decompose the Gröbner basis computation as follows:

$$\mathtt{GB}\left(\bigcup_{r=1}^{N_r} \mathcal{P}_r\right) = \mathtt{GB}(\mathcal{P}_{N_r} \cup \mathtt{GB}(\mathcal{P}_{N_{r-1}} \cup (\ldots \cup \mathtt{GB}(\mathcal{P}_1))).$$

Alternatively this method may be expressed as a selection strategy for the critical pairs in the Gröbner basis algorithm. For this method to succeed more efficiently than a direct computation of a Gröbner basis, a suitable term ordering is crucial. Besides lexicographical orderings, block orderings with graded term orderings inside the blocks and block splits at the round or layer boundaries are a suitable choice. Especially in combination with the meet-in-the-middle strategy, Gröbner surfing seems to be an interesting approach that leads to asymptotically improved performance at least in some cases.

Experimental results on the Gröbner Surfing technique applied to small instances of CTC are presented in [2]. It was found that a Gröbner Surfing strategy on CTC performed better than a straightforward degrevlex Gröbner basis computation.

2.1.5 Complexity Bounds

We state some results on the complexity of some the algorithms introduced in this chapter; these are taken from [50, 78], which focus on the XL algorithm, and from [13], which focus on F_5. In both cases, the concepts of Hilbert Theory are behind the scenes, and it is important to understand it to properly analyse these algorithms. In the case of XL, the point is to find the degree D such that an univariate equation in the last variable X_1 can be found after the Gaussian elimination process.

For quadratic polynomials, we have the following complexity result: when the number of polynomials is $m = n + c$, then the minimum degree for XL to succeed is

$$D \geq \frac{n}{\sqrt{c-1}+1}.$$

Since the number of monomials is $\binom{n}{D}$ in the binary case, and $\binom{n+D}{D}$ in the general case, this theorem implies that XL has an exponential complexity.

In [13], we are presented with the following result for F_5 for quadratic polynomials over GF(2): for n equations in n variables (without counting the field equations), the degree for

Cipher	Variables	Linear equations	Quadratic equations	D	Matrix size
Khazad	6464	1664	6000	379	2^{2076}
Misty1	3856	2008	1848	179	2^{1040}
Kasumi	4264	2264	2000	193	2^{1129}
Camelia-128	3584	1920	4304	78	2^{538}
Rijndael-128	3296	1696	4600	69	2^{479}
Serpent-128	16640	8320	9360	703	2^{4196}

Table 2.1: The degree D for the systems of equations constructed in [14]

which F_5 stops is approximately $D \approx 0.09n$, the approximation being valid even for small n. This also implies exponential complexity for F_5.

For general systems (over GF(2)) the results are the following [13]: when m grows linearly with n, the size of the matrix is exponential in n, and the complexity of F_5 is exponential; when n/m tends to zero, F_5 is subexponential; and when m grows as Nn^2, F_5 has polynomial complexity, with exponent depending on N.

Application to Cryptology

The results above indicate that, being of exponential nature, the algorithms introduced earlier should be of very limited use in cryptology, given the large sizes involved. It is known however that F_5 was used successfully for solving the HFE challenge I [55]. In fact, this experiment has also been reproduced with an independent implementation of F_4 [75], and it now takes a few hours to break HFE challenge I with the Magma software [17] on a workstation.

The reason is that the theorems above hold for *generic* systems or *regular sequences*, which are basically systems with no particular properties. In the case on finite fields, random systems take the role of generic systems. In mathematical terms, they are *semi-regular* sequences, which are conjectured to be the general case. It turns out however that the HFE systems of equations are not random-like [34, 55], and it appears that F_5 (and also F_4 as reported in [75]) is sensitive to this fact, i.e. it is a distinguisher for HFE systems. More precisely, the degree for which the matrix M_{F_5} has large enough rank is much lower than the generic bound. From the implementation of XL made in [8], it seems that the XL algorithm is not sensitive to this fact.

The question of the random behaviour of the underlying algebraic system also arises for LFSR-based stream ciphers. A first answer has been given by Rønjom and Helleseth [73, 72], who proved that the coefficients of the monomials involved in the equations representing a filter generator satisfy a particular recurring relation. This property leads to an attack whose complexity roughly corresponds to the linear complexity of the generator. Determining whether a similar property can be exhibited for the system derived from some annihilators of the filtering function remains open.

For the case of block ciphers, although no practical attack has ever been reported, it appears they also give rise to very structured systems. Table 2.1 is an extension (from [12]) of the table given in [14], where systems of quadratic equations have been constructed for various ciphers; in this case the *expected* degree reached by the F_5 algorithm and the size of the matrices have been added. We can note however that the expected degrees are quite large and that the matrices should be in principle too large to be tractable.

One should bear in mind however that the sizes given in Table 2.1 are the ones that would be reached if these systems were generic. It may well be that the systems are non generic, in which case F_5 may succeed with a lower D and smaller matrices. With the current state of knowledge, only practical experiments could tell how these systems behave.

Chapter 3

Algebraic Attacks against Block Ciphers

Algebraic cryptanalysis exploits the intrinsic algebraic structure of a block cipher. In its most common form, the attacker expresses the encryption transformation as a large set of multivariate polynomial equations, and subsequently attempts to solve the system to recover information about the encryption key.

Algebraic cryptanalysis of block ciphers has been the source of much speculation, particularly after being proposed against high profile ciphers such as AES and Serpent [44, 43]. Although the proposed method of attack (XSL) itself is now widely recognized to be incorrect, its publication can be considered as a key event that fueled the interest of the cryptographic community in algebraic methods in block cipher cryptanalysis.

3.1 Polynomial Descriptions of Block Ciphers

In theory, most block ciphers afford a polynomial representation of the encryption process; representing the encryption function as a vector of high-degree polynomials is clearly usually possible. The evaluation of such a vector with a fixed plaintext and key then yields the ciphertext.

In such a case, the structure (i.e. the terms occurring) of the polynomials is known; merely the coefficients are unknown. This leads to interpolation attacks [57], where the Lagrange Interpolation Formula is used to express the encryption transformation as a polynomial function. This function can then be used to encrypt any plaintext without knowledge of the secret key (while the inverse function, similarly constructed, can be used to decrypt ciphertexts without key). However, we see this attack as effective only if the number of coefficients to be interpolated is substantially smaller than the number of entries in the code book (that is, the degree of the polynomial is reasonably small).

Instead of attempting to describe the cipher as a single, high degree polynomial, perhaps a more promising approach is to express the encryption operation as a *system* of polynomial equations. Modelling the encryption process of a block cipher as a polynomial system presents several obvious questions; for instance, should the encryption transformation be represented by a system in few variables but with polynomials of high degree or rather have more variables but equations of lower degree? Since block ciphers can always be seen as vectorial Boolean functions, they can be described as a polynomial system over GF(2). We note however that

certain components are hard or almost impossible to express as "compact" polynomials.

In practice, systems of equations for block ciphers are either written over GF(2) or over GF(2^s) , where s is usually the bit width of the S-Box or a common divisor of the bit widths of all S-Boxes used in the cipher. To keep the degree of the polynomials low, the encryption process is modelled by considering each layer (i.e. the linear and non-linear steps of each encryption round) at a time.

We note that while there are cases where key-schedule equations can be omitted, e.g. if round keys are generated by a selection of bits of the cipher, more commonly the key schedule has a similar structure to the encryption and therefore this set can also be divided into linear and non-linear equations.

We distinguish between two different cases of equations that can be written for the substitution layers: *explicit equations* are equations that express the output variables of a layer *explicitly* as function of the input and key variables, as

$$X_{\ell+1,i} = f_{\ell,i}(X_{\ell,1}, \ldots, X_{\ell,n}, K_{\ell,1}, \ldots, K_{\ell,j}).$$

Implicit equations on the other hand relate the input, output and key variables such that the equations contain non-linear terms involving the output variables, as for example

$$g_{\ell,i}(X_{\ell+1,1}, \ldots, X_{\ell+1,n}, X_{\ell,1}, \ldots, X_{\ell,n}, K_{\ell,1}, \ldots, K_{\ell,j}) = 0.$$

We note that linear layers can often be merged with neighbouring substitution layers without changing the degree of the system. In this case however, the sparsity of the resulting equations will be reduced.

Let then $I \trianglelefteq \mathbb{F}_q[\mathcal{X}]$ be the ideal associated with the encryption process of a block cipher. The hope is thus that one can compute the Gröbner basis G of I to recover the encryption secret key .

3.1.1 Polynomial Systems over GF(2)

Modern block ciphers are designed to be either implemented in hardware or to be executed on computers. Henceforth choosing GF(2) as ground field comes as a natural choice. Essentially the polynomial system is a decomposition of a Boolean circuit implementing the block cipher. For ciphers using S-Boxes over different fields of characteristic two, such as the MISTY family of ciphers, or ciphers with contracting or expanding S-Boxes, e.g. DES, writing equations over GF(2) is the only obvious choice to obtain a polynomial system describing the complete cipher.

Polynomial systems over GF(2) for a number of widely-used block ciphers have appeared in the literature. Systems of quadratic equations for Rijndael and Serpent were initially presented in [43] (for a more thorough analysis of the systems arising from the AES, see [30]). Systems of quadratic equations over GF(2) for other block ciphers, such as Khazad, MISTY1, Kasumi and Camellia-128 together with explicit counts of the number of variables, equations and terms, were given in [14]. These are usually very large, often sparse systems, with over a thousand of variables and equations (for example, the AES with 128-bit keys can be expressed as a system with 9600 equations, of which 1600 are field equations [30]).

3.1.2 Equations for non-linear Components

The most involved part of generating a polynomial system for a block cipher is finding a suitable polynomial representation of the S-Boxes. If the S-Boxes are given in the form of

look-up tables, we can simply interpolate to obtain the corresponding polynomials. Explicit equations over GF(2) can be calculated directly if polynomial representations of the S-Boxes are known over one or more extensions fields of GF(2), such as for the AES. More generally, a generating set of implicit equations of maximum degree d for an s-bit S-Box can be found by triangulating a $t_d \times 2^s$ matrix where $t_d = \sum_{i=0}^{d} \binom{2^s}{i}$ is the total number of terms.

3.2 Developments and Experimental Results

In recent years there has been much research activity in the field algebraic cryptanalysis of block ciphers. Research has mostly concentrated in two aspects: proposing "good" polynomial representations of block ciphers, and proposing dedicated methods of solution for such systems[1]. However there has not been much progress in assessing whether proposed methods can be effective against block ciphers in general. The main reason seems to be that the size of systems arising from block ciphers are completely out of reach for the current computational power. For most methods of cryptanalysis it is quite straightforward to perform experiments on reduced versions of the cipher to understand how the attack might perform. However this has not been the case for algebraic attacks on block ciphers.

One possible approach is to work on small scale variants of block ciphers, in order to test the effectiveness of the main algorithms in solving the systems of algebraic equations. While it is clearly not an easy task to design small versions that can replicate the main cryptographic and algebraic properties of a particular cipher, the hope is however that experiments on small versions can provide a preliminary insight into the behaviour of algebraic cryptanalysis on block ciphers.

3.2.1 Small Scale Polynomial Systems

Since it is often an open research problem by itself to estimate the complexity of many algorithms for solving polynomial systems, experimental evidence has to be considered to evaluate the performance of a given algorithm. However, as noted above, the equation systems for full scale encryption algorithms are usually too complicated or too large for current algorithms. Therefore, usually small scale variants are considered.

One strategy is to consider round reduced variants. Since most modern block ciphers – including the AES and the DES – iterate the same operations N_r times, it is straightforward to consider a related cipher with fewer rounds. Although the reduced cipher has weaker security, experiments with such reduced-round version might still provide an insight into the behaviour of the full-round version of the algorithm. Some researchers have adopted this approach with DES [38, 68], performing experiments with reduced round versions of the algorithm (up to six rounds).

There are however algorithms that such a strategy may not be the most efficient form of experiment. For instance, systems of equations for one (out of ten) round of AES already can be quite large. An alternative strategy followed in [29] was to propose small scale variants of the AES. These systems aim to remodel the algebraic structure of the AES and are denoted as $SR(n, r, c, e)$, where n is the number of rounds, r (c) the number of rows (columns) in the AES state space and e the size of the words in the state space. Thus, $SR(10, 4, 4, 8)$ is a $4 \cdot 4 \cdot 8 = 128$ bit block cipher with 10 rounds, which is essentially equivalent to the full

[1] For AES-specific analysis, refer to the ECRYPT Report [67].

AES-128. Table 3.1 compiles reported timings against several small scale variants of AES and DES.

Cipher	Method of Solution	System	Time
SR(4,1,1,4)	MRHS [69]	PC (1 GB)	0.032 s
SR(10,1,1,4)	MRHS [69]	PC (1 GB)	0.32 s
SR(10,1,1,4)	PolyBoRi [19]	Opteron 2.2 GHZ (16 GB)	0.14 s
SR(10,1,2,4)	PolyBoRi [19]	Opteron 2.2 GHZ (16 GB)	6.7 s
4r DES	ElimLin [38]	Centrino 1.6 GHZ	$2^{19} \cdot 8$ s
5r DES	ElimLin [38]	Centrino 1.6 GHZ	$2^{23} \cdot 173$ s
6r DES	MiniSat [38]	Centrino 1.6 GHZ	$2^{23} \cdot 173$ s

Table 3.1: Reported runtimes of various algorithms against reduced-round ciphers.

3.2.2 Direct Construction of Gröbner Bases for Block Ciphers

For some block ciphers, a zero-dimensional Gröbner basis for the key-recovery ideal can surprisingly be constructed with minimal computational effort, namely without performing any polynomial reductions. This seems to be the case for most block ciphers; it was shown for the AES in [22] as well as for ciphers of the Flurry and Curry families in [23].

The key idea behind this method is to construct a polynomial system in which all leading terms are pairwise prime. If this condition is fulfilled, it follows from the *first Buchberger criterion* [20] that the resulting set of polynomials forms a Gröbner basis. The Gröbner basis is constructed from a direct description of the block cipher and its key schedule over GF(2^n) merely by linearly combining polynomials and choosing an appropriate graded term ordering.

Yet despite the fact that the basis given in [22] spans a zero-dimensional ideal (i.e. it has a finite solution set), the set is much larger than the one cryptanalyst is interested at. Indeed, since the system does not include the field equations, it also contains solutions in the algebraic closure K of k, and as a consequence is too large to allow exhaustive search for the solution. Furthermore, even though there exist algorithms to convert a Gröbner basis with respect to one monomial ordering to another Gröbner basis with respect to another monomial ordering (which could yield the solution sought), these algorithms have a complexity worse than exhaustive key search for the basis considered in [22]. Overall, no technique is thus far known for exploiting the construction of this Gröbner basis for the AES-128 for the cryptanalysis of the cipher.

3.3 Future Research Directions

Despite much research in algebraic cryptanalysis of block ciphers, proposed methods have had so far limited success in targeting modern block ciphers. In fact, there is no modern block cipher, with practical relevance, that has been successfully attacked using algebraic cryptanalysis faster than with other techniques.

A recent and promising trend in block cipher cryptanalysis is to combine algebraic approaches with traditional methods of cryptanalysis. In [3] an attack is proposed that combines algebraic techniques with differential cryptanalysis. In differential cryptanalysis, given a *differential characteristic* covering r out of N_r rounds of a given block cipher, the cryptanalyst

usually guesses subkey bits to overcome the last $r_d = N_r - r$ rounds. This quickly becomes impractical as the number of rounds r_d grows. In [3] the authors investigate the block cipher PRESENT [15], and are able to increase r_d from 2 to 4 rounds by using algebraic techniques. Specifically, the authors construct a system of polynomial equations for r_d rounds for pairs of plaintexts, and use Gröbner basis algorithms to perform a consistency check, and as a result determine whether a given pair satisfies the considered differential characteristic. Based on this observation, information about the encryption key could be recovered. The technique is in theory generally applicable to improve differential cryptanalysis although no experimental evidence of the feasibility of the attack against reduced versions of ciphers other than PRESENT are provided.

Chapter 4

Algebraic Attacks against Stream Ciphers

In contrast to block ciphers, algebraic attacks have been successfully used in the analysis of several LFSR-based stream ciphers. As previously discussed, algebraic attacks on stream ciphers exploit the fact that each new bit of the cipher output (keystream) gives rise to a new equation on the initial state. The cryptanalyst can collect a large number of bits from the keystream to construct a system of equations, which can then be solved. Algebraic attacks as a method for stream cipher cryptanalysis were originally introduced by Courtois and Meier in [40], and generally apply to LFSR-based stream ciphers using non-linear Boolean functions as combiner or filter.

4.1 Attack Principles

Let $(k_0, k_1, \ldots, k_{\ell-1})$ be the cipher initial state and $x_t = (x_0^t, x_1^t, \ldots, x_{\ell-1}^t)$ the state at time $t > 0$. If f is the combining function of degree d (potentially defined on a subset of x_t) and b_t is the cipher output at time t, then we have $b_t = f(x_0^t, x_1^t, \ldots, x_{\ell-1}^t)$. If we assume that L is the cipher linear recursion function, then the cryptanalyst can collect the following system of equations

$$\begin{aligned} b_0 &= f(k_0, k_1, \ldots, k_{\ell-1}), \\ b_1 &= f(L(k_0, k_1, \ldots, k_{\ell-1})), \\ b_2 &= f(L^2(k_0, k_1, \ldots, k_{\ell-1})), \\ &\ldots \\ b_N &= f(L^N(k_0, k_1, \ldots, k_{\ell-1})). \end{aligned}$$

The problem for the cryptanalyst is, given a number of output bits $b_0, b_1, \ldots, b_N$, to recover the initial state $(k_0, k_1, \ldots, k_{\ell-1})$.

Assuming that the cryptanalyst can obtain enough output bits to construct a large enough system (such that it has unique solution), the above system can be solved with one of the general techniques discussed in Chapter 2. The simplest one, called *linearization*, consists in identifying the original system with a new linear system of $\sum_{i=0}^{d} \binom{\ell}{i}$ variables, where each monomial is considered as a new variable. The entire initial state is then recovered by a

Gaussian reduction (or by more sophisticated techniques) whose time complexity is roughly

$$\left(\sum_{i=0}^{d}\binom{\ell}{i}\right)^{\omega} \simeq \ell^{\omega d} ,$$

where ω is the exponent of the matrix inversion algorithm, i.e., $\omega \simeq 2.37$ [32]. Thus the complexity of the attack is *polynomial* in the state (and key) size, but *exponential* in the degree of the Boolean function f.

There are however two noteworthy remarks regarding the estimate of the attack complexity. First, algebraic attacks have usually been applied against filter generators, and it is not clear whether the complexity may be lower for combining generators due to the potential special structure of the algebraic system (the internal state of LFSR i at time t only depends on the initial state of this LFSR and not on all bits of the initial states). Second, the estimate for solving the algebraic system also assumes that the system has a random behaviour. However it was recently shown that this is not true in some particular cases [73, 72, 71, 70].

Overall, assuming that d (and ℓ) were large enough such that complexity estimate of $\mathcal{O}(\ell^{\omega d})$ was above the complexity of exhaustive search, it could be claimed that the cipher was immune to this first, naive attempt of mounting an algebraic attack (although other methods discussed in Chapter 2 could potentially be successfully used to solve the system). However Courtois and Meier showed in [40] that one could reduce the complexity of the attack by exploiting some properties of the function f.

4.2 Algebraic Immunity of Boolean Functions

It was shown in [40] that several classes of LFSR-based stream ciphers are vulnerable to algebraic attacks if there exist relations of low degree between the output and the inputs of the associated Boolean function f. Such relations correspond to low degree multiples of f, i.e., to relations $g(x)f(x) = h(x)$ for some function g, where h has a low degree. It was further proved in [54, 62] that, in the case of algebraic attacks over $\mathbb{F}_2$, the existence of any such relation is equivalent to the existence of a low degree function in the annihilator ideal of either f or $(1 + f)$. Indeed, if $g(x)f(x) = h(x)$ with $\deg(h) \leq d$, we obtain, by multiplying this equation by $f(x)$, that

$$g(x)\left[f(x)\right]^2 = h(x)f(x) = g(x)f(x) = h(x) ,$$

leading to $h(x)\left[1 + f(x)\right] = 0$.

Suppose then that the keystream bit b_t is obtained by applying f to the current internal state of the generator, $b_t = f(x_t)$. Algebraic attacks exploit the following relations:

- if $b_t = 1$, any function g of degree at most d in the annihilator ideal of f, $AN(f) = \{g \mid g(x)f(x) = 0,\ \forall x\}$ leads to $g(x_t) = 0$;
- if $b_t = 0$, any function g' of degree at most d in $AN(1 + f)$ leads to $g'(x_t) = 0$.

The cryptographic relevance of the algebraic immunity. The relevant parameter in the context of algebraic attacks, called the *algebraic immunity* of the Boolean function, $AI(f)$, is the lowest degree achieved by a function in $AN(f) \cup AN(1 + f)$. A simple combinatorial argument implies that, for any Boolean function f of n variables, $AI(f) \leq \lceil n/2 \rceil$. We deduce

from this bound that if an n-variable function is used in the generator, the complexity of the algebraic attack will be at most $\ell^{\frac{\omega n}{2}}$, where ℓ is the size of the internal state. If we suppose that the size of the internal state is minimal with respect to key-size k, i.e. that $\ell = 2k$ (it is known that the size of the internal state must be at least twice the key size in order to resist time-memory trade-off attacks), then, for the cipher to be resistant to algebraic attacks, we must have $(2k)^{\frac{\omega n}{2}} \geq 2^k$, i.e.,

$$n \geq 0.84 \left[\frac{k}{1 + \log_2(k)} \right] .$$

For instance, a filter generator with a 128-bit key and a 256-bit LFSR must use a filtering function of at least 16 variables. Note that the recommended number of variables is probably higher than the previous bound because more efficient techniques may be used for solving the algebraic system (see Chapter 2).

Properties of the annihilator ideal of a Boolean function. The set $AN(f)$ of all annihilating functions of f is obviously an ideal in the ring of all Boolean functions, and it is generated by $(1 + f)$. It consists of the $2^{2^n - wt(f)}$ functions of n variables which vanish on the support of f, i.e., on all x such that $f(x) = 1$, where $wt(f)$ denotes the size of the support of f. It is important to note that the number of functions with a given degree in $AN(f) \cup AN(1+f)$ is less important from a cryptanalytic point of view than the algebraic immunity: the number of such annihilating functions only influences the number of keystream bits required for the attack, and not the time-complexity (except maybe for some refinements).

The number of functions of degree at most d in $AN(f)$ is equal to 2^κ where κ is the dimension of the kernel of the matrix obtained by restricting the Reed-Muller code of length 2^n and order d to the support of f. In other words, the rows of this matrix correspond to the evaluations of the monomials of degree at most d on $\{x, |f(x) = 1\}$. Since this matrix has $\sum_{i=0}^{d} \binom{n}{i}$ rows and $wt(f)$ columns, its kernel is non-trivial when

$$\sum_{i=0}^{d} \binom{n}{i} > wt(f) .$$

Similarly, $AN(1 + f)$ contains some functions of degree d or less if

$$\sum_{i=0}^{d} \binom{n}{i} > 2^n - wt(f) .$$

This shows, as pointed out in [46], that the algebraic immunity of an n-variable function is related to its Hamming weight. Most notably, for odd n, only balanced functions can have optimal algebraic immunity. For even n, the Hamming weight of a function with optimal algebraic immunity must satisfy

$$\sum_{i=0}^{\frac{n}{2}-1} \binom{n}{i} \leq wt(f) \leq \sum_{i=0}^{\frac{n}{2}} \binom{n}{i}.$$

Algebraic immunity and other cryptographic criteria Besides the Hamming weight of the function, its nonlinearity is also related to its algebraic immunity [46]. It can be shown

that, for any linear function φ, the algebraic immunity of $f+\varphi$ is at most $AI(f)+1$. Therefore, any function f of n variables with algebraic immunity at least d satisfies

$$\mathcal{NL}(f) \geq \sum_{i=0}^{d-2} \binom{n}{i} .$$

It follows that any function with optimal algebraic immunity has a high nonlinearity, more precisely

$$\mathcal{NL}(f) \geq \begin{cases} 2^{n-1} - \binom{n}{\frac{n-1}{2}} & \text{if } n \text{ is odd,} \\ 2^{n-1} - \frac{1}{2}\binom{n}{\frac{n}{2}} - \binom{n}{\frac{n}{2}-1} & \text{if } n \text{ is even.} \end{cases}$$

A high nonlinearity and a high algebraic immunity are thus compatible criteria. Another important consequence is that the nonlinearity of a function may be sufficient criteria to decide whether it has low algebraic immunity (although the converse is not true).

Another cryptographic property that implies that a function does not have maximal algebraic immunity is the notion of *normality*. A function is said to be k-normal (resp. k-weakly normal) if there exists an affine subspace of dimension k on which the function is constant (resp. affine). Since the minimum weight codewords of $RM(r,n)$ are those whose support is an affine subspace of dimension $n-r$, we deduce that any k-normal function f of n variables has algebraic immunity at most $n-k$. Similarly, any k-weakly normal function has algebraic immunity at most $n-k+1$. Non-normal (and non-weakly normal) functions such as the functions exhibited in [25] may be good candidates if we want to construct functions with optimal nonlinearity.

The algebraic immunity is also related to some other quantities as the nonlinearity profile of the function, which corresponds to its distance to the all functions of degree at most d (i.e., its distance to $R(d,n)$) when d varies [61, 26, 63]. However, the existence of links between algebraic immunity and other cryptographic criteria remains unknown. Correlation-immunity does not seem to be a priori incompatible with optimal algebraic immunity; there exists a 1-resilient function of 5 variables with optimal algebraic immunity. However, the link with other known criteria must be investigated further.

Computing the algebraic immunity of a Boolean function. The basic algorithm for computing the algebraic immunity of an n-variable function consists in performing a Gaussian elimination on the generator matrix of the punctured $RM(\lfloor\frac{n-1}{2}\rfloor,n)$ restricted to the support of f. This matrix has $k(\lfloor\frac{n-1}{2}\rfloor,n) = \sum_{i=0}^{\lfloor\frac{n-1}{2}\rfloor}\binom{n}{i}$ rows and $wt(f)$ columns. Therefore, the algorithm requires $k^2(\lfloor\frac{n-1}{2}\rfloor,n)wt(f)$ operations, which is close to 2^{3n-3} when f is balanced. As noted in [62], the complexity can be significantly reduced if we only want to check whether a function has annihilators of small degree d, since we do not need to consider all positions in the support of f. Indeed, considering a number of columns which is only slightly higher that the code dimension $k(d,n)$ is usually sufficient for proving that a function does not admit any annihilator of degree d. The complexity can then be reduced to $\mathcal{O}(k^3(d,n))$.

More advanced techniques for computing the algebraic immunity of Boolean functions were proposed in [49, 6, 48]. As a comparison, finding an annihilator of degree 8 for functions of 17 variables requires some hours with the naive algorithm using a Gaussian elimination. The algorithm in [6] was used for finding annihilators of degree 9 for functions of 19 variables. On the other hand, the algorithm in [48] is able to find annihilators of degree 11 for 23-variable functions within 11 hours.

We note that more efficient techniques apply if we only need to check that a function does not admit any annihilator of degree less than or equal to d. An recursive algorithm proposed in [49] in this context has complexity $\mathcal{O}(d)$, which is the lowest complexity that can be achieved. For instance, it requires 30 seconds for checking whether a 128-variable function has some annihilators of degree less than or equal to 3.

Resistance to fast algebraic attacks and other criteria. At CRYPTO 2003, Courtois presented some important improvements on algebraic attacks, called *fast algebraic attacks* [33]. The refinement first relies on the existence of some low degree relations between the bits of the initial state and not only one but several consecutive keystream bits. In other words, the attacker wants to find some low degree relations g between the inputs and outputs of the function

$$\begin{aligned} F_m\colon \mathbb{F}_2^\ell &\to \mathbb{F}_2^m \\ x &\mapsto ((f(x), f(L(x)), \ldots, f(L^{m-1}(x)), \end{aligned}$$

where L is the linear transition function for the internal state. This function is very similar to the so-called *augmented function* defined in [4], and the fact that it may be much weaker than the filtering function had been pointed out in [4] in the context of (fast) correlation attacks. However, the complexity required for computing the low degree relations between the n inputs and m outputs of F_m increases with m. The direct algorithm (used for multi-output functions) can only be used for small m. It is an open problem to determine whether there exist relationships between the algebraic immunity of f and the algebraic immunity of F_m. The same problem arises for other cryptographic criteria such as correlation immunity.

Since the computation of low degree relations involving several keystream bits is usually infeasible, Courtois proposed to focus on particular subclasses of relations that can be obtained much faster. The relations considered in the attack are given by linear combinations of relations of the form

$$g(x_0, \ldots, x_{\ell-1}, b_t, \ldots, b_{t+m-1}),$$

where the terms of highest degree do not involve any keystream bits. Then, an additional precomputation step consists in determining the linear combinations of the previous relations which cancel out the highest degree monomials. Some algorithms for this step have been proposed in [33, 5]. This technique helps to decrease the degree of the relations used in the attack for different practical examples.

4.3 Algebraic Attacks: Constructions and Examples

As discussed early in this chapter, algebraic attacks appear to be particularly suitable against LFSR-based ciphers, featuring linear updating of the cipher state and using a filtering Boolean function to derive the output bits. Indeed, the first instance of the attack was proposed against the cipher Toyocrypt, which presents this structure [40]. The attack was also used against LILI-128, which is a cipher featuring an irregular clocking mechanism (although the effect of the irregular clocking on the attack was eliminated by means of guessing state bits). The attack technique was then soon generalised for stream ciphers using combiners with memory, and applied to the Bluetooth generator E0 [7]. Other ciphers affected include SOBER-t32/SOBER-t16 [27] and Sfinks [36]; algebraic attacks have been more recently applied against reasonably high-profile ciphers such as the KeeLoq cipher [10] and CRYPTO1 stream cipher [41].

Much research in the area of algebraic cryptanalysis of stream ciphers has in fact concentrated on the study of properties of Boolean functions, and their resistance to algebraic attacks (Section 4.2). However some attempts have been made to extend algebraic attacks to other stream cipher constructions [47, 24, 51]. It is expected that further research effort will be dedicated to investigate the applicability of algebraic attacks against other classes of stream ciphers.

Chapter 5

Algebraic Cryptanalysis: Research Directions

Algebraic cryptanalysis against symmetric primitives has recently received much attention from the cryptographic community. From a reasonably effective technique against some stream cipher constructions, to the source of much speculation in the case of block ciphers, this is currently a very active area of research.

Current focus of attention includes the extension of the attack techniques to other stream cipher constructions (besides LFSR-based combining generators), and the recent trend of combining algebraic approaches with traditional methods of cryptanalysis, as illustrated on the attacks against KeeLoq [10] (where an algebraic attack is combined with a slide attack), and in [3], where an attack against reduced-versions of the block cipher PRESENT is proposed that combines algebraic techniques with differential cryptanalysis.

Finally, algebraic techniques have thus far been relatively unexplored for hash function cryptanalysis. Although Gröbner basis techniques have been proposed to improve existing attacks against hash functions [77], and SAT-solvers have been shown to be useful for similar analysis of SHA-1 [65], there has been limited work on algebraic attacks against hash functions. We expect however that algebraic techniques are likely to equally important in the analysis of hash functions, particularly during the upcoming SHA-3 competition.

Bibliography

[1] S.B. Akers. Binary decision diagrams. *IEEE Transactions on Computers*, 27(6):509–516, June 1978.

[2] M. Albrecht. Algebraic Attacks on the Courtois Toy Cipher. *Cryptologia*, 32(3):220–276, July 2008.

[3] M. Albrecht and C. Cid. Algebraic Techniques in Differential Cryptanalysis, 2008.

[4] R. J. Anderson. Searching for the optimum correlation attack. In *Fast Software Encryption - FSE'94*, volume 1008 of *Lecture Notes in Computer Science*, pages 137–143. Springer-Verlag, 1995.

[5] F. Armknecht. Improving fast algebraic attacks. In *Fast Software Encryption - FSE 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 65–82. Springer-Verlag, 2004.

[6] F. Armknecht, C. Carlet, P. Gaborit, S. Künzli, W. Meier, and O. Ruatta. Efficient computation of algebraic immunity for algebraic and fast algebraic attacks. In *Advances in Cryptology - EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 147–164. Springer-Verlag, 2006.

[7] F. Armknecht and M. Krause. Algebraic attacks on combiners with memory. In *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 162–176. Springer-Verlag, 2003.

[8] G. Ars, J.-C. Faugère, H. Imai, M. Kawazoe, and M. Sugita. Comparison between XL and Gröbner basis algorithms. In *Advances in Cryptology - ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*. Springer-Verlag, 2004.

[9] G. V. Bard. *Algorithms for Solving Linear and Polynomial Systems of Equations over Finite Fields with Applications to Cryptanalysis*. PhD thesis, University of Maryland, 2007. available at `http://www.cs.umd.edu/~jkatz/THESES/bard_thesis.pdf`.

[10] G. V. Bard, N. T. Courtois, and D. Wagner. Algebraic and slide attacks on keeloq. In *Fast Software Encryption – FSE 2008*, 2008. To appear.

[11] G.V. Bard, N.T. Courtois, and C. Jefferson. Efficient Methods for Conversion and Solution of Sparse Systems of Low-Degree Multivariate Polynomials over GF(2) via SAT-Solvers. Cryptology ePrint Archive, Report 2007/024, 2007. available at `http://eprint.iacr.org/2007/024`.

[12] M. Bardet. *Étude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie.* PhD thesis, Université Paris VI, 2004.

[13] M. Bardet, J.-C. Faugère, and B. Salvy. Complexity of Gröbner basis computation for semi-regular overdetermined sequences over F_2 with solutions in F_2. Technical Report 5049, INRIA, December 2003. available at `http://www.inria.fr/rrrt/rr-5049.html`.

[14] A. Biryukov and C. De Canniere. Block Ciphers and Systems of Quadratic Equations. In Thomas Johansson, editor, *Fast Software Encryption - FSE 2003*, volume 2887 of *Lecture Notes in Computer Science*, pages 274–289, New York, 2003. Springer Verlag. available at: `http://www.cosic.esat.kuleuven.be/publications/article-14.pdf`.

[15] A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, Matthew Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: An ultra-lightweight block cipher. In *CHES 2007*, volume 7427 of *Lecture Notes in Computer Science*, pages 450–466. Springer Verlag, 2007. available at `http://www.crypto.rub.de/imperia/md/content/texte/publications/conferences/present_ches2007.pdf`.

[16] B. Bollig and I. Wegener. Improving the variable ordering of OBDDs is NP-complete. *IEEE Trans. Computers*, 45(9):993–1002, 1996.

[17] W. Bosma, J. Cannon, and C. Playoust. The MAGMA Algebra System I: The User Language. In *Journal of Symbolic Computation 24*, pages 235–265. Academic Press, 1997.

[18] M. Brickenstein. Slimgb: Gröbner Bases with Slim Polynomials. In *Reports On Computer Algebra 35*. Centre for Computer Algebra, University of Kaiserslautern, 2005. available at: `http://www.mathematik.uni-kl.de/~zca/Reports_on_ca/35/paper_35_full.ps.gz`.

[19] M. Brickenstein and A. Dreyer. PolyBoRi: A framework for Gröbner basis computations with Boolean polynomials. In *Electronic Proceedings of MEGA 2007*, 2007. available at `http://www.ricam.oeaw.ac.at/mega2007/electronic/26.pdf`.

[20] B. Buchberger. A criterion for detecting unnecessary reductions in the construction of Groëbner basis. In Edward W. Ng, editor, *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposium on Symbolic and Algebraic Computation*, volume 72 of *Lectures Notes in Computer Science*. Springer Verlag, 1979.

[21] B. Buchberger. Gröebner bases: an algorithmic method in polynomial ideal theory. In N. K. Bose, editor, *Multidimensional Systems Theory*. D. Reidel Publishing Company, 1985.

[22] J. Buchmann, A. Pychkine, and R-P. Weinmann. A Zero-dimensional Gröbner Basis for AES-128. In *Fast Software Encryption 2006*, volume 4047 of *Lecture Notes in Computer Science*, pages 78–88. Springer Verlag, 2006.

[23] J. Buchmann, A. Pyshkin, and R-P. Weinmann. Block ciphers sensitive to Gröbner basis attacks. In David Pointcheval, editor, *Topics in Cryptology – CT-RSA 2006*, volume 3860, pages 313–331, 2006.

[24] C. Charnes C. McDonald and J. Pieprzyk. Attacking bivium with minisat. eSTREAM, ECRYPT Stream Cipher Project, Report 2007/040, 2007. `http://www.ecrypt.eu.org/stream`.

[25] A. Canteaut, M. Daum, H. Dobbertin, and G. Leander. Normal and non normal bent functions. *Discrete Applied Mathematics*, 154(2):202–218, 2006. Special Issue on Coding and Cryptology.

[26] C. Carlet. On the higher-order nonlinearities of algebraic-immune fonctions. In *Advances in Cryptology - CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*. Springer, 2006.

[27] J.Y. Cho and J. Pieprzyk. Algebraic attacks on sober-t32 and sober-t16 without stuttering. In Bimal K. Roy and Willi Meier, editors, *FSE*, volume 3017 of *Lecture Notes in Computer Science*, pages 49–64. Springer, 2004.

[28] C. Cid and G. Leurent. An Analysis of the XSL Algorithm. In *Advances in Cryptology — ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 333–352. Springer Verlag, 2005.

[29] C. Cid, S. Murphy, and M. Robshaw. Small Scale Variants of the AES. In *Fast Software Encryption 2005*, volume 3557 of *Lecture Notes in Computer Science*, pages 145–162. Springer Verlag, 2005. available at `http://www.isg.rhul.ac.uk/~sean/smallAES-fse05.pdf`.

[30] C. Cid, S. Murphy, and M. Robshaw. *Algebraic Aspects of the Advanced Encryption Standard*. Springer Verlag, 2006.

[31] C. Cid and R-P. Weinmann. Block Ciphers: Algebraic Cryptanalysis and Gröbner Bases. *Groebner Codes Cryptography*, page To appear.

[32] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic programming. *Journal of Symbolic Computation*, (9):251–280, 1990.

[33] N. Courtois. Fast algebraic attacks on stream ciphers with linear feedback. In *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 176–194. Springer-Verlag, 2003.

[34] N. T. Courtois. The security of hidden field equations (HFE). In David Naccache, editor, *Progress in Cryptology - CT-RSA 2001: The Cryptographers' Track at RSA Conference 2001*, volume 2020 of *Lecture Notes in Computer Science*. Springer Verlag, 2001.

[35] N. T. Courtois. Higher order correlation attacks, XL algorithm and cryptanalysis of Toyocrypt. In P.J. Lee and C.H. Lim, editors, *Information Security and Cryptology - ICISC 2002: 5th International Conference*, volume 2587 of *Lecture Notes in Computer Science*. Springer Verlag, 2003.

[36] N. T. Courtois. Cryptanalysis of sfinks. In Dongho Won and Seungjoo Kim, editors, *ICISC*, volume 3935 of *Lecture Notes in Computer Science*, pages 261–269. Springer, 2005.

[37] N. T. Courtois. How Fast can be Algebraic Attacks on Block Ciphers? Cryptology ePrint Archive, Report 2006/168, 2006. available at: `http://eprint.iacr.org/2006/168.pdf`.

[38] N. T. Courtois and G.V. Bard. Algebraic Cryptanalysis of the Data Encryption Standard. In Steven D. Galbraith, editor, *Cryptography and Coding – 11th IMA International Conference*, volume 4887 of *Lecture Notes in Computer Science*, pages 152–169, Berlin Heidelberg New York, 2007. Springer Verlag. available at `http://eprint.iacr.org/2006/402`.

[39] N. T. Courtois, A. Klimov, J. Patarin, and A. Shamir. Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations. In *Advances in Cryptology — EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 392–407. Springer Verlag, 2000.

[40] N. T. Courtois and W. Meier. Algebraic attacks on stream ciphers with linear feedback. In *Advances in Cryptology — EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 345–359. Springer Verlag, 2003.

[41] N. T. Courtois, K. Nohl, and S. O'Neil. Algebraic attacks on the crypto-1 stream cipher in mifare classic and oyster cards. Cryptology ePrint Archive, Report 2008/166, 2008. `http://eprint.iacr.org/`.

[42] N. T. Courtois and J. Patarin. About the XL Algorithm over GF(2). In Marc Joye, editor, *Topics in Cryptology - CT-RSA 2003: The Cryptographers' Track at the RSA Conference 2003; Proceedings*, volume 2612 of *Lecture Notes in Computer Science*, pages 141–157. Springer Verlag, 2003.

[43] N. T. Courtois and J. Pieprzyk. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. In *Advances in Cryptology — ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 267–287. Springer Verlag, 2002.

[44] N. T. Courtois and J. Pieprzyk. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. Cryptology ePrint Archive, Report 2002/044, 2002. available at `http://eprint.iacr.org/2002/044`.

[45] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms*. Springer Verlag, New York, 2005.

[46] D. K. Dalai, K. C. Gupta, and S. Maitra. Results on algebraic immunity for cryptographically significant boolean functions. In *Progress in Cryptology - Indocrypt 2004*, volume 1880 of *Lecture Notes in Computer Science*, pages 92–106. Springer-Verlag, 2004.

[47] B. Debraize and L. Goubin. Guess-and-determine algebraic attack on the self-shrinking generator. In *Fast Software Encryption – FSE 2008*, 2008. To appear.

[48] F. Didier. Using Wiedemann's algorithm to compute the immunity against algebraic and fast algebraic attacks. In *Progress in cryptology - INDOCRYPT 2006*, volume 4329 of *Lecture Notes in Computer Science*, pages 236–250. Springer, 2006.

[49] F. Didier and J.-P. Tillich. Computing the algebraic immunity efficiently. In *Fast Software Encryption - FSE 2006*, volume 4047 of *Lecture Notes in Computer Science*, pages 359–374. Springer, 2006.

[50] C. Diem. The XL algorithm and a conjecture from commutative algebra. In Pil Jong Lee, editor, *Advances in Cryptology - ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*. Springer Verlag, 2004.

[51] T. Eibach, E. Pilz, and G. Völkel. Attacking bivium using sat solvers. In Hans Kleine Büning and Xishun Zhao, editors, *SAT*, volume 4996 of *Lecture Notes in Computer Science*, pages 63–76. Springer, 2008.

[52] J-C. Faugère. A New Efficient algorithm for Computing Gröbner Basis (F4), 1999. available at `http://modular.ucsd.edu/129-05/refs/faugere_f4.pdf`.

[53] J-C. Faugère. A New Efficient Algorithm for Computing Gröbner Bases without Reduction to Zero (F5). In *Proceedings of ISSAC*, pages 75–83. ACM Press, 2002.

[54] J.-C. Faugère and G. Ars. An algebraic cryptanalysis of nonlinear filter generators using Gröbner bases. Technical Report 4739, INRIA, 2003. Available at `ftp://ftp.inria.fr/INRIA/publication/publi-pdf/RR/RR-4739.pdf`.

[55] J.-C. Faugère and A. Joux. Algebraic cryptanalysis of hidden field equation (HFE) cryptosystems using gröbner bases. In *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*. Springer-Verlag, 2003.

[56] G.-M. Greuel, G. Pfister, and H. Schönemann. Singular 3.0. A Computer Algebra System for Polynomial Computations, Centre for Computer Algebra, University of Kaiserslautern, 2005. available at: `http://www.singular.uni-kl.de`.

[57] T. Jakobsen and L. Knudsen. The interpolation attack on block ciphers. In Eli Biham, editor, *Fast Software Encryption – FSE 1997*, volume 1267, pages 28–40, 1997.

[58] D. Lazard. Gröbner-bases, Gaussian elimination and resolution of systems of algebraic equations. In *Proceedings of the European Computer Algebra Conference on Computer Algebra*, volume 162 of *Lectures Notes in Computer Science*. Springer-Verlag, 1983.

[59] C. Y. Lee. Representation of switching circuits by binary-decision programs. *Bell System Technical Journal*, 38:985–999, July 1959.

[60] C-W. Lim and K. Khoo. Detailed Analysis on XSL applied to BES. In *Fast Software Encryption – FSE 2007*, volume 4593 of *Lecture Notes in Computer Science*, pages 242–253. Springer–Verlag, 2007.

[61] M. Lobanov. Tight bound between nonlinearity and algebraic immunity. Technical Report 2005/441, IACR Preprint, 2005. Available at `http://eprint.iacr.org/2005/441/`.

[62] W. Meier, E. Pasalic, and C. Carlet. Algebraic attacks and decomposition of Boolean functions. In *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 474–491. Springer-Verlag, 2004.

[63] S. Mesnager. Improving the lower bound on the higher order nonlinearity of boolean functions with prescribed algebraic immunity. *IEEE Transactions on Information Theory*, 54(8), 2008.

[64] S-I. Minato. Zero-suppressed BDDs for set manipulation in combinatorial problems. In *DAC*, pages 272–277, 1993.

[65] I. Mironov and L. Zhang. Applications of sat solvers to cryptanalysis of hash functions. In Armin Biere and Carla P. Gomes, editors, *SAT*, volume 4121 of *Lecture Notes in Computer Science*, pages 102–115. Springer, 2006.

[66] S. Murphy and M. Paterson. A geometric view of cryptographic equation solving. Technical report, 2007. available at `http://www.rhul.ac.uk/mathematics/techreports`.

[67] European Network of Excellence in Cryptology. AES Security Report. In C. Cid and H. Gilbert, editors, *ECRYPT Report (D.STVL.2)*, 30 January 2006.

[68] H. Raddum and I. Semaev. New technique for solving sparse equation systems. Cryptology ePrint Archive, Report 2006/475, 2006. available at `http://eprint.iacr.org/2006/475`.

[69] H. Raddum and I. Semaev. Solving MRHS linear equations. Cryptology ePrint Archive, Report 2007/285, 2007. available at `http://eprint.iacr.org/2007/285`.

[70] P. Rizomiliotis. Remarks on the new attack on the filter generator and the role of high order complexity. In *Cryptography and Coding, 11th IMA International Conference*, volume 4887 of *Lecture Notes in Computer Science*, pages 204–219. Springer, 2007.

[71] S. Rønjom, G. Gong, and T. Helleseth. A survey of recent attacks on the filter generator. In *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes- AAECC 2007*, volume 4851 of *Lecture Notes in Computer Science*, pages 7–17. Springer, 2007.

[72] S. Rønjom and T. Helleseth. Attacking the Filter Generator over $GF(2^m)$. In ECRYPT Network of Excellence, editor, *SASC 2007 Workshop record*, 2007.

[73] S. Rønjom and T. Helleseth. A new attack on the filter generator. *IEEE Transactions on Information Theory*, 53(5):1752–1758, May 2007.

[74] C.E. Shannon. Communication Theory of Secrecy Systems. In *Bell System Technical Journal 28*, pages 656–715, 1949.

[75] A. Steel. Allan Steel's Gröbner basis timings page, 2004. available at `http://magma.maths.usyd.edu.au/users/allan/gb/`.

[76] T. Stegers. Faugère's F5 Algorithm Revisited. Master's thesis, Technische Universit"at Darmstadt, 2005.

[77] M. Sugita, M. Kawazoe, L. Perret, and H. Imai. Algebraic cryptanalysis of 58-round sha-1. In Alex Biryukov, editor, *FSE*, volume 4593 of *Lecture Notes in Computer Science*, pages 349–365. Springer, 2007.

[78] B.-Y. Yang and J.-M. Chen. Theoretical analysis of XL over small fields. In Huaxiong Wang, Josef Pieprzyk, and Vijay Varadharajan, editors, *Information Security and Privacy: 9th Australasian Conference, ACISP 2004*, volume 3108 of *Lecture Notes in Computer Science*. Springer-Verlag, 2004.