Computational and Algebraic Aspects of the Advanced Encryption Standard

Carlos Cid, Sean Murphy and Matthew Robshaw

Information Security Group,
Royal Holloway, University of London,
Egham, Surrey, TW20 0EX, U.K.
carlos.cid@rhul.ac.uk
s.murphy@rhul.ac.uk
m.robshaw@rhul.ac.uk

Abstract. The new Advanced Encryption Standard (AES) has been recently selected by the US government to replace the old Data Encryption Standard (DES) for protecting sensitive official information. Due to its simplicity and elegant algebraic structure, the choice of the AES algorithm has motivated the study of a new approach to the analysis of block ciphers. While conventional methods of cryptanalysis (e.g. differential and linear cryptanalysis) are usually based on a "statistical" approach, where an attacker attempts to construct statistical patterns through many interactions of the cipher, the so-called *algebraic attacks* exploit the intrinsic algebraic structure of a cipher. More specifically, the attacker expresses the encryption transformation as a set of multivariate polynomial equations and attempts to recover the encryption key by solving the system.
In this paper we consider a number of algebraic aspects of the AES, and examine a few computational and algebraic techniques that could be used in the cryptanalysis of cipher. We show how one can express the cipher as a very large, though surprisingly simple, system of multivariate quadratic equations over the finite field $\mathbb{F}_{2^8}$, and consider some approaches that can be used to solve this system.

1 Introduction

In 1997 the US National Institute of Standards and Technology (NIST) announced an open competition to select a substitute for the old 64-bit block, 56-bit key Data Encryption Standard (DES). The new encryption standard, called Advanced Encryption Standard (AES), should support block lengths of 128 bits and key lengths of 128, 192 and 256 bits. The AES is to replace DES for securing "sensitive but unclassified" information and systems. Additionally, the AES is likely to be adopted as a standard by the private sector, in particular in commerce and the banking sector. In 2000 NIST announced the Belgian candidate *Rijndael* as the winner.

Rijndael has a simple and elegant structure. It has been designed to offer strong resistance against *known attacks*, in particular differential and linear cryptanalysis, while enabling efficient implementation on different platforms. We refer to [14] for a very good overview of the development of the AES.

Rijndael has also a highly algebraic structure. The cipher round transformations are based on operations on the finite field $\mathbb{F}_{2^8}$. This has led to a growing interest in applying algebraic techniques in the cryptanalysis of block ciphers in recent times. One reason is that conventional methods of cryptanalysis (e.g. differential and linear cryptanalysis) are generally based on a "statistical" approach: the attacker attempts to construct probabilistic characteristics through as many rounds of the cipher as possible, in order to distinguish the cipher from a random permutation. Most modern ciphers have been designed with these attacks in mind, and therefore do not generally have their security affected by them. Additionally, due to the very nature of the attacks, the complexity usually grows *exponentially* with the number of rounds, ensuring that such attacks become rapidly impractical.

In contrast, algebraic attacks exploit the intrinsic algebraic structure of a cipher. More specifically, the attacker expresses the encryption transformation as a (large) set of multivariate polynomial equations, and subsequently attempts to solve such a system to recover the encryption key.

Algebraic attacks open new perspectives in the cryptanalysis of block ciphers, as they can be seen as a "structural" attack on the cipher, and one if successful, might not be easily avoided by simply increasing the number of rounds. In particular, there are indications that the complexity of such attacks might not grow exponentially with the number of rounds [6].

While in theory most modern block ciphers can be fully described by a system of multivariate polynomials over a finite field, for the majority of the cases such systems prove to be just too complex for any practical purpose. Yet there are a number of recent designed ciphers that present a highly algebraic structure, and could therefore be more vulnerable to algebraic attacks [1]. In the particular case of the AES, Courtois and Pieprzyk have shown in [6] how one can express the encryption operation as a large, sparse, overdefined system of multivariate quadratic equations over $\mathbb{F}_2$. In the same paper they propose a method called XSL, which could provide an efficient way to recover the encryption key for certain ciphers.

Around the same time, Murphy and Robshaw [17] showed how to express the AES encryption as a far simpler system of equations over $\mathbb{F}_{2^8}$. If XSL is in fact a valid method, this system should be faster to solve than the original one over $\mathbb{F}_2$, and in theory, could provide an attack more efficient than exhaustive search of the key, which would be devastating for the AES.

In this paper we will recall some of the work that has been developed so far in the algebraic cryptanalysis of the AES and consider some approaches that could be applied to solve the corresponding system of multivariate equations by using computational algebraic geometry and commutative algebra techniques.

2 The Basic Structure of the AES

Rijndael is a key-iterated block cipher, alternating key-independent round transformations and key addition. We refer to [9] for a full description of the cipher. Here we consider the basic version of the AES, which encrypts a 16-byte block using a 16-byte key with 10 encryption rounds. The treatment of the representation of the AES given in this section and in Sections 3 and 4 is largely based in [17].

The input to the AES round function can be viewed as a rectangular array of bytes or, equivalently, as a column vector of bytes, known as the *state* (Figure 1). Throughout the encryption process this byte-structure is fully respected.

$$\mathbf{a} = \begin{array}{|c|c|c|c|}\hline a_{00} & a_{01} & a_{02} & a_{03} \\ \hline a_{10} & a_{11} & a_{12} & a_{13} \\ \hline a_{20} & a_{21} & a_{22} & a_{23} \\ \hline a_{30} & a_{31} & a_{32} & a_{33} \\ \hline \end{array} = (a_{00}, \ldots, a_{30}, a_{01}, \ldots, a_{31}, a_{02}, \ldots, a_{33})^T$$

Fig. 1. The AES-128 state.

In the AES, each byte is regarded as an element of the field

$$\mathbb{K} = \frac{\mathbb{F}_2[x]}{< f(x) >} \cong \mathbb{F}_{2^8} \ ,$$

where $f(x) \in \mathbb{F}_2[x]$ is the irreducible polynomial $x^8 + x^4 + x^3 + x + 1$.

The AES specification defines a round in terms of the following three transformations[1]:

1. **The AES S-Box**. This is the only non-linear operation of the cipher. The value of each byte in the array is substituted according to a table look-up. This table look-up is the combination of three transformations:

[1] The first and last rounds have a slightly different (but related) form.

(a) The input w is mapped to $x = w^{(-1)}$, where $w^{(-1)}$ is defined by

$$w^{(-1)} = w^{254} = \begin{cases} w^{-1} & w \neq 0 \\ 0 & w = 0 \end{cases}$$

Thus the "AES inversion" is identical to the standard field inversion in $\mathbb{K}$ for non-zero field elements, with $0^{(-1)} = 0$.

(b) The intermediate value x is regarded as a $\mathbb{F}_2$-vector of dimension 8 and transformed using an (8×8) $\mathbb{F}_2$-matrix L_A. The transformed vector $L_A \cdot x$ is then regarded in the natural way as an element of $\mathbb{K}$.

(c) The output of the AES S-Box is $(L_A \cdot x) + d$, where d is a constant element of $\mathbb{K}$.

2. **The AES linear diffusion (mixing) layer.**
   (a) Each row of the array is rotated by a certain number of byte positions. This operation is called `ShiftRow`.
   (b) Each column $\mathbf{y}$ of the array is considered as a vector of $\mathbb{K}^4$, and is transformed into the column $C \cdot \mathbf{y}$, where C is a (4×4) $\mathbb{K}$-matrix. This operation is called `MixColumn`.
3. **The AES subkey addition.** Before encryption, the original key is expanded into 11 *round subkeys*, each having 16 bytes. Then following the diffusion layer, each byte of the array is added (in $\mathbb{K}$) to a byte from the corresponding array of round subkeys.

2.1 Remarks on the Structure of the AES

In [16] Murphy and Robshaw make the following remarks about the round structure of the AES:

- The additive constant $d \in \mathbb{K}$ in the AES S-box can be removed by incorporating it within a (slightly) modified key schedule.
- The linear transformation L_A in the S-Box can be now viewed as part of an augmented linear diffusion layer. This seems to give a more natural presentation of the round function as:
  - a *S-Box Layer* : inversion in $\mathbb{F}_{2^8}$;
  - a *Linear Diffusion Layer* : $\mathbb{F}_2$-linear transformation M;
  - a *Subkey Layer* : addition of the modified round subkey.
- The augmented linear layer M is very structured ($M^{16} = Id$), with some quite interesting properties [16].

They also note that, although these particular properties may offer little advantage to conventional cryptanalysis, it is not clear whether one can find other ways to combine the rich structure of the diffusion layer and the highly structured inverse map. In particular, by simply re-writing the round transformation as above, it seems already that some of the original design criteria for the cipher have not been met (e.g. an S-Box without fixed and "opposite-fixed" points [9]).

3 The Big Encryption System (BES)

One of the main difficulties in exploring the properties described above in the algebraic cryptanalysis of the AES is the existence of operations in two distinct fields ($\mathbb{F}_2$ and $\mathbb{F}_{2^8}$). To address the conflict between these two operations in the AES, a new iterated block cipher was introduced in [17]. The *Big Encryption System (BES)* operates on 128-byte blocks with 128-byte keys. BES has a very simple algebraic structure: one round of the cipher consists of inversion, matrix multiplication and key addition, all operations over $\mathbb{F}_{2^8}$.

Both the AES and the BES use a *state vector* of bytes, which is transformed by the basic operations within a round. The state spaces of the AES and the BES are the vector spaces $\mathbf{A} = \mathbb{K}^{16}$ and $\mathbf{B} = \mathbb{K}^{128}$, respectively.

The relation between the AES and the BES is established by using the *vector conjugate mapping* ϕ from $\mathbb{K}$ to $\mathbb{K}^8$, which is defined by

$$\widetilde{\mathbf{a}} = \phi(a) = \left(a^{2^0}, a^{2^1}, a^{2^2}, a^{2^3}, a^{2^4}, a^{2^5}, a^{2^6}, a^{2^7}\right).$$

This definition extends in the obvious way to a vector conjugate mapping ϕ from $\mathbb{K}^n$ to a subset of $\mathbb{K}^{8n}$.

We can therefore use the map ϕ to embed an element of the AES state space $\mathbf{A}$ into the BES state space $\mathbf{B}$. We define

$$\mathbf{B_A} = \phi(\mathbf{A}) \subset \mathbf{B}$$

to be the embedded image of the AES state space in the BES state space. The subspace $\mathbf{B_A}$ is an additively closed set that also preserves inverses. In fact, BES is defined in such way that the diagram in Figure 2 is commutative.

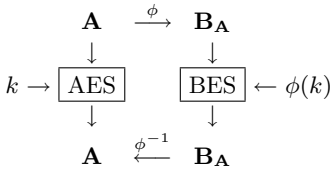

Fig. 2. The relationship between the AES and the BES.

3.1 The Structure of the BES

We refer to [17] for the full description of the BES cipher. The basic operations over $\mathbb{F}_{2^8}$ are the same as for the AES: subkey addition, row and column operations, S-Box inversion.

The main problem appears to come from the AES S-Box $\mathbb{F}_2$-linear operation, as there is no easy way to represent it as a matrix multiplication for the AES. However, there is a simple matrix representation of this operation for the BES.

In the AES specification, the S-Box $\mathbb{F}_2$-linear operation is defined by considering $\mathbb{K} = \mathbb{F}_{2^8}$ as the vector space $(\mathbb{F}_2)^8$. To accomplish this change, the natural mapping $\psi : \mathbb{F}_{2^8} \rightarrow (\mathbb{F}_2)^8$ is used in the AES. The componentwise AES $\mathbb{F}_2$-linear operation $f : \mathbb{K} \rightarrow \mathbb{K}$ is then defined as $f(a) = \psi^{-1}(L_A(\psi(a)))$ for $a \in \mathbb{K}$. It is the need for the maps ψ and ψ^{-1} that complicates the algebraic analysis of the AES.

However, since f is $\mathbb{F}_2$-linear over $\mathbb{F}_{2^8}$, it can be represented as a *linearized* polynomial on $\mathbb{F}_{2^8}$, i.e. it can described by a linear combination of conjugates. In fact we have

$$f(a) = \sum_{k=0}^{7} \lambda_k a^{2^k} \text{ for } a \in \mathbb{K},$$

where

$$(\lambda_0, \lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6, \lambda_7) = (\texttt{05}, \texttt{09}, \texttt{f9}, \texttt{25}, \texttt{f4}, \texttt{01}, \texttt{b5}, \texttt{8f}).$$

Thus we have that the $\mathbb{F}_2$-linear operation from the AES S-box can now be defined in the BES using an (8×8) $\mathbb{F}_{2^8}$-matrix. This matrix replicates the AES action of the $\mathbb{F}_2$-linear map on the first byte of a vector conjugate set while ensuring that the property of *vector conjugacy* is preserved on the remaining bytes.

The Round Function of the BES. Given as input to the BES round function the array $\mathbf{b} \in \mathbf{B}$ and subkey $(\mathbf{k}_B)_i \in \mathbf{B}$, then, following a rearrangement of the operations as described in Section 2.1, the BES round function is given by

$$\text{Round}_B(\mathbf{b}, (\mathbf{k}_B)_i) = M_B \cdot (\mathbf{b}^{(-1)}) + (\mathbf{k}_B)_i,$$

where M_B is a (128×128) $\mathbb{F}_{2^8}$-matrix performing linear diffusion within the BES. Similarly to the AES, the matrix M_B has a surprisingly simple structure (a sparse matrix, with minimum polynomial $(x+1)^{15}$).

Furthermore, if the input to the AES round function are $\mathbf{a} \in \mathbf{A}$ and subkey $(\mathbf{k}_A)_i \in \mathbf{A}$, then we have

$$\mathrm{Round}_A(\mathbf{a}, (\mathbf{k}_A)_i) = \phi^{-1}\left(\,\mathrm{Round}_B\left(\,\phi(\mathbf{a}), \phi((\mathbf{k}_A)_i)\,\right)\right).$$

In summary, the BES is a 128-byte block cipher consisting entirely of simple algebraic operations over $\mathbb{K} = \mathbb{F}_{2^8}$, which when restricted to the subspace $\mathbf{B}_\mathbf{A} = \phi(\mathbf{A})$, provides an alternative description of the AES. This relationship between both ciphers may well provide new ways for the cryptanalysis of the AES.

4 Algebraic Cryptanalysis of the AES

Given the rich algebraic structure of the AES, it seems natural that one would attempt to explore it in the cryptanalysis of the cipher. In [12] the authors describe how one can express the AES encryption transformation in a compact algebraic formula, which contains around 2^{50} terms. However, they acknowledge that it seems very unlikely that an attacker could take advantage of such algebraic description to mount a *practical attack*.

Later Courtois and Pieprzyk [6] exhibit a large system of multivariate equations whose solution would recover the AES encryption key. They explore the fact that the only non-linear component of the algorithm (the S-Box) is based on the inverse map on a finite field, and were able to obtain a small set of quadratic multivariate equations (in the input and output bits) that completely described the S-Box transformation. By combining all equations throughout the cipher, they were able to express the full encryption transformation as a large, sparse and overdefined system of multivariate quadratic equations over $\mathbb{F}_2$ (8000 quadratic equations with 1600 variables for the AES with 128-bit key).

In the same paper they propose a method called *XSL (eXtended Sparse Linearization)*, as an attempt to efficiently solve the system. XSL is one of a number of methods based on the well-known method of *linearization*. The XSL method attempts to take advantage of the particular properties of the system which arises from AES (overdefined and sparse). If valid, the XSL could provide an effective way to recover the encryption key.

Note that the system given in [6] is over $\mathbb{F}_2$, as the variables represent bits which arise during the running of the cipher. As we have shown, the representation of the AES as the BES with a restricted message and key space should give rise to a far simpler system of equations over $\mathbb{F}_{2^8}$. If XSL is indeed a valid method, this system should be faster to solve than the original one over $\mathbb{F}_2$.

4.1 A Simple Multivariate Quadratic System for the BES

Recall that the round function of the BES (and therefore essentially the AES) is given by

$$\mathbf{b} \mapsto M_B.\mathbf{b}^{-1} + (\mathbf{k}_B)_i.$$

Let us denote the plaintext and ciphertext by $\mathbf{p} \in \mathbf{B}$ and $\mathbf{c} \in \mathbf{B}$ respectively, and the state vectors before and after the i^{th} invocation of the inversion layer by $\mathbf{w}_i \in \mathbf{B}$ and $\mathbf{x}_i \in \mathbf{B}$ $(0 \leq i \leq 9)$ respectively. Also let $\mathbf{k}_i$ denote the i^{th} round subkey. Then the BES encryption can be described by the following system of equations:

$$\begin{aligned}
\mathbf{w}_0 &= \mathbf{p} + \mathbf{k}_0, \\
\mathbf{x}_i &= \mathbf{w}_i^{(-1)} && \text{for } i = 0, \ldots, 9, \\
\mathbf{w}_i &= M_B \mathbf{x}_{i-1} + \mathbf{k}_i && \text{for } i = 1, \ldots, 9, \\
\mathbf{c} &= M_B^* \mathbf{x}_9 + \mathbf{k}_{10}, &&
\end{aligned}$$

where M_B^* is a modified diffusion matrix (the final round in both BES and AES does not use the `MixColumn` operation).

If we denote the matrix M_B by (α) and the matrix M_B^* by (β), and represent the $(8j+m)^{\text{th}}$ component of $\mathbf{x}_i$, $\mathbf{w}_i$ and $\mathbf{k}_i$ by $x_{i,(j,m)}$, $w_{i,(j,m)}$ and $k_{i,(j,m)}$ respectively, we can consider the above equations componentwise, and thus obtain a collection of simultaneous multivariate quadratic

equations which fully describe a BES encryption. These are given for $j = 0, \ldots, 15$ and $m = 0, \ldots, 7$ by:

$$\begin{aligned}
&0 = w_{0,(j,m)} + p_{(j,m)} + k_{0,(j,m)}, \\
&0 = x_{i,(j,m)} w_{i,(j,m)} + 1 && \text{for } i = 0, \ldots, 9, \\
&0 = w_{i,(j,m)} + k_{i,(j,m)} + \textstyle\sum_{(j',m')} \alpha_{(j,m),(j',m')} x_{i-1,(j',m')} && \text{for } i = 1, \ldots, 9, \\
&0 = c_{(j,m)} + k_{10,(j,m)} + \textstyle\sum_{(j',m')} \beta_{(j,m),(j',m')} x_{9,(j',m')}.
\end{aligned}$$

Note that we assume that 0-inversions do not occur as part of the encryption or the key schedule. This assumption is true for approximately 53% of encryptions and 85% of 128-bit keys.

When we consider an AES encryption embedded in the BES framework, we obtain more multivariate quadratic equations because the embedded state variables of an AES encryption are in $\mathbf{B_A}$ and possess the conjugacy property. Thus we need to add the following equations to the system above:

$$\begin{aligned}
&0 = x^2_{i,(j,m)} + x_{i,(j,m+1)} && \text{for } i = 0, \ldots, 9, \\
&0 = w^2_{i,(j,m)} + w_{i,(j,m+1)} && \text{for } i = 0, \ldots, 9.
\end{aligned}$$

An AES encryption can therefore be described as an overdefined multivariate quadratic system with 5248 equations over $\mathbb{K} = \mathbb{F}_{2^8}$, of which 3840 are (extremely sparse) quadratic equations and 1408 are linear equations. These encryption equations comprise 7808 terms, made from 2560 state variables and 1408 key variables. Furthermore, the AES key schedule can be expressed as a similar system. In its most sparse form, the key schedule system has 2560 equations over $\mathbb{K}$, of which 960 are quadratic and 1600 are linear equations. These key schedule equations comprise 3308 terms made from the 2048 variables, of which 1408 are basic key variables and 640 are auxiliary variables. Note that one can reduce the sizes of the systems by using the linear equations to substitute for state and key variables, though the resulting system is slightly less sparse [17].

5 Potential Attack Techniques

Given the BES algebraic formulation, it is clear that an efficient method for the solution of this type of multivariate quadratic system would give a cryptanalysis of the AES with potentially very few plaintext-ciphertext pairs. While the problem of solving generic large systems of multivariate equations of degree greater than one over a finite field is known to be NP-complete, it is not entirely unlikely that a technique can be developed which exploits the particular algebraic structure of the AES and BES systems. Below we investigate a few approaches for solving such systems.

5.1 Linearization Methods

The method of *linearization* 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. In order to be able to apply the 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 LI equations so that one can apply the linearization method.

In [5] the authors propose an algorithm for solving systems of multivariate quadratic equations called *XL* (standing for *eXtended Linearization*). XL is a simple algorithm: if A is a system of m quadratic equations f_i in n variables over a field K, and $D \in \mathbb{N}$, one executes the following steps:

1. **Multiply:** Generate all the products $\prod_{j=1}^{k} x_{i_j} * f_i$ with $k \leq D - 2$;
2. **Linearize:** Consider each monomial of degree $\leq D$ as a new variable and perform Gaussian elimination on the system obtained in step 1;
3. **Solve:** Assume that step 2 yields at least one univariate equation. Solve this equation;
4. **Repeat:** Simplify the equations and repeat the process to find the values of the other variables.

The hope is that after few iterations the algorithm will yield a solution for the system.

In [5] the authors present some estimates for the complexity of the XL algorithm for random systems with $m \approx n$. In particular, they provide evidence that XL can solve randomly generated

overdefined systems of polynomial equations in subexponential time. However, one can show that there are cases where the algorithm might not work at all (for example, when the associated projective variety has positive dimension). Additionally, there has been strong evidence that some of the heuristics used were too optimistic [2]. The main discrepancy arises from the fact that one might overestimate the number of *linearly independent* equations generated by the algorithm. In any case, it is widely agreed that application of the XL algorithm against the polynomial system which arises from the AES (either over $\mathbb{F}_{2^8}$ or $\mathbb{F}_2$) does not give an efficient attack against the cipher.

Since the introduction of the XL method in 2000, a number of variants have been proposed in the attempt to exploit specific properties of the polynomial system, such as how overdefined the system is, or the order of the field[2]. Of particular relevance for the AES is the method proposed in [6] by Courtois and Pieprzyk.

XSL is based on the XL method, but uses the sparsity and specific structure of the equations to mount the attack; instead of multiplying the equations by all monomials of degree $\leq D-2$, in the XSL algorithm the equations are multiplied only by "carefully selected monomials" (we refer to [6] and its earlier version [7] for a full description of the method). While this has the intention to create less new terms when generating the new equations, it is not entirely clear the exact criteria used for selecting the monomials.

The system used in [6] to mount the attack has 8000 quadratic equations and 1600 variables, over $\mathbb{F}_2$ (the variables represent the input/output bits). Two attacks are described in [7]: the first one ignores the key schedule and therefore needs 11 known plaintext/ciphertext pairs (for the AES-128); the second attack uses the key schedule, and in theory could be mounted with a single known plaintext/ciphertext pair. In [6] it is claimed that the second XSL attack would have complexity of $\approx 2^{230}$ and $\approx 2^{255}$ when applied against the 128-bit and 256-bit AES, respectively. So the XSL attack would represent a (at least theoretical) successful attack against the 256-bit AES.

XSL Attack on the BES. As shown earlier, the AES $\mathbb{F}_{2^8}$-system derived from the BES is much simpler than the $\mathbb{F}_2$-system presented in [6]. In particular, it is far sparser. This would strongly suggest that the XSL attack is more suited to the BES system than to the original AES system.

Murphy and Robshaw consider in [18] the consequences of the XSL attack against the AES system derived from the BES. Using the estimates given in [6], they conclude that if XSL is in fact a *valid technique*, an AES key recovery might be possible with a work effort of about 2^{100} AES encryptions. This would clearly represent a successful attack against the AES-128.

Accuracy of the XSL Estimates. The main issue when considering XSL attacks (in fact, all the XL-based attacks) on the AES is how accurate the estimates for the number of *linearly independent* equations are. As explained above, there is evidence that some of the heuristics in the original XL paper were too optimistic. In fact, there is even more concern when considering the XSL method [3]. The method is based on a number of heuristics arguments, and although this might not invalidate the XSL technique entirely, it makes harder to consider whether the XSL attacks described in [6] work *as claimed*.

We have considered very small versions of BES, with reduced block length and number of rounds, and smaller field. We ran a few simulations with these versions, and it appears that the attacks do not work in the manner predicted in [6]. Again, while this might not invalidate the XSL technique, it could raise doubts on whether the method is generally applicable to the BES system.

Variants of the XSL Method. A possible approach to overcome some of the difficulties when applying the XSL method against the BES system is to use a similar idea to the one given in [4], where a variant of the XL method called *XLF* is proposed. XLF attempts to address some of the possible limitations of the XL method over $\mathbb{F}_{2^k}$, with $k > 1$. It does this by introducing k new variables for each variable x_i present in the system (the conjugates of x_i) and k new equations for each original equation (image of the Fröbenius automorphism on the equations). Furthermore, the quadratic equations relating variables with its conjugates are introduced.

[2] Fields of characteristic 2 are the most important for cryptographic applications

This method seems to have been based on the BES system, which has already the form above. But one interesting new step is the use of the relations between the variables and their conjugates to reduce the degree of terms generated by the algorithm. For example, given the (conjugate) relations $x_{00}^2 + x_{01} = 0$ and $y_{11}^2 + y_{12} = 0$, then when multiplying the equation $f : x_{00}y_{21} + y_{11}z_{01} = 0$ by the monomial $x_{00}y_{11}$, we would obtain the new equation

$$y_{11}x_{00} * x_{00}y_{21} + x_{00}y_{11} * y_{11}z_{01} = x_{01}y_{11}y_{21} + x_{00}z_{01}y_{12} = 0$$

This step corresponds to a simple reduction of f by the conjugate relations. However one must be careful when analysing the effect of introducing this new step to the XSL algorithm. In fact, the technique of reducing monomials by using relations in the base field has appeared in number of proposed variants of the XL algorithm, though it was not part of the original XL algorithm (no reduction appeared in the description of the algorithm in [5]).

So although this new step appears to improve the efficiency of XL-type algorithms for solving such multivariate quadratic systems over finite fields (and experiments in [4] seem to suggest so), more research is needed to determine whether such technique can improve the chances of the XSL attack against the BES system.

5.2 Gröbner Bases and Other Computational Algebra Techniques

Solving multivariate polynomial systems is a typical problem studied in Algebraic Geometry and Commutative Algebra. The classical algorithm for solving this type of problem is the Buchberger algorithm for calculating Gröbner Bases (see [8] for definitions and description of the algorithm). The algorithm generates a basis for the ideal derived from the set of equations, which can then be used to obtain the solutions.

The complexity of most algorithms used for calculating a Gröbner basis of an ideal is closely related to the total degree of the intermediate polynomials that are generated during the running of algorithm. In the worst case the Buchberger algorithm is known to run in double exponential time. One of the most efficient algorithms known, due to Faugère [10], appears to be single exponential. In any case, in practice it is widely believed that Gröbner Bases algorithms cannot be used for efficiently solving *generic* systems with more than a handful of variables (e.g. 15).

However, the type of systems which arise from cryptosystems are usually very structured, and therefore far from looking "random". For example, in [11] the authors exploit the algebraic properties of the private key to solve the first *Hidden Field Equations (HFE)* cryptosystem challenge, which consisted of a system with 80 equations and variables, by computing a Gröbner basis for the system.

Gröbner Bases of the BES System. As mentioned above, we have considered small versions of BES, with reduced block length and number of rounds, and smaller field. One of the resulting systems consisted of 112 equations and 64 variables. In our experiments, this system was successfully solved using off-the-shelf Computer Algebra package (Magma) in a couple of seconds. While this by no means guarantees the tractability of the problem of solving the BES system, it can indicate that solving the original BES system might not be as hard as one would expect.

While the behavior of Gröbner bases algorithms for generic systems is quite complicated, the BES system has a very regular structure. It can be considered as an "iterated" system of equations, with similar "sub-systems" repeated for every round. One could also use the transformation $\mathbf{x} \mapsto \mathbf{x}^{254}$ as the S-Box inversion to eliminate a number of variables (the BES system considered in Section 4 has the simplest form, with only quadratic and linear equations).

Furthermore, since the system includes the equations relating every variable with its conjugates, we have the following easy proposition:

Proposition 1. *The maximal degree of polynomials occurring in the computation of a Gröbner basis of a BES-type system with n variables is at most n.*

This is clearly an upper bound, and we expect that in practice the degrees are much lower. This fact, together with the particular structure of the system, can be exploited to infer more precise bounds for the complexity of the attack.

One can also seek alternatives for the use of the usual Gröbner bases algorithm. For example, the concept of *Involutive Bases* for polynomial ideals was introduced in [13]. The idea was derived from the theory of algebraic analysis of PDEs. By calculating the involutive basis of a system, one can study the same kind of problems addressed by Gröbner bases. In fact, one can show that an involutive basis is a special, though usually redundant, form of Gröbner basis. Involutive Bases algorithms have shown to be particularly efficient, and could therefore be also a useful tool for solving the BES system.

Additionally, there are a number of common techniques used in cryptanalysis that could be used in conjunction with computer algebra methods mentioned above. For example, by adapting a technique known as *meet-in-the-middle* (see [15], Chapter 7), the cryptanalyst could consider 2 systems with half of the size of the original one. This has the potential to reduce the complexity of the attack. One should also note that in practice the attacker is not interested in the full solution of the system, but only in the key variables. In fact, in a "partial key recovery" attack, only few key variables might suffice.

Therefore, it is possible that one may be able to use a combination of cryptanalytic and algebraic techniques (including Gröbner bases) to mount a successful attack without actually computing the solution of the entire system.

The Ideal Generated by the BES System. In the attempt to solve the polynomial system derived from the BES cipher, it may be interesting to obtain some information about the ideal generated by the polynomials in the system.

Let S be the system described above. If one fixes the encryption key $\mathcal{K}$, then for every plaintext/ciphertext pair (P, C) we have a derived system $S_{(P,C)}$ and the ideal [3]

$$I_{(P,C)} = \langle S_{(P,C)} \rangle \subseteq \mathbb{K}[x_{i,(j,m)}, \ldots, w_{i,(j,m)}, \ldots, k_{i,(j,m)}].$$

In fact, when analysing the AES, we are mostly interested in the ideal

$$I^{\mathcal{K}}_{(P,C)} = I_{(P,C)} \cap \mathbb{K}[k_0, k_1, \ldots, k_{15}]$$

where $k_0, k_1, \ldots, k_{15}$ are the first key addition variables (i.e. the cipher key, ignoring the conjugates).

Thus for every key $\mathcal{K}$, we can associate an ideal of $\mathbb{F}[k_0, k_1, \ldots, k_{15}]$ defined as:

$$I_{\mathcal{K}} = \bigoplus_{(P,C)} I^{\mathcal{K}}_{(P,C)}$$

where (P, C) run through all plaintext/ciphertext pairs. Equivalently, we want to calculate the variety

$$\mathbf{V}(I_{\mathcal{K}}) = \bigcap_{(P,C)} \mathbf{V}(I^{\mathcal{K}}_{(P,C)}).$$

Given a random key K, a random plaintext block P, and C such that $E_K(P) = C$, the probability that there exists another key K' with $E_{K'}(P) = C$ is approximated by a conditional Poisson distribution to give $(1 - 1/(e-1)) \cong 42\%$.

Therefore we expect that in many cases, for a given plaintext/ciphertext pair (P, C), the $\mathbb{K}$-dimension of the residue class ring $\mathbb{K}[k_0, k_1, \ldots, k_{15}]/I$ is greater than 1 (i.e., the corresponding reduced Gröbner basis should contain polynomials with degree greater than 1).

On the other hand, the $\mathbb{K}$-dimension of $\mathbb{K}[k_0, k_1, \ldots, k_{15}]/I_{\mathcal{K}}$ is almost certainly 1. In other words, we expect $I_{\mathcal{K}}$ to be of the form

$$I_{\mathcal{K}} =< k_0 - \kappa_0, k_1 - \kappa_1, \ldots, k_{15} - \kappa_{15} >$$

with $\kappa_i \in \mathbb{K}$. If this is not true, then there are *at least* two keys K_1 and K_2 such that

$$E_{K_1}(P) = E_{K_2}(P)$$

for *every plaintext block* P, and K_1 and K_2 induce the same permutation on the set of possible plaintext blocks, which is very unlikely for the AES.

[3] To avoid inconsistent systems, we will make sure to describe the system in such way that it does not include "0-inversions" (i.e. use the map $x \mapsto x^{254}$ when necessary).

5.3 Further Algebraic Properties

Even if the polynomial systems which have been described above cannot be efficiently solved to recover the cipher key, other approaches could well be used in order to mount less ambitious attacks against the BES (and therefore the AES), such as decryption of related ciphertexts.

At the very least, a cryptanalyst would like to be able to find a polynomial-time distinguisher to distinguish between the cipher and a random permutation. This could be used either to mount a practical attack or simply to show some structural weakness of the cipher.

Given the rich algebraic structure of the cipher, it is not entirely impossible that an "algebraic" distinguisher exists. This would most likely exploit the byte-oriented structure of the cipher and the typical round version of the BES, which consists of inversion, matrix multiplication and key addition, all operations over $\mathbb{F}_{2^8}$:

$$\mathbf{b} \mapsto M_B.\mathbf{b}^{-1} + (\mathbf{k}_B)_i$$

In particular, the linear layer seems to be highly structured. Recall that it has been shown in [17] that the matrix M_B has minimal polynomial $(x+1)^{15}$, and thus order 16. In fact, it is easy to show that the entire *affine layer*

$$\mathbf{x} \mapsto M_B.\mathbf{x} + (\mathbf{k}_B)_i$$

has order 16 (for any subkey $(\mathbf{k}_B)_i$).

While many of these properties might not prove to be relevant in the cryptanalysis of the AES, it is not inconceivable that one could find a novel way to explore this structure in the analysis of the cipher.

6 Conclusion

We investigated some computational and algebraic techniques that can be applied in the analysis of the Advanced Encryption Standard (AES). One promising approach is to exploit the large, though surprisingly simple, system of multivariate quadratic equations over the finite field $\mathbb{F}_{2^8}$ derived from the BES cipher. Given its algebraic formulation, an efficient method for solving this system would provide us with an attack on the AES with potentially very few plaintext-ciphertext pairs. While the problem of solving such systems is known to be hard, it is not entirely unlikely that a technique can be developed which exploits the particular algebraic structure of the AES and BES systems.

References

1. Alex Biryukov and Christophe De Canniere. Block Ciphers and Systems of Quadratic Equations. In *FSE'2003*, 2003.
2. Jiun-Ming Chen and Bo-Yin Yang. Theoretical Analysis of XL over Small Fields. In *Proceedings of the 9th Australasian Conference on Information Security and Privacy*, 2004. to appear.
3. D. Coppersmith. Personal communication, 30 April 2002.
4. Nicolas Courtois. Algebraic Attacks over $GF(2^k)$, Applications to HFE Challenge 2 and Sflash-v2. In F. Bao et al., editor, *PKC 2004*, volume 2947 of *LNCS*, pages 201–217. Springer-Verlag, 2004.
5. Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations. In *Eurocrypt'2000*, pages 392–407. Springer, 2000.
6. Nicolas Courtois and Josef Pieprzyk. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. In Yuliang Zheng, editor, *Advances in Cryptology - ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 267–287. Springer, 2002.
7. Nicolas Courtois and Josef Pieprzyk. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. Cryptology ePrint Archive, Report 2002/044, 2002.
8. David Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms*. Undergraduate Texts in Mathematics. Springer, Second edition, 1997.
9. Joan Daemen and Vincent Rijmen. *The Design of Rijndael*. Springer-Verlag, 2002.
10. Jean-Charles Faugère. A new efficient algorithm for computing Gröbner Bases without reduction to zero F5. In T. Mora, editor, *International Symposium on Symbolic and Algebraic Computation - ISSAC 2002*, pages 75–83, July 2002.

11. Jean-Charles Faugère and Antoine Joux. Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems using Gröbner Bases. In D. Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *LNCS*, pages 44–60. Springer-Verlag, 2003.
12. N. Ferguson, R. Shroeppel, and D. Whiting. A simple algebraic representation of Rijndael. In *Proceedings of Selected Areas in Cryptography*, pages 103–111. Springer-Verlag, 2001.
13. Vladimir Gerdt and Yuri Blinkov. Involutive Bases of Polynomial Ideals. *Mathematics and Computers in Simulation*, 45:519–542, 1998.
14. Susan Landau. Polynomials in the Nation's Service: Using Algebra to Design the Advanced Encryption Standard. *American Mathematical Monthly*, pages 89–117, February 2004.
15. Alfred J. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.
16. Sean Murphy and Matthew Robshaw. New observations on Rijdael, August 2000. NIST AES website.
17. Sean Murphy and Matthew Robshaw. Essential Algebraic Structure within the AES. In M. Yung, editor, *Advances in Cryptology - CRYPTO 2002*, volume 2442 of *LNCS*, pages 1–16. Springer-Verlag, 2002.
18. Sean Murphy and Matthew Robshaw. Comments on the Security of the AES and the XSL Technique. *Electronic Letters*, 39:26–38, 2003.