Groups-Complexity-Cryptology
Volume 1 (2009), No. 2, 231–259

Algebraic Attacks Galore!

Martin Kreuzer
Fakultät für Informatik und Mathematik,
Universität Passau, 94030 Passau, Germany
Martin.Kreuzer@uni-passau.de

Dedicated to Benjamin Fine on the occasion of his 60th birthday.

Received: May 15, 2009

This is the first in a two-part survey of current techniques in algebraic cryptanalysis. After introducing the basic setup of algebraic attacks and discussing several attack scenarios for symmetric cryptosystems, public key cryptosystems, and stream ciphers, we discuss a number of individual methods. The XL, XSL, and MutantXL attacks are based on linearization techniques for multivariate polynomial systems. Then we look at Gröbner basis and border bases methods. In the last section we introduce attacks based on integer programming techniques and try them in some concrete cases.

Keywords: Cryptosystem, algebraic attack, polynomial system solving

2000 Mathematics Subject Classification: Primary 11T71; Secondary 13P10, 94A60

Contents

1 Introduction **232**

2 Cryptosystems **233**

3 From Cryptosystems to Polynomial Systems **235**

4 Attack Methods Based on Polynomials **236**

5 The XL, XSL and MutantXL Attacks **240**

6 The Gröbner Basis Attack **245**

7 The Border Basis Attack **249**

8 The Integer Programming Attack **253**

ISSN 1867-1144 / $ 2.50

1. Introduction

Cryptosystems (also called ciphers or encryption schemes) are important building blocks of many cryptographic protocols that play an essential role in modern life. Already in his seminal paper [44], C. E. Shannon remarked:

> *Thus, if we could show that solving a certain system requires at least as much work as solving a system of simultaneous equations in a large number of unknowns, of a complex type, then we would have a lower bound of sorts for the work characteristic.*

Since it is well-known that any encryption map between finite dimensional vector spaces over a finite field is polynomial, it is natural to represent the task of breaking a cryptosystem by the problem of solving a multivariate polynomial system of equations over a finite field. Such techniques are usually called *algebraic attacks*.

The fact that the serious study of algebraic attacks did not get off the ground for a long time after Shannon's paper is probably due to the lack of sufficiently fast computers and efficient computer algebra methods. This situation has changed dramatically in the last decades, and now algebraic attacks are an active area of research. The purpose of this paper (and its follow-up) is to give a sampling of the methods that have been developed, are being developed, or could be developed, based on the (limited) understanding of the author. The main emphasis is not to provide fine details or the latest optimizations, but to give an overview of a variety of different techniques, originating in a variety of mathematical disciplines, and to hint at possible connections or synergies.

Let us describe briefly what is ahead of us. In Section 2 we introduce some basic definitions, and in Section 3 we describe the fundamental setup of algebraic attacks. In particular, we recall the Buchberger-Möller algorithm which can be used to compute polynomial relations between plaintext and ciphertext units or between key bits and ciphertext units. Then, in Section 4, we discuss some general attack scenarios, in particular for attacking symmetric cryptosystems, public key cryptosystems, and stream ciphers.

The discussion of individual methods for performing the cryptanalysis starts in Section 5 with the XL, XSL, and MutantXL attacks. The XL attack uses a linearization technique to reduce the solution of a multivariate polynomial system to a linear system of equations. Contrary to some initial hopes, it does not solve the problem in subexponential time and, worse, it suffers from a rapid increase in memory consumption. To overcome this difficulty, several remedies have been proposed. We discuss briefly the XSL algorithm which is the subject of ongoing lively discussion, and a recent idea of J. Ding called the MutantXL algorithm. Especially the last suggestion seems to be able to rival the Gröbner basis techniques and certainly merits further investigation.

In Section 6 we introduce the Gröbner basis attack which is mainly based on the F4 and F5 algorithms of J.-C. Faugère. They rose to popularity after they were

successfully used to break the "HFE 80 Challenge". Recent improvements and additions made them even more powerful and several stream ciphers and digital signature streams have come under serious attack. An idea which is similar to the F5 algorithm and which makes it possible to include some of the techniques underlying the MutantXL attack is presented in Section 7 where we explain the border basis attack. This attack is based on the border basis algorithm (BBA) which computes (surprise!) border bases, a generalization of Gröbner bases. The BBA offers several points in which the memory consumption of the algorithm can be minimized, thereby improving the search heuristics of the Gröbner basis oriented approaches.

Finally, in Section 8, we move to a different branch of mathematics to search for efficient attack mechanisms: discrete optimization. After formulating the solution of a system of polynomial equations over $\mathbb{F}_2$ as an integer programming problem over $\mathbb{Z}$ in the straightforward way, we can apply standard IP solvers inside our algebraic attacks. Using some concrete examples, we show that this "IP attack" seems to be rather efficient and deserves to be the subject of further investigations. We end the section with a first suggestion for how to improve this IP attack in certain cases.

In view of the large number of different techniques, this overview has been split into two parts. The second part is called *Algebraic Attacks Ante Portas!* and will contain further methods coming from mathematical logic, numerical analysis, linear and multilinear algebra, algebraic geometry and other fields.

Although I have tried to be reasonably complete and quote as generously as possible, the sheer amount of work published on this subject makes it virtually impossible to paint an entirely fair and complete picture. A sincere apology goes in advance to all colleagues whose contributions have been overlooked or not represented adequately. Even so, the paper ends with a list of 46 references. For basic definitions and notation, I will adhere to the books [32] and [33] because they are the ones I know best. Of course, whatever I quote from there is also available in other monographs on computer algebra.

I can't tell you the secret of the universe,
said Mullah Nasrudin,
because then it would not be a secret anymore.

2. Cryptosystems

Algebraic attacks can be used against a variety of cryptosystems: symmetric or public key, block ciphers and stream ciphers. Let us briefly recall some basic definitions.

Definition 2.1. A *cryptosystem* (or *cipher* or *encryption scheme*) consists of the following parts:

(1) A set $\mathcal{P}$ called the set of *plaintext units*.
(2) A set $\mathcal{C}$ called the set of *ciphertext units*.

(3) A set $\mathcal{K}$ called the *key space*.
(4) For every key $k \in \mathcal{K}$, an *encryption map* $\varepsilon_k : \mathcal{P} \longrightarrow \mathcal{C}$.
(5) For every key $k \in \mathcal{K}$, a *decryption map* $\delta_k : \mathcal{C} \longrightarrow \mathcal{P}$.
(6) A map $\eta : \mathcal{K} \longrightarrow \mathcal{K}$ such that $\delta_{\eta(k)} \circ \varepsilon_k = \mathrm{id}_{\mathcal{K}}$ for all $k \in \mathcal{K}$. A pair $(k, \eta(k))$ is called a *key pair*.

The cryptosystem is called *symmetric* if $\eta(k)$ can be computed efficiently given the knowledge of k and ε_k. Otherwise, the system is called a *public key cryptosystem*. In a *block cipher*, the plaintext is broken into plaintext units and encrypted using a fixed key k. In a *stream cipher*, there is a function to generate a key stream $k_1, k_2, \ldots$ from an initial key, and then $k_1, k_2, \ldots$ are used to encrypt the individual plaintext units.

In the cases we are going to consider, the sets $\mathcal{P}$ and $\mathcal{C}$ are (subsets of) finite dimensional vector spaces over a finite field, usually of characteristic 2. The field will be denoted by $K = \mathbb{F}_q$ where $q = p^e$ and p is the characteristic. The following easy observation is the basis of all algebraic attacks.

Remark 2.2. Over a finite field K, every map $\varphi : K^n \longrightarrow K^m$ is polynomial, i.e. there exist polynomials $f_1, \ldots, f_m \in K[x_1, \ldots, x_n]$ such that

$$\varphi(a_1, \ldots, a_n) = (f_1(a_1, \ldots, a_n), \ldots, f_m(a_1, \ldots, a_n))$$

for all $a_1, \ldots, a_n \in K$. The polynomials f_i are not uniquely determined. Considering $\mathbb{X} = K^n$ as a finite point set, the polynomials f_i can be modified by adding elements of the *vanishing ideal*

$$I(\mathbb{X}) = \{g \in K[x_1, \ldots, x_n] \mid g(a_1, \ldots, a_n) = 0 \text{ for all } (a_1, \ldots, a_n) \in \mathbb{X}\}.$$

Let us illustrate this phenomenon with a non-standard look at the RSA cryptosystem.

Example 2.3. We use the RSA cryptosystem with $n = 15 = p \cdot q$, where $p = 3$ and $q = 5$. The public exponent is $e = 5$, the secret one is $d = 5$, so that $de \equiv 1 (\mathrm{mod}\ 8)$ with $8 = \varphi(n)$. The plaintext and ciphertext units are represented as tuples $(a_0, a_1, a_2, a_3) \in \mathbb{F}_2^4$ corresponding to elements $a_0 + 2a_1 + 4a_2 + 8a_3 \in \mathbb{Z}/(15)$. A straightforward calculation shows that $\varepsilon_5(a_0, a_1, a_2, a_3) = (a_0 + 2a_1 + 4a_2 + 8a_3)^5$ is then represented by $(c_0, c_1, c_2, c_3) \in \mathbb{F}_2^4$ where

$$\begin{aligned}
c_0 &= a_0a_1a_2a_3 + a_0a_1a_2 + a_0a_2 + a_0a_3 + a_2a_3 + a_0 + a_3 \\
c_1 &= a_0a_1a_2a_3 + a_0a_1a_2 + a_0a_1a_3 + a_0a_2a_3 + a_0a_1 + a_1 + a_2 + a_3 \\
c_2 &= a_1a_2a_3 + a_0a_1 + a_1a_2 + a_1a_3 + a_1 + a_2 \\
c_3 &= a_0a_1a_2a_3 + a_0a_1a_2 + a_1a_2a_3 + a_0a_1 + a_0a_2 + a_0a_3 + a_2a_3 + a_3.
\end{aligned}$$

Thus we can decipher for example the ciphertext $(1, 1, 0, 0)$ by finding the $\mathbb{F}_2$-rational solutions of the polynomial system $c_0 - 1 = 0$, $c_1 - 1 = 0$, $c_2 = 0$, $c_3 = 0$.

For instance, this can be done by computing a reduced Gröbner bases of the ideal $I = \langle c_0 - 1, c_1 - 1, c_2, c_3, a_0^2 - a_0, \ldots, a_3^2 - a_3 \rangle$. The result $G = \{a_0 - 1, a_1 - 1, a_2, a_3\}$ tells us that the plaintext unit was $(1, 1, 0, 0)$ which agrees with $3^5 \equiv 3 (\mathrm{mod}\ 15)$.

From now on we let $P = K[x_1, \ldots, x_n]$. Since the multiplicative group $\mathbb{F}_q^\times$ is cyclic, the elements of $\mathbb{F}_q$ are the zeros of $x^q - x$ and the vanishing ideal of $\mathbb{X} = K^n$ is $I(\mathbb{X}) = \langle x_1^q - x_1, \ldots, x_n^q - x_n \rangle$. We shall call this the *field ideal*. If we represent an encryption map (or a family of encryption maps) via polynomials $f_1, \ldots, f_m$ and use them to devise an algebraic attack involving the solution of a polynomial system, it is usually safe to add the equations of the field ideal to the system, since we are looking for K-rational solutions.

3. From Cryptosystems to Polynomial Systems

The construction of the polynomials $f_1, \ldots, f_m$ which represent the encryption maps ε_k is frequently performed on a case-by-case basis, taking into account the specific descriptions of ε_k. If the space of plaintext units $\mathcal{P}$ (and possibly the key space) is not too large, the algorithm below can be applied to the set of all plaintext – cyphertext pairs.

However, for large real-world cryptosystems, it is sometimes not possible to determine the actual polynomials $f_1, \ldots, f_m$. In particular, this happens if we are dealing with a symmetric cryptosystem and if the polynomials $f_1, \ldots, f_m$ depend on further indeterminates representing the key bits. In this case we can still generate polynomial relations between the (plaintext, key bits)-tuple and the ciphertext-tuple which hold for a large number of corresponding tuples via the following well-known and efficient algorithm from computer algebra.

Proposition 3.1 (Buchberger-Möller Algorithm). *Let $\mathbb{Y} = \{(p_1, k_1), \ldots, (p_s, k_s)\} \subseteq K^n \times K^\ell$ be a finite set of points. (Here the tuples p_i represent plaintext units and the tuples k_i represent the keys used to encrypt them.) Furthermore, let $q_i = \varepsilon_{k_i}(p_i)$ be the corresponding ciphertext units for $i = 1, \ldots, s$. Form the polynomial ring $Q = K[x_1, \ldots, x_n, y_1, \ldots, y_\ell]$ and consider the following instructions.*

(1) *Let $\mathcal{G} = \emptyset$, $\mathcal{O} = \emptyset$, $\mathcal{S} = \emptyset$, $L = \{1\}$, and let $\mathcal{M} = (m_{ij}) \in \mathrm{Mat}_{0,s}(K)$ be a matrix having s columns and initially zero rows. Choose a term ordering σ on Q.*

(2) *If $L = \emptyset$, continue with step (6). Otherwise, choose the term $t = \min_\sigma(L)$ and delete it from L.*

(3) *Compute the evaluation vector $(t(p_1, k_1), \ldots, t(p_s, k_s)) \in K^s$ and reduce it against the rows of $\mathcal{M}$ to obtain*

$$(v_1, \ldots, v_s) \;=\; (t(p_1, k_1), \ldots, t(p_s, k_s)) \;-\; \sum_i a_i \, (m_{i1}, \ldots, m_{is})$$

with $a_i \in K$.

(4) *If* $(v_1, \ldots, v_s) = (0, \ldots, 0)$*, append the polynomial* $t - \sum_i a_i s_i$ *to* $\mathcal{G}$*, where* s_i *is the* i^{th} *element in* $\mathcal{S}$*. Remove from* L *all multiples of* t*. Then continue with step* (2)*.*

(5) *Otherwise we have* $(v_1, \ldots, v_s) \neq (0, \ldots, 0)$*. Append* $(v_1, \ldots, v_s)$ *as a new row to* $\mathcal{M}$ *and* $t - \sum_i a_i s_i$ *as a new element to* $\mathcal{S}$*. Add* t *to* $\mathcal{O}$*, and add to* L *those elements of* $\{x_1 t, \ldots, x_n t, y_1 t, \ldots, y_\ell t\}$ *which are neither multiples of an element of* L *nor of* $\mathrm{LT}_\sigma(\mathcal{G})$*. Continue with step* (2)*.*

(6) *Row reduce* $\mathcal{M}$ *to a diagonal matrix and mimic these row operations on the elements of* $\mathcal{S}$*, considered as a column vector. Next replace* $\mathcal{S}$ *by* $\mathcal{M}^{-1}\mathcal{S}$*.*

(7) *For* $i = 1, \ldots, s$*, write* $q_i = (q_{i1}, \ldots, q_{im})$ *with* $q_{ij} \in K$*. For* $j = 1, \ldots, m$*, form the polynomial* $f_j = \sum_{i=1}^{s} q_{ij} s_i$ *where* s_i *is the* i^{th} *element of* $\mathcal{S}$*. Return* $(f_1, \ldots, f_m)$ *and* $\mathcal{G}$ *and stop.*

This is an algorithm which computes a tuple of polynomials $(f_1, \ldots, f_m) \in Q^m$ *and a tuple* $\mathcal{G}$ *such that* $(f_1(p_i, k_i), \ldots, f_m(p_i, k_i)) = q_i$ *for* $i = 1, \ldots, s$ *and* $\mathcal{G}$ *is the reduced* σ*-Gröbner basis of* $I(\mathbb{Y})$*.*

In other words, the Buchberger-Möller Algorithm yields all polynomials which model the encryption map correctly for the given plaintext units and keys. For a proof, see for instance [33], Thm. 6.3.10 and Cor. 6.3.11.

In his diploma thesis [35], J. Limbeck implemented the polynomials representing the encryption functions of a number of important cryptosystems, e.g. DES, AES, Serpent, Keeloq, HFE and a number of variations of these. These implementations in ApCoCoA (see [3]) are available from the author upon request.

4. Attack Methods Based on Polynomials

In this section we describe several attack scenarios using the algebraic representation of the encryption and decryption maps discussed above. Depending on whether one deals with block ciphers or stream ciphers, and depending on the information available to and the goals of the attacker, the following attack methods have been proposed.

Attack Methods for Symmetric Block Ciphers.

Symmetric block ciphers are typically constructed along the lines of the following figure. The cipher consists of several *rounds* in which the plaintext repeatedly undergoes (essentially) the same transformations. Here *key addition* means that a *round key* is added bitwise to the plain text unit, the *key schedule* is a function to compute from the initial key a new round key for each round, and the *S-boxes* and *diffusion layers* are various maps to achieve a homogeneous distribution of the plaintext information over the ciphertext unit.

With the exception of the S-boxes, these components are usually modeled by linear polynomials, and for the S-boxes one can normally get away with quadratic

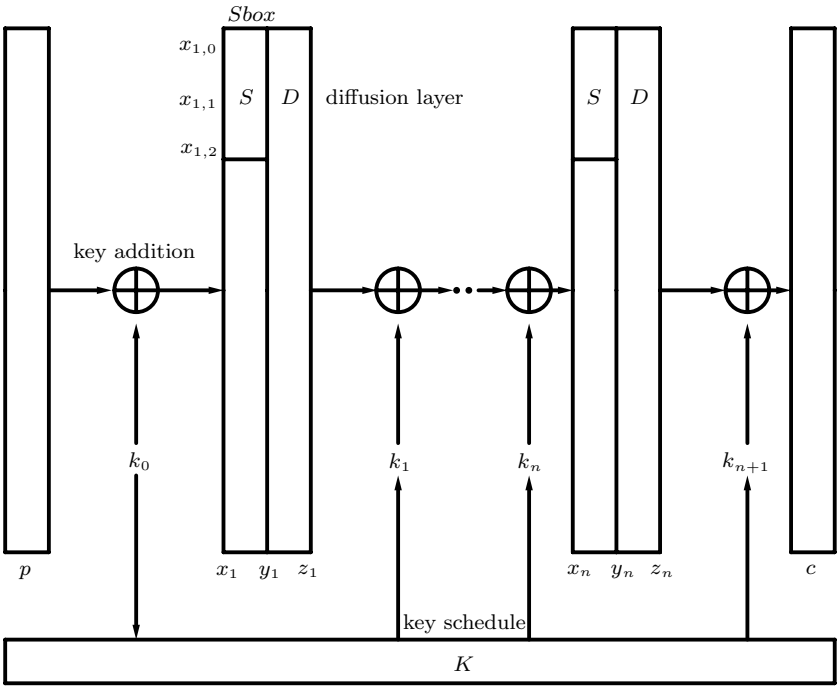

Figure 4.1: Schematic of a Symmetric Block Cipher

polynomials. Thus the basic attack method here is the following type of *known plaintext attack:*

Attack 1. *Assume that one or more plaintext-ciphertext pairs are known. Write down the encryption map as a system of linear and quadratic polynomials in the indeterminates using the initial key (or key bits) and the indeterminates representing the intermediate text units after each round. Then find a solution of the resulting multivariate polynomial system and use the key to decrypt other plaintext-ciphertext pairs.*

It is clear that we are only interested in the indeterminates representing the key. In this attack method, it is usually sufficient to know one or two plaintext-ciphertext pairs in order to have a unique solution of the polynomial system. This assumption is not unreasonable because, for instance, certain files or file types always start with the same sequence of bytes. Less likely to succeed, but in that case even more dangerous, is the following *ciphertext only attack:*

Attack 2. *Assume that the attacker knows several ciphertext units which have been encrypted using the same key. Write down the polynomials representing the encryption map a corresponding number of times, using distinct indeterminates for the various plaintexts and the same indeterminates for the keys. Solve the resulting polynomial system and obtain both the plaintext and the key.*

Finally, we note the following suggestion in [16]. It is not a specific attack but a method to find *weak keys* or statistical defects of the encryption maps. A *dif-*

ferential characteristic of an encryption map is a pair $(\Delta p, \Delta c_n)$ such that two plaintext units differing by Δp are transformed by n rounds of the encryption map into ciphertext units differing by Δc_n with a "large enough" probability. Differential cryptanalysis uses information about the number of pairs of ciphertexts having a difference predicted by the characteristic to recover (parts of) the secret key.

Attack 3. *Write down the polynomial system describing the application of the encryption map to two plaintext units differing by Δp and to intermediate or ciphertext units differing by Δc_n where $(\Delta p, \Delta c_n)$ is a differential characteristic. Solve the system to find weak keys or, for a fixed key, to find* right pairs*, i.e. pairs of plaintexts for which the characteristic holds.*

In [2], the authors propose several ways to combine the methods of differential cryptanalysis and algebraic attacks.

Algebraic Attacks for Public Key Cryptosystems.

The basic algebraic attack of a public key cryptosystem is straightforward and has already been used in Example 2.3.

Attack 4. *Write down the encryption map as a system of polynomials in the indeterminates representing the plaintext unit(s), substituting the public key and the given cipertext unit(s). Solve the polynomial system and recover the plaintext.*

Since one has complete knowledge of the encryption map, one can also generate plaintext-cipertext pairs and attack the secret key.

Attack 5. *Using the public key and arbitrarily chosen plaintext units, compute the corresponding ciphertext units. Write down a polynomial representation of the decryption map using the bits of the secret key as indeterminates. Solve the polynomial system and thus find the secret key.*

It is clear that this attack applies to all kinds of one-way functions, e.g. to the ones used to determine hash values in signature or authentication protocols.

Algebraic Attacks for Stream Ciphers

This kind of algebraic attacks has the longest history and may be considered as a well-established technique. The basic type of *synchronous stream cipher* considered here uses a sequence of *key bits* which are generated from a given initial secret key using a *key generator*. This *keystream* is then combined with the stream of plaintext bits using a XOR operation to obtain the ciphertext. Thus, schematically, the system works as follows. Typically, there is a tuple of indeterminates $(s_{i1}, \ldots, s_{in})$ which describes the *i-th internal state* of the key generator. The initial state is computed from the secret key k. Then there exists a function $f(s_{i1}, \ldots, s_{in})$ which computes the next state $(s_{i+1\,1}, \ldots, s_{i+1\,n})$ of the

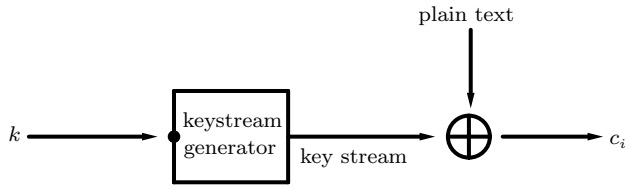

Figure 4.2: Schematic of a Synchronous Stream Cipher

key generator, a function $z_i = g(s_{i1}, \ldots, s_{in})$ which produces the next bit of the keystream, and the addition $c_i = z_i + p_i$ which yields the next ciphertext bit c_i from the next plaintext bit p_i. Let us have a look at an actual example.

Example 4.1. The synchronous stream cipher *Trivium* has been suggested for the eStream project (see [24]). It has the following structure:

(1) The initial state $(s_{01}, \ldots, s_{0\,288}) \in \mathbb{F}_2^{288}$ is computed from the 80-bit key and an 80-bit initial value.
(2) For $i = 1, 2, \ldots$, the following commands are repeated.
(3) Define three temporary values by $t_1 = s_{i\,66} + s_{i\,93}$, $t_2 = s_{i\,162} + s_{i\,177}$, and $t_3 = s_{i\,243} + s_{i\,288}$.
(4) Compute the i-th keystream bit $z_i = t_1 + t_2 + t_3$.
(5) Replace t_1 by $t_1 + s_{i\,91}s_{i\,92} + s_{i\,171}$, replace t_2 by $t_2 + s_{i\,175}s_{i\,176} + s_{i\,264}$, and replace t_3 by $t_3 + s_{i\,286}s_{i\,287} + s_{i\,69}$.
(6) Let $(s_{i+1\,1}, \ldots, s_{i+1\,93})$ be $(t_3, s_{i1}, \ldots, s_{i\,92})$, let $(s_{i+1\,94}, \ldots, s_{i+1\,177})$ be $(t_1, s_{i\,94}, \ldots, s_{i\,176})$, and let $(s_{i+1\,178}, \ldots, s_{i+1\,288})$ be $(t_2, s_{i\,178}, \ldots, s_{i\,287})$.

It is evident that all assignments are given by simple, sparse, linear or quadratic polynomials over $\mathbb{F}_2$.

The typical attack against such a stream cipher is a known plaintext attack which proceeds as follows.

Attack 6. *Introduce indeterminates $(s_1, \ldots, s_n)$ representing the initial state of the key generator. Moreover, introduce indeterminates $z_0, z_1, \ldots$ representing the key stream bits. For each clock i, get an equation which represents z_i as a polynomial in the indeterminates $s_1, \ldots, s_n$. Using the known plaintext-ciphertext pairs, determine a part of the keystream and solve the polynomial system to recover the internal state of the keystream generator. Then use the result to decrypt the remainder of the ciphertext.*

Of course, one can also try to use the knowledge of the internal state to clock the stream cipher backwards and to recover the secret key. A more special type of attack (proposed in [18]) is possible for the common type of stream cipher in which the function $f(s_{i1}, \ldots, s_{in})$ that computes the next state is *linear* (e.g. stream ciphers based on LFSRs). The details of this method (called *fast algebraic attack*) can be found in [14] and [30].

Attack 7. *Suppose that* $f(s_{i1}, \ldots, s_{in})$ *is linear, so that* $z_i = g(f^i(s_{01}, \ldots, s_{0n}))$ *is a polynomial of degree* $\deg(g)$ *in the indeterminates* $s_{01}, \ldots, s_{0n}$. *Find a relation of the form* $z_i \cdot h_1(s_{01}, \ldots, s_{0n}) = h_2(s_{01}, \ldots, s_{0n})$ *with polynomials* h_1, h_2 *of "small" degrees* $deg(h_1) < deg(h_2)$ *and use it to derive a relation of the form* $\sum_{i=i_0}^{i_1} \alpha_i z_i h_1(f^i(s_{01}, \ldots, s_{0n})) = 0$ *with* $\alpha_i \in \mathbb{F}_2$. *Finally, write down this relation for many consecutive values of* i_0 *and use the additional polynomials to speed up Attack* 6 *substantially.*

In all cases, we have seen that the main task for a successful attack is to solve a multivariate polynomial system over a finite field. Therefore the rest of this article deals with different methods that have been used or suggested for this purpose in the context of polynomial systems derived from algebraic attacks.

5. The XL, XSL and MutantXL Attacks

The XL Attack.

The XL Attack uses the XL Algorithm which in turn is based on a technique called *relinearization* introduced by A. Kipnis and A. Shamir in [34]. Let us describe this technique via the example given there.

Example 5.1. Suppose we want to solve the following system of polynomial equations in $\mathbb{F}_7[x_1, x_2, x_3]$.

$$\begin{aligned}
3x_1^2 + 5x_1x_2 + 5x_1x_3 + 2x_2^2 + 6x_2x_3 + 4x_3^2 &= 5\\
6x_1^2 + x_1x_2 + 4x_1x_3 + 4x_2^2 + 5x_2x_3 + x_3^2 &= 6\\
5x_1^2 + 2x_1x_2 + 6x_1x_3 + 2x_2^2 + 3x_2x_3 + 2x_3^2 &= 5\\
2x_1^2 + x_1x_3 + 6x_2^2 + 5x_2x_3 + 5x_3^2 &= 0\\
4x_1^2 + 6x_1x_2 + 2x_1x_3 + 5x_2^2 + x_2x_3 + 4x_3^2 &= 0.
\end{aligned}$$

For every product x_ix_j, we introduce a new indeterminate y_{ij} and solve the resulting *linearized* system of equations. We get $y_{11} = 2+5z$, $y_{12} = z$, $y_{13} = 3+2z$, $y_{22} = 6+4z$, $y_{23} = 6+z$, and $y_{33} = 5+3z$ with $z \in \mathbb{F}_7$. To isolate the correct solution, we use the fundamental syzygies of the terms x_ix_j, namely $y_{11}y_{23} = y_{12}y_{13}$, $y_{12}y_{23} = y_{13}y_{22}$, and $y_{12}y_{33} = y_{13}y_{23}$ and obtain new equations for z, namely

$$3z^2 + z + 5 = 0, \qquad 4z + 4 = 0, \qquad z^2 + 4z + 3 = 0.$$

Now we apply a *relinearization step*: we introduce $z_1 = z$ and $z_2 = z^2$, solve the linear system, and find $z_1 = 6$, $z_2 = 1$. This yields $y_{11} = 4$, $y_{22} = y_{33} = 2$, and hence $x_1 = \pm 2$, $x_2 = \pm 3$, $x_3 = \pm 3$. Finally, $y_{12} = 6$ and $y_{23} = 5$ imply $(x_1, x_2, x_3) \in \{(2, 3, 4), (5, 4, 3)\}$.

It is already apparent from this small example that a potentially huge number of new indeterminates has to be introduced, so that the linear system has to be solved with great care to preserve sparseness. Based on this relinearization

technique, N. Courtois, A. Klimov, J. Patarin and A. Shamir proposed in [17] the XL Algorithm (which stands for *eXtended Linearization*) for solving a system of multivariate quadratic equations

$$\begin{aligned} f_1(x_1,\ldots,x_n) &= 0 \\ &\vdots \\ f_m(x_1,\ldots,x_n) &= 0 \end{aligned}$$

with $f_i \in K[x_1,\ldots,x_n]$ over a field K. (Recall that, in our case, K is finite and we are after just *one* K-rational solution.)

Remark 5.2. By *XL Algorithm* we mean the procedure defined by the following steps.

(1) Choose a number $d > 2$ such that $d \geq n/\sqrt{m}$.
(2) Form all products $x^\alpha \cdot f_i$ where $1 \leq i \leq m$ and $x^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ is a term of degree $\leq d-2$.
(3) Linearize the set of all $x^\alpha f_i$. In other words, introduce a new indeterminate y_j for every term of degree $\leq d$ in $K[x_1,\ldots,x_n]$ and solve the linear system using Gaußian elimination. The elimination has to be performed in such a way that the indeterminates y_j corresponding to some set $\{1, x_k, \ldots, x_k^d\}$ are eliminated last.
(4) Assume that step (3) yields at least one univariate equation $c_0 + c_1 x_k + \cdots + c_d x_k^d = 0$. Solve this equation.
(5) Substitute the values of x_i back into the system and simplify it. Repeat the process to find the values of the other indeterminates.

If the base field is $K = \mathbb{F}_2$, it suffices to use squarefree terms x^α in step (2).

The hope expressed by the authors of [17] was that this method could find solutions of large overdefined systems of quadratic equations over finite fields in *subexponential time*. However, careful analyses of its running time in [20] and [21] make this appear unlikely. In a number of papers the XL Algorithm has been related to the Gröbner basis methods we discuss in the next section (see for instance [5], [11], and [45]). Let us apply it to the following simple example taken from [1].

Example 5.3. Over the polynomial ring $\mathbb{F}_{127}[x_1, x_2]$, consider the quadratic polynomial system

$$\begin{aligned} f_1 &= x_1^2 + 80x_1x_2 + 114 = 0 \\ f_2 &= x_2^2 + 107x_1x_2 + 29 = 0 \end{aligned}$$

and apply the XL Algorithm.

(1) Choose $d = 4$.
(2) Form the products $x_1^2 f_i$, $x_1x_2 f_i$, $x_2^2 f_i$, $x_1 f_i$, $x_2 f_i$, and f_i with $i = 1, 2$.

(3) Linearization and Gaußian elimination (where $1, x_2, x_2^2, x_2^3, x_2^4$ are eliminated last) amounts to interreducing the products $x^\alpha f_i$ and yields the set

$$\begin{aligned}\{&x_1^4 + 106x_2^2 + 115,\ x_1^3x_2 + 66x_2^2 + 28,\ x_1^3 + 119x_2^3 + 109x_2,\\ &x_1^2x_2^2 + 61x_2^2 + 113,\ x_1^2x_2 + 4x_2^3 + 103x_2,\ x_1^2 + 4x_2^2 + 103,\\ &x_1x_2^3 + 34x_2^2 + 124,\ x_1x_2^2 + 119x_2^3 + 43x_2,\ x_1x_2 + 119x_2^2 + 43x_2,\\ &x_1 + 24x_2^3 + 17x_2,\ x_2^4 + 74x_2^2 + 67\}\end{aligned}$$

(4) The equation $x_2^4 + 74x_2^2 + 67 = (x_2 - 36)(x_2 + 36)(x_2^2 - 27) = 0$ yields $x_2 = \pm 36$.

(5) Substituting x_2 back, we find the solution $x_2 = 91$, $x_1 = 89$.

The XSL Attack

To overcome the inefficiency of the XL Algorithm and take advantage of the sparseness present in polynomial systems arising from algebraic attacks, N. Courtois and J. Pieprzyk proposed in [19] a further method called XSL (which stands for *eXtended Sparse Linearization*). This method is particularly tailored to attack the following kind of cryptosystem.

Definition 5.4. A symmetric block cipher is called an *XSL cipher* if it has the following structure.

(1) The first round $i = 1$ starts with a XOR with the session key k.
(2) There is a layer of b parallel S-boxes, each working on s bits.
(3) Next there is a linear diffusion layer.
(4) Finally, there is a XOR with a round key k_i.
(5) Repeat (2)–(4) until N rounds have been completed.

For an XSL cipher, the XSL Attack can informally be described as follows (see [19] and [12]).

Remark 5.5. Given an XSL cipher, perform the following steps.

(1) Form the system of multivariate quadratic polynomial equations describing the cryptosystem. Let each S-box correspond to r equations which involve a total of t terms. Choose a basis of $t - r$ terms (involving 1, but avoiding indeterminates) and express the remaining r terms as linear combinations of the basis.
(2) Select a parameter p. (Some heuristics for this choice are given in [19].) Multiply the linear layer equations (from the encryption and the key schedule) by terms chosen from the bases of $p-1$ different S-boxes. (Ensure that the generated equations contain only terms from *different* S-boxes.)
(3) As much as possible, perform substitutions of terms not in the basis by their expressions using linear combinations of terms in the basis.
(4) Apply the so-called *T'-method:* Multiply selected equations by single indeterminates to create further equations. If $\mathcal{T}$ denotes the set of all terms in

the set and x_i is an indeterminate, let $\mathcal{T}_i' = \{t \in \mathcal{T} \mid x_i t \in \mathcal{T}\}$. Perform a Gaußian elimination to bring the system to a form in which every term is a linear combination of the terms in $\mathcal{T}_i'$. Then multiply these equations by x_i, reduce modulo the field equations and append any new linearly independent equations to the system. Repeat this process as long as possible.

(5) Finally, apply linearization to the entire system and use Gaußian elimination to solve it.

As can be seen, the XSL Attack is a dedicated method for solving specific block ciphers, for instance AES and Serpent. However, its practical applicability has been the subject of intense discussion (see for instance [12], [40], and [46]). Several variants and improvements have been proposed, but a final conclusion of the debate has not been reached. Since the analysis of the XSL Attack in [19] is not universally accepted, we leave this topic here and move to a more recent idea on how to improve the XL Attack.

The MutantXL Attack

In [22], J. Ding *et al.* suggested a new strategy to speed up the XL Attack which is based on the concept of *mutants*. The idea is that in the process of generating new equations, polynomials of small degree should be treated preferentially. Let $f_1 = \cdots = f_m = 0$ be a system of polynomial equations over $\mathbb{F}_q$, i.e. let $f_1, \ldots, f_m \in \mathbb{F}_q[x_1, \ldots, x_n]$. We are interested in finding $\mathbb{F}_q$-rational solutions of this system. Thus we will be working over the ring

$$R \;=\; \mathbb{F}_q[x_1, \ldots, x_n]/\langle x_1^q, \ldots, x_n^q \rangle$$

where we reduce everything modulo the field equations. In the process of solving the system, we are typically generating futher elements of the ideal $I = \langle f_1, \ldots, f_m \rangle$ in R. In this setting, mutants are defined as follows.

Definition 5.6. Let $g = h_1 f_1 + \cdots + h_m f_m \in I$. The *level* of this representation is the number $\ell = \max\{\deg(h_i f_i) \mid i \in \{1, \ldots, m\}, h_i \neq 0\}$. Then the *level* of g with respect to $(f_1, \ldots, f_m)$ is defined to be the minimal level of any representation of g. We say that g is a *mutant* with respect to $(f_1, \ldots, f_m)$ if $\deg(g)$ is smaller than the level of g.

In other words, a mutant of degree d is a polynomial which cannot be found by forming linear combinations of products tf_i where $t \in \mathbb{T}^n$ is a term such that $\deg(tf_i) \leq d$ and where $i \in \{1, \ldots, m\}$. The MutantXL Attack is based on the following modification of the XL algorithm.

Remark 5.7. As above, let $f_1, \ldots, f_m \in R$. The *MutantXL Algorithm* is the procedure defined by the following steps.

(1) Interreduce $F = \{f_1, \ldots, f_m\}$, let $d = e = \min\{\deg(f_1), \ldots, \deg(f_m)\}$, and let $G = F$.

(2) Linearize G and use Gaußian elimination to bring it into row echelon form.
(3) If there are univariate polynomials in G, determine the values of the corresponding indeterminates. If this solves the system, return the result. Otherwise, substitute the values and continue with step (1), applied to polynomials in a smaller ring.
(4) Form the set M of all polynomials of degree $< e$ in G. (These polynomials are mutants with respect to F.)
(5) If $M \neq 0$, multiply each $g \in M$ by all terms of degree $d - \deg(g)$ and replace g in G by the resulting polynomials. Let $e = \min\{\deg(g) \mid g \in M\} + 1$ and continue with step (2).
(6) Replace all $g \in G$ of degree d by all possible products $x_i g$ with $i \in \{1, \ldots, n\}$, increase d by one, let $e = d$, and continue with step (2).

Let us consider this algorithm in a small example (due to J. Ding) of a polynomial system derived from the HFE cryptosystem.

Example 5.8. Over the ring $R = \mathbb{F}_2[x_1, \ldots, x_4]/\langle x_1^2 - x_1, \ldots, x_4^2 - x_4\rangle$, consider the following system of equations.

$$\begin{aligned} f_1 &= x_1x_2 + x_2x_3 + x_2x_4 + x_3x_4 + x_1 + x_3 + 1 = 0 \\ f_2 &= x_1x_2 + x_1x_3 + x_1x_4 + x_3x_4 + x_2 + x_3 + 1 = 0 \\ f_3 &= x_1x_2 + x_1x_3 + x_2x_3 + x_3x_4 + x_1 + x_4 + 1 = 0 \\ f_4 &= x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + 1 = 0. \end{aligned}$$

We follow the steps of the MutantXL algorithm.

(1) Let $d = e = 2$ and $G = F$.
(2) Gaußian elimination yields the system

$$\begin{aligned} g_1 &= x_1x_2 + x_2x_3 + x_2x_4 + x_3x_4 + x_1 + x_3 + 1 = 0 \\ g_2 &= x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_1x_2 = 0 \\ g_3 &= x_1x_4 + x_2x_3 + x_1 + x_2 + x_3 + x_4 = 0 \\ g_4 &= x_1 + x_2 + 1 = 0. \end{aligned}$$

(4) We have $M = \{g_4\}$.
(5) Let $G = \{g_1, g_2, g_3, g_5, g_6, g_7\}$ with $g_5 = x_1x_2$, $g_6 = x_1x_3 + x_2x_3 + x_3$, and $g_7 = x_1x_4 + x_2x_4 + x_4$. Set $e = 2$.
(2) Gaußian elimination yields the new polynomials $\tilde{g}_5 = x_2x_3 + x_2x_4 + x_3x_4 + x_1 + x_3 + 1$, $\tilde{g}_6 = x_3x_4 + x_1 + x_3 + x_4 + 1$, and $\tilde{g}_7 = x_3 + x_4 + 1$.
(4) We have $M = \{\tilde{g}_7\}$.
(5) Replace $\tilde{g}_7$ in G by $g_8 = x_1x_3 + x_1x_4$, $g_9 = x_2x_3 + x_2x_4 + x_2$, and $g_{10} = x_3x_4$. Set $e = 2$.
(2) Gaußian elimination yields $\tilde{g}_8 = x_2 + x_4$ and $\tilde{g}_9 = x_4 + 1$.
(3) We substitute $x_4 = 1$ everywhere and get the solution $(x_1, x_2, x_3, x_4) = (0, 1, 0, 1)$.

The MutantXL Attack has recently been improved further in [37]. It seems to be comparable in speed to the Gröbner basis methods explained in the next section. Moreover, it is clear that the mutant strategy can also be used to guide and possibly improve the Gröbner basis and border basis computations below.

6. The Gröbner Basis Attack

In the following, we assume that the reader has a basic understanding of Gröbner bases and the Buchberger algorithm, e.g. as laid out in Chapters 1 and 2 of [32]. Let $K = \mathbb{F}_q$ be a finite field, and let $f_1, \ldots, f_m \in K[x_1, \ldots, x_n]$ be a set of polynomials. We want to find the K-rational solutions of the polynomial system of equations

$$\begin{aligned} f_1(x_1, \ldots, x_n) &= 0 \\ &\vdots \\ f_m(x_1, \ldots, x_n) &= 0. \end{aligned}$$

To this end, we form the ideal $I = \langle f_1, \ldots, f_m \rangle + \langle x_1^q - x_1, \ldots, x_n^q - x_n \rangle$. If we are able to compute a Gröbner basis of I, we can use it to solve the system in several ways.

Remark 6.1. In the above setting, let σ be a term ordering on $\mathbb{T}^n$, and let G be the reduced σ-Gröbner basis of I.

(1) If I is in normal x_n-position and $\sigma = \texttt{Lex}$ is the lexicographic term ordering, then the *Shape Lemma* (see [32], Thm. 3.7.25) says that G is of the form

$$G \;=\; \{x_1 - h_1(x_n), \ldots, x_{n-1} - h_{n-1}(x_n), h_n(x_n)\}$$

with $h_1, \ldots, h_n \in K[x_n]$. After factoring h_n, we can immediately read off the zeros of I, i.e. the solutions of the system. Notice that the ideal I is automatically a 0-dimensional radical ideal. We say that I is *in normal x_n-position* if the x_n-coordinates of the zeros of I are pairwise distinct. This is no serious restriction in the cases we are interested in, because the ideals under consideration have usually only very few solutions. If I is not in normal x_n-position, a linear coordinate change may be employed (see [32], Sect. 3.8).

(2) Suppose that the polynomial system has only one K-rational solution $(a_1, \ldots, a_n) \in K^n$. Then *every* reduced Gröbner basis G of I is of the form $G = \{x_1 - a_1, \ldots, x_n - a_n\}$. Similarly, if there are exactly two solutions, the reduced Gröbner bases of I have the form $G = \{(x_i - a_i)(x_i - b_i)\} \cup \{\ell_j(x_i, x_j) \mid j \neq i\}$ for some $i \in \{1, \ldots, n\}$, where $\ell_j \in K[x_i, x_j]$ is linear.

The computation of a Gröbner basis of I is usually achieved by a variant of the Buchberger algorithm introduced in [10]. (For a description using our notation, see [32], Thm. 2.5.5.) In his papers [25] and [26], J.-C. Faugère presented improved

versions of Buchberger's algorithm. Since they are the bases of essentially all successful Gröbner basis attacks until now, let us briefly sketch their main ideas.

The algorithm called "F4" is, in effect, a particular strategy for performing the steps of the Buchberger algorithm in a certain way which takes advantage of fast linear algebra techniques and possible sparseness of the generating polynomials $f_1, \ldots, f_m$ of I. It uses the following matrix representation of vector spaces of polynomials.

Definition 6.2. Let $G = (g_1, \ldots, g_r) \in K[x_1, \ldots, x_n]^r$, let σ be a term ordering on $\mathbb{T}^n$, and let $S = \bigcup_{i=1}^r \mathrm{Supp}(g_i)$. We write $S = \{t_1, \ldots, t_s\}$ with $t_1 >_\sigma \cdots >_\sigma t_s$ and form the matrix $M_\sigma(V) \in \mathrm{Mat}_{r,s}(K)$ whose (i,j)-entry is the coefficient of t_j in g_i. Then $M_\sigma(G)$ is called the *coefficient matrix* of the tuple $(g_1, \ldots, g_r)$ with respect to σ.

With this terminology in mind, we can describe a simplified version of the F4 Algorithm as follows.

Remark 6.3. Let $F = (f_1, \ldots, f_m) \in K[x_1, \ldots, x_n]^m$. The *F4 Algorithm* for computing a Gröbner basis of I consists of the following steps.

(1) Choose a degree compatible term ordering σ and form the coefficient matrix $M_\sigma(F)$.
(2) Reduce $M_\sigma(F)$ to row echelon form and put the polynomials corresponding to the non-zero rows into a tuple G. Let B be the set of *critical pairs* of G.
(3) For $d = 1, 2, \ldots$, repeat the following steps until $B = \emptyset$. Then return G and stop.
(4) Form the set of critical pairs $B_{\leq d}$ of degree $\leq d$ and remove it from B. Let S_d be the set of all left-hand and right-hand sides of S-polynomials S_{ij} of $B_{\leq d}$. Form the set S'_d of all tg where $t \in \mathbb{T}^n$ and $g \in G$ which may be used during the top-reduction of S_d. (This is called *symbolic preprocessing.*)
(5) Form the coefficient matrix of (G, S_d, S'_d) and reduce it to row echelon form. Put the polynomials corresponding to new non-zero rows into G, put all new critical pairs into B, and continue with (3).

There are two main advantages of this algorithm in the cryptanalytic setting: by the way the new rows of the matrices are generated in the symbolic preprocessing phase, the growth of the space consumption of the algorithm is kept under control, and for the reduction to row echelon form one can choose variants of Gaußian elimination which take advantage of possible sparseness of the matrices (e.g. structured Gaußian elimination, see [36] and [41], or Lanczos algorithm, see [38]).

If one modifies the XL algorithm such that it proceeds degree by degree, it can be seen as a version of the F4 algorithm, as was shown in [45]. Let us try the F4 algorithm in a small example.

Example 6.4. Over the field $K = \mathbb{F}_2$, consider the ideal $I = \langle x^2 - x, y^2 -$

$y, z^2 - z, f_1, f_2, f_3\rangle$ in $K[x, y, z]$, where $f_1 = xy + xz + 1$, $f_2 = xz + yz + z$, and $f_3 = xy + xz + y + 1$. We compute a Gröbner basis of I with respect to $\sigma = \texttt{DegrevLex}$ following the above version of the F4 algorithm, applied to $F = (x^2 - x, y^2 - y, z^2 - z, f_1, f_2, f_3)$.

(1) We have $\mathrm{Supp}(F) = (x^2, xy, xz, y^2, yz, z^2, x, y, z, 1)$ and

$$M_\sigma(F) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

(2) For the row echelon form, we replace f_3 by $g_3 = f_3 - f_1$ and have the new last row $(0, 0, 0, 0, 0, 0, 0, 1, 0, 0)$. The pivot elements correspond to the leading terms $\{x^2, y^2, z^2, xy, xz, y\}$ and yield $G = \{g_1, g_2, g_3, g_4, g_5, g_6\}$ with $g_1 = x^2 + x$, $g_2 = y^2 + y$, $g_3 = z^2 + z$, $g_4 = f_1$, $g_5 = f_2$, and $g_6 = y$. Taking into account Buchberger's first criterion, the critical pairs are $B = \{(1, 4), (1, 5), (2, 4), (2, 6), (3, 5), (4, 5), (4, 6)\}$.

(4) For $d = 2$, we have $B_{\le 2} = \{(2, 6), (4, 6)\}$. They yield the additional rows

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

(5) The row echelon form has one new non-zero row $(0, 0, 0, 0, 0, 0, 0, 1, 1)$, corresponding to $g_7 = z + 1$. There is no new critical pair.

(4) In degree $d = 3$ we have $B_{\le 3} = \{(1, 4), (1, 5), (2, 4), (3, 5)\}$. The new rows have additional entries corresponding to $(x^2y, x^2z, xy^2, xyz, xz^2, y^2z, yz^2)$ and are

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

(5) The row echelon form has one new non-zero row, namely

$$(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1)$$

corresponding to $g_8 = x + 1$.

(2) Since $B = \emptyset$, the algorithm stops and returns $G = \{g_1, \ldots, g_8\}$.

The polynomials g_6, g_7, g_8 of the resulting Gröbner basis show that $(1, 0, 1)$ is the only zero of I.

One problem in the F4 algorithm is that there may still be many critical pairs whose corresponding S-polynomials reduce to zero. To address it, J.-C. Faugère developed his algorithm "F5". The details of this algorithm are too complicated to be described here. For extensive discussions, we refer to the original paper [26] and to the elaborations in [43] and [28]. Let us collect a few aspects which are important to our algebraic attacks.

Remark 6.5.

(1) The F5 algorithm is optimized for the case when the input polynomials form a regular sequence. In this case, it has been shown to avoid all unnecessary reductions to zero. For highly overdetermined systems (such as the ones coming from algebraic attacks), its efficiency is not well-understood, although very good in practise.

(2) There is no complete proof yet that the F5 algorithm terminates on regular sequence inputs. (The original proof in [26] seems to contain a gap.) For overdetermined systems, in particular the ones defined over finite fields and containing the field equations, the "algorithm" apparently fails to terminate quite often (see [43], p. 43).

(3) One of the main points of the F5 algorithm is the replacement of the traditional Buchberger criteria for avoiding the treatment of unnecessary critical pairs with new "F5 criteria". It seems to be possible to combine the F5 techniques with the classical Buchberger criteria (see [28]) but has not been done so far. Similarly, the current forms of the F5 algorithm seem not to use the F4 techniques described above.

(4) The F5 algorithm is an *incremental Gröbner basis algorithm*, i.e. it adds the input polynomials one by one and computes bases of the subideals iteratively. Therefore the order in which the input polynomials are passed to the algorithm matters considerably.

Despite these remarks, the F5 algorithm has been the algorithm of choice for Gröbner basis attacks. In part, this is certainly due to the success in [27] where the "HFE 80 Challenge" was solved. Further successes with algebraic attacks using the F4 and F5 algorithms were reported in [23] against the perturbed Matsumoto-Imai cryptosystem, in [18] and [4] against certain stream ciphers, and in [7] and [8] against the digital signature schemes TRMS and UOV.

Our last remark about the use of variants of the Buchberger algorithm for algebraic attacks is that, over the base field $K = \mathbb{F}_2$, the usual Buchberger criteria for avoiding unnecessary critical pairs may be complemented by the following new criterion shown in [9].

Proposition 6.6. *Let σ be a term ordering on $\mathbb{T}^n$, let $I \subseteq P = \mathbb{F}_2[x_1, \ldots, x_n]$ be an ideal containing the field polynomials, and let $f \in I$ be a polynomial of*

the form $f = \ell \cdot g$ *where* $\ell, g \in P$ *and* $\mathrm{LT}_\sigma(\ell) = x_i$. *Then the S-polynomial* $S(f, x_i^2 + x_i)$ *has a t-representation for some term* $t <_\sigma \mathrm{lcm}(\mathrm{LT}_\sigma(f), x_i^2)$.

In particular, the critical pair $(\mathrm{LT}_\sigma(f), x_i^2)$ *may be skipped during a Gröbner basis computation.*

7. The Border Basis Attack

All of the preceding attacks were in essence based on the construction of further polynomials in the ideal I generated by the original system. This system is defined over a finite field K and contains the field equations. Hence I is a 0-dimensional radical ideal in $K[x_1, \ldots, x_n]$. And all of these attacks face more or less the same difficulty: the number of newly generated polynomials grows very fast and all too soon the algorithms run out of memory.

This leads us to the idea to use border bases in order to solve the polynomial system. For 0-dimensional polynomial ideals, they provide a more flexible concept than Gröbner bases. The border basis algorithm for computing them can be tailored to proceed as space-efficiently as possible, and it can be conditioned to search for special polynomials in the ideal, e.g. for the elimination polynomials in $I \cap K[x_i]$. Since the theory of border bases is not as well-known as that of Gröbner bases, we briefly recall the main definitions. For details and proofs we refer to [33], Sect. 6.4.

As usual, we let K be a (finite) field and $P = K[x_1, \ldots, x_n]$. We assume that $f_1, \ldots, f_m \in P$ generate a 0-dimensional polynomial ideal $I = \langle f_1, \ldots, f_m \rangle$.

Definition 7.1. Let $\mathcal{O} = \{t_1, \ldots, t_\mu\}$ be a finite set of terms in $\mathbb{T}^n$.

(1) The set $\mathcal{O}$ is called an *order ideal* if $t \in \mathcal{O}$ and $t' \mid t$ imply $t' \in \mathcal{O}$, i.e. if $\mathcal{O}$ is closed under forming divisors.
(2) The set $\partial\mathcal{O} = (x_1\mathcal{O} \cup \cdots \cup x_n\mathcal{O}) \setminus \mathcal{O}$ is called the *border* of $\mathcal{O}$.
(3) Let $\partial\mathcal{O} = \{b_1, \ldots, b_\nu\}$. A set of polynomials $G = \{g_1, \ldots, g_\nu\} \subset P$ is called an *$\mathcal{O}$-border prebasis* if its elements are of the form $g_j = b_j - \sum_{i=1}^{\mu} c_{ij} t_j$ with $c_{ij} \in K$.
(4) An $\mathcal{O}$-border prebasis is called an *$\mathcal{O}$-border basis* if the residue classes of the elements of $\mathcal{O}$ form a K-vector space basis of the ring $P/\langle g_1, \ldots, g_\nu \rangle$.

In the following we let $\mathcal{O} = \{t_1, \ldots, t_\mu\}$ be an order ideal and $\partial\mathcal{O} = \{b_1, \ldots, b_\nu\}$. Of course, we shall say that G is an $\mathcal{O}$-border (pre-) basis of I if G is an $\mathcal{O}$-border (pre-) basis and G generates I. Let us collect some basic properties of $\mathcal{O}$-border (pre-) bases.

Proposition 7.2. *Let* $G = \{g_1, \ldots, g_\nu\}$ *be an* $\mathcal{O}$*-border prebasis of* I.

(*1*) *Using the* Border Division Algorithm*, we can represent every* $f \in P$ *in the form* $f = h_1 g_1 + \cdots + h_\nu g_\nu + c_1 t_1 + \cdots + c_\mu t_\mu$ *with* $h_i \in P$ *and* $c_j \in K$.
(*2*) *The residue classes of the elements of* $\mathcal{O}$ *generate the* K*-vector space* P/I.
(*3*) *If* $\mathcal{O} = \mathbb{T}^n \setminus \mathrm{LT}_\sigma(I)$ *for some term ordering* σ*, then* I *has an* $\mathcal{O}$*-border*

basis G. The elements of G corresponding to the corners of $\mathrm{LT}_\sigma(I)$ (i.e. the minimal generators of this monomial ideal) form the reduced σ-Gröbner basis of I.

(4) *If I has an $\mathcal{O}$-border basis, it is uniquely determined.*

The following easy example shows that I has, in general, many more border bases than reduced Gröbner bases. One of the goals of the border basis attack is to capitalize on this increased flexibility.

Example 7.3. Let $I \subset \mathbb{F}_2[x,y]$ be the ideal generated by $\{x^2+xy, y^2+xy, x^3\}$, and let $\mathcal{O} = \{1, x, y, xy\}$. Then we have $\partial\mathcal{O} = \{x^2, x^2y, xy^2, y^2\}$, and the set $G = \{x^2+xy, x^2y, xy^2, y^2+xy\}$ is an $\mathcal{O}$-border basis of I.

However, this $\mathcal{O}$-border basis does not contain a reduced σ-Gröbner basis for any term ordering σ because $x >_\sigma y$ implies $xy = \mathrm{LT}_\sigma(y^2+xy) \in \mathrm{LT}_\sigma(I)$ and $x <_\sigma y$ implies $xy = \mathrm{LT}_\sigma(x^2+xy) \in \mathrm{LT}_\sigma(I)$. Thus the order ideal $\mathcal{O} = \{1, x, y, xy\}$ is not of the form $\mathbb{T}^2 \setminus \mathrm{LT}_\sigma(I)$.

Given a system of generators $\{f_1, \ldots, f_m\}$ of I, the *Border Basis Algorithm (BBA)* computes an order ideal $\mathcal{O}$ and an $\mathcal{O}$-border basis of I. A generic version of this algorithm was introduced in [39]. The following discussion is based on the detailed elaboration in [31].

All the computations performed by the BBA take place in a finite dimensional K-vector subspace U of P called the *computational universe*. At certain points of the algorithm the space U has to be enlarged, and exactly these enlargements enable us to control the "direction" and the "speed" of the spacial growth of the computation. A second important ingredient is the following method to approximate the intersection $I \cap U$.

Definition 7.4. Let $F \subseteq U$ be two finite dimensional K-vector subspaces of P. Inductively, we define the vector subspaces $F_0 := F$ and $F_{i+1} = F_i^+ \cap U$ for $i = 0, 1, \ldots$, where $F_i^+ = F_i + x_1F_i + \cdots + x_nF_i$. Then the union $F_U = \bigcup_{i\geq 0} F_i$ is called the *U-stable span* of F.

The vector space F in this definition should be viewed as the part of $I \cap U$ that we know already. By computing F_U, we enlarge it to produce a kind of "approximation" of $I \cap U$. The following criterion is then the key point of the BBA (cf. [31], Prop. 16).

Proposition 7.5. *Let U be a vector subspace of P, let F be a vector subspace of I which generates I and satisfies $F^+ \cap U = F$, and let $\mathcal{O}$ be an order ideal such that $U = F \oplus \langle\mathcal{O}\rangle_K$ and $\partial\mathcal{O} \subseteq U$. Then I has an $\mathcal{O}$-border basis which is contained in U.*

The last ingredient of the BBA is the *Final Reduction Algorithm* which can be used in the setting of the preceding proposition to extract the $\mathcal{O}$-border basis of I

from the bases of F and U. Since it is a kind of technicality, we do not repeat the details here but refer instead to [31], Prop. 17. Finally, we are ready to enunciate the basic version of the BBA (which is actually called the *Improved Border Basis Algorithm* in [31], Prop. 21).

Proposition 7.6. (The Border Basis Algorithm)
Let $I = \langle f_1, \ldots, f_m \rangle$ be a 0-dimensional ideal in $P = K[x_1, \ldots, x_n]$. Then the following algorithm computes an order ideal $\mathcal{O}$ and the $\mathcal{O}$-border basis of I.

(*1*) *Let U be the order ideal generated by $\bigcup_{i=1}^{m} \mathrm{Supp}(f_i)$. Choose a degree compatible term ordering σ.*
(*2*) *Interreduce $\{f_1, \ldots, f_m\}$ to get a K-basis V of $F = \langle f_1, \ldots, f_m \rangle_K$ with pairwise distinct leading terms.*
(*3*) *Compute a basis extension $V \cup W'$ of V such that $V \cup W'$ is a K-basis of F^+ with pairwise distinct leading terms.*
(*4*) *Let $W = \{w \in W' \mid \mathrm{LT}_\sigma(w) \in U\}$.*
(*5*) *If $\bigcup_{w \in W} \mathrm{Supp}(w) \not\subseteq U$, enlarge U by the terms in the order ideal generated by this set and continue with step (4).*
(*6*) *If $W \neq \emptyset$, append W to V, replace F by F^+, and continue with step (3).*
(*7*) *Let $\mathcal{O} = U \setminus \mathrm{LT}_\sigma(V)$. If $\partial\mathcal{O} \not\subseteq U$, replace U by U^+ and continue with step (3).*
(*8*) *Apply the Final Reduction Algorithm and return the set $G = \{g_1, \ldots, g_\nu\}$ it computes.*

The term ordering σ in this algorithm is of a purely auxiliary nature. It is merely used to guide the computation and to make sure that step (*7*) yields an order ideal. It is possible to replace it by other rules guiding the computation. However, we note that in this case one has to either prove that the new rule produces an order ideal $\mathcal{O}$ or modify the computation so that it backtracks some steps if necessary.

Example 7.7. Consider the ideal $I = \langle f_1, \ldots, f_6 \rangle$ in $P = \mathbb{F}_2[x, y, z]$ of Example 6.4, i.e. let $f_1 = xy+xz+1$, $f_2 = xz+yz+z$, $f_3 = xy+xz+y+1$, $f_4 = x^2+x$, $f_5 = y^2+y$, and $f_6 = z^2+z$. Let us follow the steps of the BBA.

(*1*) Let $U = \mathbb{T}^3_{\leq 2}$ and $\sigma = \texttt{DegRevLex}$.
(*2*) Interreduction yields the basis $V = \{f_1, f_2, \tilde{f}_3, f_4, f_5, f_6\}$ with $\tilde{f}_3 = y$.
(*3*) We interreduce the 24 polynomials $V \cup xV \cup yV \cup zV$ and get $V \cup W' = V \cup \{x+1,\ z+1,\ yz,\ x^3+1,\ x^2y,\ xy^2,\ y^3,\ x^2z+1,\ xyz,\ y^2z,\ xz^2+1,\ yz^2,\ z^3+1\}$.
(*4*) We have $W = \{x+1,\ z+1,\ yz\}$.
(*6*) Let $V = \{f_1, f_2, \tilde{f}_3, f_4, f_5, f_6, f_7, f_8, f_9\}$ with $f_7 = x+1$, $f_8 = z+1$, and $f_9 = yz$.
(*3*) We interreduce the 36 polynomials in $V \cup xV \cup yV \cup zV$ and get $V \cup W' = V \cup \{x^3+1,\ x^2y,\ xy^2,\ y^3,\ x^2z+1,\ xyz,\ y^2z,\ xz^2+1,\ yz^2,\ z^3+1\}$.
(*4*) We have $W = \emptyset$.
(*7*) We let $\mathcal{O} = U \setminus \mathrm{LT}_\sigma(V) = \{1\}$ and obtain $\partial\mathcal{O} = \{x, y, z\} \subseteq U$.

(*8*) The Final Reduction Algorithm returns $G = \{x + 1, y, z + 1\}$.

Thus we see that $(1, 0, 1)$ is the only zero of I.

A preliminary implementation of the BBA is available in the ApCoCoA library (see [3]). Since it works for general 0-dimensional ideals, it is not optimized for performing algebraic attacks. In the following remarks we collect a number of possible improvements.

Remark 7.8. After choosing a number $N > 0$, we can replace step (*8*) of the BBA by the following instruction.

(*8′*) If $\partial\mathcal{O} \not\subseteq U$, enlarge U by the order ideal generated by $\partial\mathcal{O}$. Every N^{th} time this is done, replace U by U^+ instead.

The resulting algorithm still computes an order ideal $\mathcal{O}$ and an $\mathcal{O}$-border basis of I. In general, it keeps the size of the computational universe U even smaller than the BBA, but it may require more iterations. In other words, we are sacrificing time efficiency for gaining space efficiency.

Note that the second part of step (*8′*) has been introduced as a safeguard since we have not been able to prove termination without it. In practice, there is no problem and N can be chosen quite large.

Remark 7.9. In the BBA we may try to reduce the number of iterations of the loop in steps (*3*)–(*6*) by replacing them with the following instructions.

(*3′*) Enlarge F to a vector space F^{++} by adding for each basis element v of V all products tv such that $t \in \mathbb{T}^n$ and $\mathrm{LT}_\sigma(tv) \in U$. Compute a basis extension $V \cup W'$ of V such that $V \cup W'$ is a K-basis of F^{++} with pairwise distinct leading terms.

(*4′*) Let $W = \{w \in W' \mid \mathrm{LT}_\sigma(w) \in U\}$.

(*5′*) If $\bigcup_{w \in W} \mathrm{Supp}(w) \not\subseteq U$, enlarge U by the terms in the order ideal generated by this set and continue with step (*4′*).

(*6′*) If $W \neq \emptyset$, append W to V, replace F by F^{++}, and continue with step (*3*).

Notice that we are imitating J. Ding's mutant idea here: if, after a certain iteration, one of the elements of V has a particularly small degree, it is used to generate more polynomials in the next iteration. Moreover, we can try to reduce the work for the next iterations by saving the polynomials in $W' \setminus W$ because it is quite likely that they can be reused in the next iteration.

Remark 7.10. Suppose that $f_1, \ldots, f_m$ define a system of equations for which we expect several solutions, and we are mostly interested in the elimination polynomials $I \cap K[x_i]$. In this case we can choose a number $N > 0$ and modify the search heuristics in the BBA by replacing step (*8*) as follows.

(*8′*) If $\partial\mathcal{O} \not\subseteq U$, enlarge U by the order ideal generated by $\{x_i^{\alpha_i} \mid 1 \leq i \leq n, \alpha_i = \min\{j \mid x_i^{\alpha_j} \notin U\}\}$. Every N^{th} time this is done, replace U by U^+ instead.

Again the last part of step (*8′*) has to be introduced in order to secure termination. Clearly, the computational universe will now tend to grow faster in the direction of the *axes* $\{x_i^j \mid j \geq 0\}$. This increases our chances of finding the elimination polynomials sooner.

In the spirit of these remarks, it is obviously possible to generate a number of further variations of the BBA which have the potential to speed up algebraic attacks considerably while at the same time avoiding a quick exhaustion of the available memory.

As a final note, we point out that the vector space basis extension computed in step (*3*) of the BBA can be found with the help of all the sparse linear algebra techniques which lie at the heart of the F4 algorithm. In this sense, the BBA is able to combine several techniques for improving the best known solution algorithms for polynomial systems and deserves further efficient implementation and experimentation.

8. The Integer Programming Attack

In this section we will restrict our attention to algebraic attacks based on polynomial systems defined over $\mathbb{F}_2$. Although the generalization to other finite base fields is straightforward, we want to concentrate on the fundamental principles in the most important case. The task of solving a polynomial system $f_1 = \cdots = f_m = 0$ with $f_1, \ldots, f_m \in \mathbb{F}_2[x_1, \ldots, x_n]$ can be rephrased as follows: Find a tuple $(a_1, \ldots, a_n) \in \{0,1\}^n$ such that

$$\begin{aligned} F_1(a_1, \ldots, a_n) &\equiv 0 \pmod 2 \\ &\vdots \\ F_m(a_1, \ldots, a_m) &\equiv 0 \pmod 2 \end{aligned}$$

where $F_i \in \mathbb{Z}[x_1, \ldots, x_n]$ is the canonical representative of f_i. Thus we are looking for an integer solution $(a_1, \ldots, a_n)$ of this system which satisfies $0 \leq a_i \leq 1$. This formulation suggests to linearize the system and to apply an Integer Programming (IP) algorithm for finding a solution satisfying the stated bounds. The following proposition turns this idea into an effective algorithm.

Proposition 8.1 (The Integer Programming Attack). *Let $f_1, \ldots, f_m \in P = \mathbb{F}_2[x_1, \ldots, x_n]$. Then the following instructions define an algorithm which computes a tuple $(a_1, \ldots, a_n) \in \{0,1\}^n$ which defines a zero of the 0-dimensional radical ideal $I = \langle f_1, \ldots, f_m, x_1^2 + x_1, \ldots, x_n^2 + x_n \rangle$.*

(*1*) *Reduce $f_1, \ldots, f_m$ modulo the field equations, i.e. make their support square-free. For $i = 1, \ldots, m$, let S_i be the set of terms of degree ≥ 2 in f_i and $s_i = \#\operatorname{Supp}(f_i)$.*

(*2*) *For $i = 1, \ldots, m$, introduce a new indeterminate k_i and write down the linear inequality $K_i : \; k_i \leq \lfloor s_i/2 \rfloor$.*

(3) *For every $t_j \in S_i$, introduce a new indeterminate y_{ij}. For $i = 1, \ldots, m$, write $f_i = \sum_j t_j + \ell_i$ where the sum extends over all j such that $t_j \in S_i$ and where $\ell_i \in P_{\leq 1}$. Form the linear equation $F_i : \sum_j y_{ij} + \ell_i - 2\,k_i = 0$.*

(4) *For $i \in \{1, \ldots, m\}$ and $t_j \in S_i$, write $t_j = x_{j_1} \cdots x_{j_r}$ with $1 \leq j_1 < \cdots < j_r \leq n$. Form the linear inequalities $Y_{ij} : \; y_{ij} - x_i \leq 0$ and $Z_{ij} : \; -y_{ij} + x_{j_1} + \cdots + x_{j_r} - r + 1 \leq 0$.*

(5) *For all $i \in \{1, \ldots, m\}$, let $X_i : \; x_i \leq 1$.*

(6) *Choose a linear polynomial $C \in \mathbb{Q}[x_i, y_{ij}, k_i]$ and use an IP solver to find the tuple of natural numbers (a_i, b_{ij}, c_i) which solves the system of linear equations and inequalities $\{K_i,\, F_i,\, Y_{ij},\, Z_{ij},\, X_i\}$ and minimizes C.*

(7) *Return $(a_1, \ldots, a_n)$ and stop.*

Proof. Since we are looking for natural numbers a_i for which X_i holds, we have $a_i \in \{0, 1\}$. Similary, we have $b_{ij} \in \{0, 1\}$ by X_i and Y_{ij}. Moreover, if $t_j \in S_i$ and if one of the numbers $a_{j1}, \ldots, a_{j_r}$ is zero then Y_{ij} implies $b_{ij} = 0$. On the other hand, if $a_{j1} = \cdots = a_{jr} = 1$ then Z_{ij} implies $b_{ij} \geq 1$. Altogether, this means that b_{ij} equals $a_{j_1} \cdots a_{j_r}$, the value of t_j at $(a_1, \ldots, a_n)$.

Next it follows from F_i that $f_i(a_1, \ldots, a_n) = 2\,k_i$ is an even number, and K_i is nothing but the trivial bound for k_i implied by the size of the support of f_i. In this way the solutions of the IP problem correspond uniquely to the tuples $(a_1, \ldots, a_n) \in \{0, 1\}^n$ which satisfy the above reformulation of the given polynomial system. □

In this proposition we have not taken any advantage of the possibility to choose the *cost function* C. Obviously, the number of additional indeterminates y_{ij} (and k_i) we have to introduce depends on the sparsity of the system defined by $f_1, \ldots, f_m$. For systems with few quadratic or higher degree terms, even a straightforward, non-optimized implementation yields satisfactory results, as our next example shows.

Example 8.2. Given the CTC ("Courtois Toy Cipher") cryptosystem introduced in [15] and a plaintext – ciphertex pair, we construct an overdetermined algebraic system of equations in terms of the indeterminates representing key bits and certain intermediate quantities. The task is to solve the system for the key bits. The size of the system depends mainly on two parameters: the number b of simultaneous S-boxes and the number N of encryption rounds used. In the following table we collect the sizes of the resulting polynomial systems over $\mathbb{F}_2$ and compare the timings for their solution with the `GBasis5(...)` command of CoCoA(cf. [13]) and with the GLPK package (cf. [29]) applied to the IP problem of Prop. 8.1.

In this table t denotes the total number of non-linear terms in each system. The timings were obtained on a small laptop with a 2.0 GHz processor and 2 GB of RAM. If possible, the systems were constructed to have a unique solution, as reported in the last column of the table. The construction of the polynomials

CTC(b,N)	n	m	t	time GBasis5	time GLPK	sol. unique?
CTC(2,2)	54	98	60	0.3 s	0.2 s	yes
CTC(2,3)	78	144	90	2.0 s	1.0 s	yes
CTC(3,2)	81	147	90	2.8 s	2.0 s	yes
CTC(3,3)	117	216	135	∞	12.7 s	yes
CTC(3,4)	153	285	180	∞	85.8 s	no

Table 8.1: Algebraic attacks at the CTC cryptosystem

systems used the implementation of the CTC cryptosystem in [35]. The symbol ∞ indicates a running time of more than an hour.

As one can see from the table, for very sparse systems the running time of the IP attack compare favorably to the running times of a Gröbner basis attack. Even for examples involving many indeterminates, such as CTC(3,4), the above timings compete with individually tailored Gröbner basis methods, such as the ones reported in [1].

When the non-linear part of the polynomial system $f_1 = \cdots = f_m = 0$ is less sparse, the effectiveness of the IP attack decreases rapidly. For example, in an instance of the HFE cryptosystem with $m = 50$ equations in $n = 25$ indeterminates, the fact that there are $t = 325$ quadratic terms means that the method of Prop. 8.1 is much slower that the Gröbner basis timings given in [42].

In view of this example it is clear that the IP attack is very sensitive to the number of additional indeterminates we have to introduce. For many symmetric cryptosystems, the main non-linear contribution to this polynomial system is derived from the S-box equations. For small S-boxes (i.e. S-boxes involving only a small number of input and output bits), we can try to improve the IP attack as follows.

Remark 8.3. Let $\sigma : \mathbb{F}_2^d \longrightarrow \mathbb{F}_2^d$ be a boolean function representing a non-linear part of an encryption map, e.g. an S-box. Now we proceed as follows.

(1) Form the graph $\Gamma = \{(p, \sigma(p)) \mid p \in \mathbb{F}_2^d\}$ of σ in $\mathbb{F}_2^{2d}$. Let $\widetilde{\Gamma} \subset \{0,1\}^{2d}$ be the canonical set of representatives of Γ.

(2) Using a standard convex hull computation (e.g. the algorithm of Avis and Fukuda, cf. [6]), we calculate a set S of linear inequalities of the form $a_{i1}x_1 + \cdots + a_{i\,2d}x_{2d} \leq b_i$ with $a_{ij}, b_i \in \mathbb{Z}$ such that $\widetilde{\Gamma}$ is the set of integer solutions of S.

Then we can find a preimage of a point $(q_1, \ldots, q_d) \in \mathbb{F}_2^d$ under σ by substituting the canonical representative of q_i for x_{d+i} in S and solving the resulting system of linear inequalitites.

The time and memory requirements of this procedure are known to grow approximately as $(2^d)^{\lfloor d/2 \rfloor}$. Thus it is only feasible if d is small. Let us apply it in the

case of the CTC cryptosystem where we have $d = 3$.

Example 8.4. An S-box of the CTC cryptosystem is given by the map $\sigma : \mathbb{F}_2^3 \longrightarrow \mathbb{F}_2^3$ with $\sigma(0,0,0) = (1,1,1)$, $\sigma(0,0,1) = (1,1,0)$, $\sigma(0,1,0) = (0,0,0)$, $\sigma(0,1,1) = (1,0,0)$, $\sigma(1,0,0) = (0,1,0)$, $\sigma(1,0,1) = (1,0,1)$, $\sigma(1,1,0) = (0,0,1)$, and $\sigma(1,1,1) = (0,1,1)$. The the set of representatives $\widetilde{\Gamma}$ of the graph of σ is the set of integral solutions of the system

$$\begin{aligned}
x_1 + x_3 - x_4 - x_5 - x_6 &\leq 0\\
-2x_1 + x_3 - x_1 - x_5 + 2x_6 &\leq 0\\
4x_1 + 3x_2 - 2x_3 + 5x_4 + 2x_5 - x_6 &\leq 6\\
-2x_1 - 3x_2 + x_3 - 4x_4 - x_5 + 2x_6 &\leq -3\\
x_1 + 3x_2 - 2x_3 + 2x_4 + 2x_5 - x_6 &\leq 3\\
x_1 - 2x_3 + 2x_4 - x_5 - x_6 &\leq 0\\
-x_2 + 2x_3 + 2x_4 - x_5 - x_6 &\leq -1\\
3x_1 - x_2 + 2x_3 - 2x_4 - x_5 + 2x_6 &\leq -1\\
3x_1 + 2x_2 - x_3 + 4x_4 + 2x_5 - x_6 &\leq 5\\
-3x_1 - 4x_2 + 2x_3 - 5x_4 - x_5 + 2x_6 &\leq -4\\
2x_2 - x_3 + x_4 + 2x_5 - x_6 &\leq 2\\
-x_2 - x_3 + x_4 - x_5 - x_6 &\leq -1.
\end{aligned}$$

When available, e.g. in the case of the CTC cryptosystem mentioned above, this method should provide a substantial speed-up of the IP attack. For a precise determination of its pros and contras, further experiments are needed.

At this point we have arrived at the end of the first leg of our journey through the land of algebraic attacks. By comparing the different approaches, we can already gain a better understanding of the relative merits and the true difficulties in attacking cryptosystems via polynomial system solving. In the second part of this paper, we will also apply methods coming from mathematical logic, numerical analysis, linear and multilinear algebra, classical algebraic geometry, and some ad-hoc tricks. Hence the final discussion of the landscape of algebraic attacks will be delayed until we have completed the voyage. Meanwhile, the readers are cordially invited to try experimenting themselves: uncovering actual encrypted messages by applying your favorite mathematical tools is fun!

Acknowledgements. First and foremost, I would like to thank J. Ding (University of Cincinnati) and S. Pokutta (Technische Universität Darmstadt) for deep and valuable discussions on the subjects of this paper. Their input was instrumental in bringing the attacks described in Sections 7 and 8, respectively, to fruition. I am indebted to my students J. Brandt, J. Limbeck and E. Ullah for implementing various software packages which aided the experimentation underlying several of the attacks. Moreover, I thank G. Rosenberger for his encouragement and useful suggestions. The idea for this

article was born during a stay at Fairfield University for a conference in honor of Ben Fine. I am very grateful for the generous hospitality I experienced there.

References

[1] M. Albrecht: Algebraic Attacks on the Courtois Toy Cipher, Diploma Thesis, Universität Bremen (2006).

[2] M. Albrecht, C. Cid: Algebraic techniques in differential cryptoanalysis, in: Fast Software Encryption (Leuven, 2009), O. Dunkelmann (ed.), Lect. Notes Comput. Sci. 5665, Springer, Berlin (2009) 193–208.

[3] The ApCoCoA Team: ApCoCoA: Approximate Computations in Commutative Algebra, available at http://www.apcocoa.org .

[4] G. Ars, J.-C. Faugère: An algebraic cryptanalysis of nonlinear filter generators using Gröbner bases, Rapport de Recherche Inria, HAL-CCSd-CNRS 2003, available at http://hal.ccsd.cnrs.fr/docs/00/07/18/48/PDF/RR-4739.pdf .

[5] G. Ars, J.-C. Faugère, H. Imai, M. Kawazoe, M. Sugita: Comparison between XL and Gröbner basis algorithms, in: Advances in Cryptology – ASIACRYPT 2004 (Jeju Island, 2004), P. J. Lee (ed.), Lect. Notes Comput. Sci. 3329, Springer, Berlin (2004) 338–353.

[6] D. Avais, K. Fukuda: A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra, Discrete Comput. Geom. 8 (1992) 295–313.

[7] L. Bettale, J.-C. Faugère, L. Perret: Cryptanalysis of the TRMS signature scheme of PKC'05, in: Progress in Cryptology – AFRICACRYPT 2008 (Casablanca, 2008), S. Vaudenay (ed.), Lect. Notes Comput. Sci. 5023, Springer, Berlin (2008) 143–155.

[8] L. Bettale, J.-C. Faugère, L. Perret: Hybrid approach for solving multivariate systems over finite fields, J. Math. Cryptol., to appear.

[9] M. Brickenstein, A. Dreyer: PolyBoRi: A framework for Gröbner basis computations with Boolean polynomials, J. Symb. Comput. 44 (2009) 1326–1345.

[10] B. Buchberger: Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal, Ph.D. Thesis, Universität Innsbruck (1965).

[11] J.-M. Chen, N.T. Courtois, B.-Y. Yang: On asymptotic security estimates in XL and Gröbner-basis related algebraic cryptoanalysis, in: Information and Communications Security (Malaga, 2004), J. Lopez, S. Qing, E. Okamoto (eds.), Lect. Notes Comput. Sci. 3269, Springer, Berlin (2004) 401–413.

[12] C. Cid, G. Leurent: An analysis of the XSL algorithm, in: Advances in Cryptology – ASIACRYPT 2005 (Chennai, 2005), B. Roy (ed.), Lect. Notes Comput. Sci. 3788, Springer, Berlin (2005) 333–352.

[13] The CoCoA Team: CoCoA: A System for doing Computations in Commutative Algebra, available at http://cocoa.dima.unige.it .

[14] N. T. Courtois: Fast algebraic attacks on stream ciphers with linear feedback, in: Advances in Cryptology – CRYPTO 2003 (Santa Barbara, 2003), D. Boneh (ed.), Lect. Notes Comput. Sci. 2729, Springer, Berlin (2003) 176–194.

[15] N. T. Courtois: How fast can be algebraic attacks on block ciphers, in: Symmetric Cryptography (Dagstuhl, 2007), E. Biham et al. (ed.), Dagstuhl Sem. Proc. 07021, available at http://drops.dagstuhl.de/opus/volltexte/2007/1013 .

[16] N. T. Courtois, G. V. Bard: Algebraic crytoanalysis of the Data Encryption Standard, in: Cryptography and Coding (Cirencester, 2007), S. D. Galbraith (ed.), Lect. Notes Comput. Sci. 4887, Springer, Berlin (2007) 152–169.

[17] N. T. Courtois, A. Klimov, J. Patarin, A. Shamir: Efficient algorithms for solving overdefined systems of multivariate polynomial equations, in: Advances in Cryptology – EUROCRYPT 2000 (Bruges, 2000), B. Preneel (ed.), Lect. Notes Comput. Sci. 1807, Springer, Berlin (2000) 392–407.

[18] N. T. Courtois, W. Meier: Algebraic attacks on stream ciphers with linear feedback, in: Advances in Cryptology – EUROCRYPT 2003 (Warsaw, 2003), E. Biham (ed.), Lect. Notes Comput. Sci. 2656, Springer, Berlin (2003) 345–359.

[19] N. T. Courtois, J. Pieprzyk: Cryptanalysis of block ciphers with overdefined systems of equations, in: Y. Zheng (ed.), Advances in Cryptology – ASIACRYPT 2002 (Queenstown, 2002), Lect. Notes Comput. Sci. 2501, Springer, Berlin (2002) 267–287.

[20] J.-M. Chen, B.-Y. Yang: Theoretical analysis of XL over small fields, in: Information Security and Privacy (Sydney, 2004), H. Wang, J. Pieprzyk, V. Varadharajan (eds.), Lect. Notes Comput. Sci. 3108, Springer, Berlin (2004) 277–288.

[21] C. Diem: The XL Algorithm and a conjecture from commutative algebra, in: P. J. Lee (ed.), Advances in Cryptology – ASIACRYPT 2004 (Jeju Island, 2004), Lect. Notes Comput. Sci. 3329, Springer, Berlin (2004) 323–337.

[22] J. Ding, J. Buchmann, M. S. Mohamed, W. S. Mohamed, R.-P. Weinmann: MutantXL, in: Proc. Symbolic Computation and Cryptography (Beijing, 2008) 16-22.

[23] J. Ding, J.E. Gower, D. Schmidt, C. Wolf, Z. Yin: Complexity estimates for the F_4 attack on the Perturbed Matsumoto-Imai cryptosystem, in: Cryptography and Coding (Cirencester, 2005), N. P. Smart (ed.), Lect. Notes Comput. Sci. 3796, Springer, Berlin (2005) 262–277.

[24] eSTREAM: ECRYPT stream cipher project, see http://www.ecrypt.eu.org /stream/ .

[25] J.-C. Faugère: A new efficient algorithm for computing Gröbner bases (F_4), J. Pure Appl. Algebra 139 (1999) 61–88.

[26] J.-C. Faugère: A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5), in: Proc. Conf. ISSAC 2002 (Lille, 2002), T. Mora (ed.), ACM Press, New York (2002) 75–83.

[27] J.-C. Faugère, A. Joux: Algebraic cryptanalysis of Hidden Field Equation (HFE) cryptosystems using Gröbner bases, in: Advances in Cryptology – CRYPTO 2003 (Santa Barbara, 2003), D. Boneh (ed.), Lect. Notes Comput. Sci. 2729, Springer, Berlin (2003) 44–60.

[28] J. M. Gash: On Efficient Computation of Gröbner Bases, Ph.D. Thesis, Indiana University, Bloomington (2008).

[29] GNU Linear Programming Kit, available at http://www.gnu.org/software/glpk /glpk.html .

[30] P. Hawkes, G. Rose: Rewriting variables: the complexity of fast algebraic attacks on stream ciphers, in: Adcances in Cryptology – CRYPTO 2004 (Santa Barbara, 2004), M. K. Franklin (ed.), Lect. Notes Comput. Sci. 3152, Springer, Berlin (2004) 390–406.

[31] A. Kehrein, M. Kreuzer: Computing border bases, J. Pure Appl. Algebra 205 (2006) 279–295.

[32] M. Kreuzer, L. Robbiano: Computational Commutative Algebra 1, Springer, Berlin (2000).

[33] M. Kreuzer, L. Robbiano: Computational Commutative Algebra 2, Springer, Berlin (2005).

[34] A. Kipnis, A. Shamir: Cryptanalysis of the HFE public key cryptosystem by relinearization, in: Advances in Cryptology – CRYPTO '99 (Santa Barbara, 1999), M. Wiener (ed.), Lect. Notes Comput. Sci. 1666, Springer, Berlin (1999) 19–30.

[35] J. Limbeck: Implementation und Optimierung algebraischer Angriffe, Diploma Thesis, Universität Passau (2008).

[36] B. A. LaMacchia, A. M. Odlyzko: Solving large sparse systems over finite fields, in: Advances in Cryptology – CRYPTO '90 (Santa Barbara, 1990), A. J. Menezes et al. (ed.), Lect. Notes Comput. Sci. 537, Springer, Berlin (1991) 109–133.

[37] M. S. Mohamed, W. S. Mohamed, J. Ding, J. Buchmann: MXL2: Solving polynomial equations over GF(2) using an improved mutant strategy, in: Post-Quantum Cryptography – PQCrypto 2008 (Cincinnati, 2008), J. Buchmann, J. Ding (eds.), Lect. Notes Comput. Sci. 5299, Springer, Berlin (2008) 203–215.

[38] P. L. Montgomery: A block Lanczos algorithm for finding dependencies over GF(2), in: Advances in Cryptology – EUROCRYPT'95 (Saint-Malo, 1995), L. Gouillou, J.-J. Quisquater (eds.), Lect. Notes Comput. Sci. 921, Springer, Berlin (1995) 106–120.

[39] B. Mourrain: A new criterion for normal form algorithms, in: Applied Algebra, Algebraic Algorithms and Error correcting Codes (Honolulu, 1999), M. Fossorier et al. (ed.), Lect. Notes Comput. Sci. 1719, Springer, Berlin (1999) 440–443.

[40] S. Murphy, M. Robshaw: Comments on the security of the AES and the XSL technique, Electron. Lett. 39 (2003) 26–38.

[41] C. Pomerance, J. W. Smith: Reduction of huge sparse matrices over finite fields via created catastrophes, Exp. Math. 1 (1992) 89–94.

[42] A. Steel: Gröbner basis timings page, available at http://magma.maths.usyd.edu/au/users/allan/gb/ .

[43] T. Stegers: Faugère's F5 Algorithm Revisited, Diploma Thesis, Technische Universität Darmstadt (2005).

[44] C. E. Shannon: Communication theory of secrecy systems, Bell Syst. Tech. J. 28 (1949) 656–715.

[45] M. Sugita, M. Kawazoe, H. Imai: Relation between XL algorithm and Gröbner Bases Algorithms, IEICE Trans. 89A (2006) 11–18.

[46] L. Xiao: Applicability of XSL attacks to block ciphers, Electron. Lett. 39 (2003) 1810–1811.