

Block Ciphers Sensitive to Gröbner Basis Attacks

Johannes Buchmann, Andrei Pyshkin*, Ralf-Philipp Weinmann

Technische Universität Darmstadt, Fachbereich Informatik,
Hochschulstr. 10, D-64289 Darmstadt, Germany
{buchmann,pyshkin,weinmann}@cdc.informatik.tu-darmstadt.de

Abstract. We construct and analyze Feistel and SPN ciphers that have a sound design strategy against linear and differential attacks but for which the encryption process can be described by very simple polynomial equations. For a block and key size of 128 bits, we present ciphers for which practical Gröbner basis attacks can recover the full cipher key requiring only a minimal number of plaintext/ciphertext pairs. We show how Gröbner bases for a subset of these ciphers can be constructed with negligible computational effort. This reduces the key-recovery problem to a Gröbner basis conversion problem. By bounding the running time of a Gröbner basis conversion algorithm, FGLM, we demonstrate the existence of block ciphers resistant against differential and linear cryptanalysis but vulnerable against Gröbner basis attacks.

1 Introduction

Since the publication of Courtois' and Pieprzyk's XSL method [9] and Murphy and Robshaw's embedding of the AES [21], a considerable interest in algebraic attacks on block ciphers has been provoked. While linearization based attacks on stream ciphers have been shown to be very successful, the claimed attacks on the AES and Serpent have thus far been highly controversial, if not outright refuted [5]. Gröbner bases however are a proven tool for solving polynomial systems. Cid, Murphy and Robshaw [6] recently did a first step of investigating the viability of an algebraic attack using Gröbner bases on scaled-down versions of the AES.

The goal of this paper is to show that non-trivial iterated block ciphers with a reasonable block and key length – in our case 128 bits – can be constructed that are resistant against linear and differential cryptanalysis but which can be broken by computing an appropriate Gröbner basis.

The paper is organized as follows. In Section 2 we present two families of ciphers, FLURRY, a Feistel network and CURRY, a SPN construction, together with suitable parameters. We explain how to obtain polynomial equations describing the key recovery problem. Section 3 then introduces the methodology to obtain estimates on the complexity of attacks using linear and differential

* Supported by a stipend of the Marga und Kurt Möllgaard-Stiftung.

cryptanalysis. Section 4 details how Gröbner bases can be used break the ciphers and gives experimental results for selected examples. Finally we show how the key recovery problem for a subset of these ciphers is related to the problem of Gröbner basis conversion.

1.1 Notation

We define the notation that we will be used throughout the rest of this paper.

All operations of the block ciphers described in this paper are carried out over a finite field $F := GF(2^n)$ with $n \in \{8, 16, 32, 64\}$. We fix θ to be a generating element of F over $GF(2)$, i.e. $F := GF(2)(\theta)$.

The internal state of our cipher consists of multiple elements of F . To refer to individual elements of the state after the execution of a complete round transformation we use the following conventions:

- For Feistel ciphers, the internal state is represented by a vector. We use variables $x_i^{(e)}$ to denote elements of the internal state of the cipher after the e th application of the round function and variables $k_i^{(e)}$ to denote elements of the expanded key used in round e .
- For SPN ciphers, the internal state is represented by a square matrix. We denote the *internal state variables* after the e th application of the round function by $x_{i,j}^{(e)}$ and the *expanded key variables* by $k_{i,j}^{(e)}$.

We define the state of round 0 to be the initial state and call the variables of the initial state *plaintext variables*. Correspondingly the variables referring to the state after the execution of the last round are called *ciphertext variables*. The set of state variables of a cipher is denoted by \mathcal{X} , the set of expanded key variables by \mathcal{K} . All polynomials considered are then elements of the polynomial ring $R = F[\mathcal{X} \cup \mathcal{K}]$.

A power product of variables of $(\mathcal{X} \cup \mathcal{K})$ shall be called a *term*, whilst the product of a *term* and a *coefficient* $c \in F$ shall be called a *monomial*.

2 Description of the Cipher Families

In this section we give blueprints for Feistel and SPN ciphers that allow for simple algebraic representations. For these we select parameters sets offering a high resistance against differential and linear cryptanalysis and describe how to construct systems of polynomial equations for them.

2.1 The Feistel Case: FLURRY

We construct the family FLURRY(n, m, r, f, D) of Feistel ciphers. The parameters used are:

- $m \in \mathbb{N}$: the plaintext space, the ciphertext space and the cipher key space are F^{2m} .

- $r \in \mathbb{N}$: the number of rounds
- $f : F \rightarrow F$: a non-linear mapping giving the S-Box of the round function
- $D = (d_{i,j}) \in F^{m \times m}$: a matrix describing the linear diffusion mapping of the round function.

We set $R = (r_1, \dots, r_m) \in F^m$, $L = (l_1, \dots, l_m) \in F^m$ and $K = (k_1, \dots, k_m) \in F^m$. The round function $\rho : F^m \times F^m \times F^m \rightarrow F^m \times F^m$ of a FLURRY cipher is then defined as:

$$\rho(L, R, K) = (R, G(R, K) + L)$$

with $G : F^m \times F^m \rightarrow F^m$ being the parallel application of m S-Boxes followed by a linear transform:

$$G(r_1, \dots, r_m, k_1, \dots, k_m) = D \times \begin{pmatrix} f(r_1 + k_1) \\ f(r_2 + k_2) \\ \vdots \\ f(r_m + k_m) \end{pmatrix}.$$

A plaintext (L_0, R_0) is encrypted into a ciphertext (L_r, R_r) by iterating the round function ρ over r rounds:

$$\begin{aligned} (L_i, R_i) &= \rho(L_{i-1}, R_{i-1}, K_{i-1}) \quad i = 1, 2, \dots, r-1 \\ (L_r, R_r) &= \rho(L_{r-1}, R_{r-1}, K_{r-1}) + (K_r, K_{r+1}) \end{aligned}$$

After the last round transformation, an additional key addition is performed on both halves of the state. Analogously, using the inverse round function ρ^{-1}

$$\rho^{-1}(L, R, K) = (G(L, K) + R, L)$$

we can decrypt a ciphertext with the following sequence of steps:

$$\begin{aligned} (L_{r-1}, R_{r-1}) &= \rho^{-1}(L_r + K_r, R_r + K_{r+1}, K_{r-1}) \\ (L_{i-1}, R_{i-1}) &= \rho^{-1}(L_i, R_i, K_{i-1}) \quad i = r-1, r-2, \dots, 1 \end{aligned}$$

The number of F -components of a cipher key, plaintext or ciphertext is denoted by $t = 2m$.

The key schedule The key schedule is affine over F . We write the cipher key as a tuple of vectors $(K_0, K_1) \in F^m \times F^m$. Let the round keys for the first two rounds be K_0, K_1 and recursively compute subsequent round keys for $2 \leq i \leq r+1$ as follows:

$$K_i = D \cdot K_{i-1}^T + K_{i-2} + v_i$$

where D is the same matrix used in the round function of the cipher and the v_i are round constants:

$$v_i = ((\theta + 1)^i, (\theta + 1)^{i+1}, \dots, (\theta + 1)^{i+m-1})$$

2.2 The SPN Case: CURRY

In this section we construct a family $\text{CURRY}(n, m, r, f, D)$ of ciphers similar to SQUARE [11]. We explain the parameters used:

- $m \in \mathbb{N}$: the plaintext space, the ciphertext space and the cipher key space are $F^{m \times m}$.
- $r \in \mathbb{N}$: the number of rounds
- $f : F \rightarrow F$: a bijective non-linear mapping giving the S-Box of the round function
- $D = (d_{i,j}) \in F^{m \times m}$: an invertible matrix used for diffusion

The round function $\rho : F^{m \times m} \times F^{m \times m} \rightarrow F^{m \times m}$ of a CURRY cipher is defined as:

$$\rho(S, K) = D \cdot G(S + K)^T$$

with $G : F^{m \times m} \rightarrow F^{m \times m}$ being the parallel application of m^2 S-Boxes:

$$G((s_{i,j})) = (f(s_{i,j}))$$

A plaintext S_0 is encrypted into a ciphertext S_r by iterating the round function ρ exactly r times followed by an additional key addition after the last round:

$$\begin{aligned} S_i &= \rho(S_{i-1}, K_{i-1}) & i = 1, 2, \dots, r-1 \\ S_r &= \rho(S_{r-1}, K_{r-1}) + K_r \end{aligned}$$

Analogously, using the inverse round function ρ^{-1}

$$\rho^{-1}(S, K) = G^{-1}((D^{-1} \cdot S)^T) + K$$

we can decrypt a ciphertext with the following sequence of steps:

$$\begin{aligned} S_{r-1} &= \rho^{-1}(S_r + K_r, K_{r-1}) \\ S_{i-1} &= \rho^{-1}(S_i, K_i) & i = r-1, r-2, \dots, 1 \end{aligned}$$

Just as for FLURRY, let the number of F -components of a key, plaintext or ciphertext be denoted by t , this time $t = m^2$.

The key schedule For CURRY the first round key is equivalent to the cipher key $K_0 \in F^{m \times m}$. Just as for FLURRY the key schedule is affine over F . Subsequent round keys $K_i, i \geq 1$ are recursively computed as follows:

$$K_i = D \cdot K_{i-1} + M_i$$

where D is the same matrix used in the round function and $M_i = ((a_{j,l}))$ with $a_{j,l} = \theta^{i+(j-1)m+l}$. The matrices M_i are round constants.

2.3 Selected Parameters

We will now specify suitable parameters for the S-Box function and the linear transformation. These will be used to more thoroughly investigate instances of our cipher constructions throughout this paper. The number of rounds shall be left unspecified for now.

The S-Box functions The only non-linear components of FLURRY and CURRY are the S-Boxes. In order to achieve good resistance against differential and linear cryptanalysis even for low number of rounds these must be chosen very carefully. Two important characteristics of a S-Box are its differential uniformity and its nonlinearity. These are defined as follows:

Definition 1. Let $f : F \rightarrow F$ be a mapping and

$$\delta = \max_{\substack{a, b \in F \\ a \neq 0}} \#\{x \in F : f(x+a) = f(x) + b\}.$$

Then f is called differentially δ -uniform.

In the following definition we use the bijective map

$$F \rightarrow GF(2)^n, a = \sum_{i=0}^{n-1} (a_i \theta^i) \mapsto (a_0, \dots, a_{n-1})$$

to identify F with $GF(2)^n$. For $a = (a_0, \dots, a_{n-1})$, $b = (b_0, \dots, b_{n-1})$ we set

$$\langle a, b \rangle = \sum_{i=0}^{n-1} a_i b_i$$

Definition 2. The nonlinearity of a function $f : F \rightarrow F$ is defined as

$$\mathcal{N}(f) = \min_{\substack{a, b \in F \\ b \neq 0}} \#\{x \in F \mid \langle x, a \rangle \neq \langle f(x), b \rangle\}$$

For monomial functions as well as the multiplicative inverse over finite fields of characteristic two the δ -uniformity and the nonlinearity have been well studied in the literature [22,2,13]. We want to keep the degree of our S-Box functions low in order to make Gröbner basis attacks feasible. Table 1 shows the S-Box functions that we have picked.

We call f_3, f_5 and f_7 *monomial S-Boxes* and f_{-1} the *inversion S-box*.

Lemma 1. 1. f_3 is a 2-uniform mapping

2. f_{-1} and f_5 are 4-uniform mappings.

3. f_7 has δ -uniformity of 6 or less.

Proof. Obviously for all $a, b \in F$ with $a \neq 0$ the equation $x^7 + (x+a)^7 = b$ has at most 6 roots. For claims 1 and 2, see [22].

Lemma 2. 1. The nonlinearity of f_{-1} is $2^{n-2} - 2^{\frac{n}{2}}$.

2. For a polynomial function $f : F \rightarrow F$ of degree d the following holds true:

$$\mathcal{N}(f) \geq 2^{n-1} - \lfloor \frac{d-1}{2} \rfloor 2^{\frac{n}{2}}$$

Proof. For claim 1, see [13], for claim 2 see [4].

Table 1. S-Box mappings over $GF(2^n)$ with $n \in \{8, 16, 32, 64\}$

| function | mapping | bijective over $GF(2^n)$ | δ -uniformity | $\mathcal{N}(f)$ |
|----------|--|--------------------------|----------------------|--|
| f_{-1} | $x \mapsto \begin{cases} x^{-1} & \text{iff } x \neq 0 \\ 0 & \text{iff } x = 0 \end{cases}$ | yes | 4 | $2^{n-1} - 2^{\frac{n}{2}}$ |
| f_3 | $x \mapsto x^3$ | no | 2 | $\geq 2^{n-1} - 2^{\frac{n}{2}}$ |
| f_5 | $x \mapsto x^5$ | no | 4 | $\geq 2^{n-1} - 2^{\frac{n}{2}+1}$ |
| f_7 | $x \mapsto x^7$ | yes | ≤ 6 | $\geq 2^{n-1} - 3 \cdot 2^{\frac{n}{2}}$ |

The linear transformations We use matrices of Maximum Distance Separable codes – *MDS matrices* for short – for the matrix D in the linear layer and the key schedule. We chose these types of linear transformations since they have optimal diffusion properties. This strategy is widely used in modern block cipher design; all ciphers following the wide-trail design use diffusion optimal matrices. The matrix D_4 below actually is the matrix used in the `MixColumns` step of Rijndael, D_2 is equivalent to a Pseudo-Hadamard Transform over F .

$$D_2 = \begin{pmatrix} \theta & 1 \\ 1 & 1 \end{pmatrix} \quad D_4 = \begin{pmatrix} \theta & \theta+1 & 1 & 1 \\ 1 & \theta & \theta+1 & 1 \\ 1 & 1 & \theta & \theta+1 \\ \theta+1 & 1 & 1 & \theta \end{pmatrix}$$

Rijmen and Daemen introduced the notion of the *branch number* of a linear transformation to measure the quality of the diffusion provided. For a F -vector $X := (x_1, \dots, x_m)$ we define $w(X)$ to be the hamming weight of X , i.e. the count of all non-zero coordinates of this vector. The following definition is according to [12]:

Definition 3. Let $M \in F^{m \times m}$ be a matrix describing a linear map. The differential branch number $\mathcal{B}_d(M)$ of M is then defined as

$$\mathcal{B}_d(M) = \min_{\substack{X \in F^m \\ X \neq 0}} (w(X) + w(MX))$$

while the linear branch number $\mathcal{B}_l(M)$ is defined as $\mathcal{B}_l(M) = \mathcal{B}_d(M^T)$.

For a symmetric matrix such as D_2 , the linear and the differential branch number clearly coincide. For the circulant matrix D_4 the linear and differential branch number coincide as well [12]. Thus in our case it suffices to speak of *the branch number* $\mathcal{B}(M)$ of a matrix M . For MDS matrices the branch number is maximal [12], i.e. $\mathcal{B}(M) = m + 1$ with m being the size of the matrix M . For block ciphers with $m = 1$ we use the identity matrix of size one, I_1 , trivially resulting in $\mathcal{B}(I_1) = 2$.

2.4 Polynomial Representation of the Ciphers

In the following we will detail how to obtain a system of polynomial equations that describes the transformation of a plaintext into a ciphertext block round

by round using intermediate state variables. Please note that our description is slightly simplified. For the sake of legibility we have omitted the round key addition after the final round; for our experiments the final key addition has of course been retained.

– FLURRY

For Feistel ciphers the left half of the state in round e is identical to the right half of the state in round $e - 1$, giving rise to the following mr trivial linear equations:

$$x_j^{(e)} + x_{j+m}^{(e-1)} = 0$$

Each monomial S-Box of the cipher induces a polynomial equation of degree $\deg(f)$. Thus we get a total of mr non-linear equations of form:

$$x_{m+j}^{(e)} + x_j^{(e-1)} + \sum_{l=1}^m d_{j,l} \cdot f\left(x_{m+l}^{(e-1)} + k_l^{(e-1)}\right) = 0$$

with $1 \leq e \leq r$, $1 \leq j \leq m$. When using the inversion S-Box the polynomial system is correct only with probability $\left(\frac{2^n-1}{2^n}\right)^{mr}$. The equations in this case are of a different form:

$$\left(x_j^{(e-1)} + x_{m+j}^{(e)}\right) \prod_{i=1}^m \left(x_{m+i}^{(e-1)} + k_i^{(e-1)}\right) + \sum_{l=1}^m d_{j,l} \prod_{\substack{i=1 \\ i \neq l}}^m \left(x_{m+i}^{(e-1)} + k_i^{(e-1)}\right) = 0$$

The linear equations for the key schedule of FLURRY can be written as:

$$k_j^{(e)} + k_j^{(e-2)} + (\theta + 1)^{et+j} + \sum_{l=1}^m d_{j,l} k_l^{(e-1)} = 0$$

with $2 \leq e \leq r$, $1 \leq j \leq m$.

– CURRY

No trivial linear equations hold between intermediate state variables.

Denote by $x_{(i,j)}^{(e)}$ the variable in row i , column j of the state in round e , analogously for $k_{(i,j)}^{(e)}$. Then for all rounds $e > 0$ the following equations hold with $1 \leq i, j \leq m$:

$$x_{i,j}^{(e)} + \sum_{l=1}^m d_{i,l} \cdot f\left(x_{j,l}^{(e-1)} + k_{j,l}^{(e-1)}\right) = 0$$

Again for f_{-1} the non-linear equations look different:

$$x_{i,j}^{(e)} \prod_{u=1}^m \left(x_{j,u}^{(e-1)} + k_{j,u}^{(e-1)}\right) + \sum_{l=1}^m d_{i,l} \prod_{\substack{u=1 \\ u \neq l}}^m \left(x_{j,u}^{(e-1)} + k_{j,u}^{(e-1)}\right) = 0$$

Using the above equations, the polynomial system also does not hold with probability one but with probability $\left(\frac{2^n-1}{2^n}\right)^{m^2 r}$. The linear equations for the key schedule can be expressed as follows:

$$k_{i,j}^{(e)} + (\theta)^{e+(i-1)m+j} + \sum_{l=1}^m d_{i,l} k_{l,j}^{(e-1)} = 0$$

with $1 \leq e \leq r$, $1 \leq i, j \leq m$.

Additionally, for each variable $\mathbf{v} \in (\mathcal{X} \cup \mathcal{K})$ the relation $\mathbf{v}^{2^n} + \mathbf{v} = 0$ holds. These relations are called *field equations*; they will not be included in our polynomial system however.

3 Resistance against Classical Attacks

In this section we determine the strength of our cipher constructions against differential and linear cryptanalysis. Differential cryptanalysis is a chosen-ciphertext attack due to Biham and Shamir and was the first successful attack on the DES [3]. This type of attack exploits biases in the first order derivative of the cipher. For carefully chosen plaintexts with specific differences a cryptanalyst makes assumption about their propagation through the cipher and predicts output differences in ciphertext pairs. If these predictions are correct with sufficiently high probability they allow an attacker to determine round key bits.

Linear cryptanalysis is a known plaintext attack that was devised by Matsui [20] to attack the DES. For this attack to succeed, the cryptanalyst has to construct a probable key-independent linear approximation for individual output bits of the cipher. By counting the number of time this linear approximation agrees with the actual output of the cipher she can establish which value for the key bit is more likely.

The notion of *practical security* of block ciphers against differential and linear cryptanalysis was introduced by Knudsen [19]. The exact definition of this notion is postponed to the end of Section 3.2. We will derive the number of rounds that will make our cipher practically secure against differential and linear cryptanalysis.

Note that our objective was not to evaluate the strength of our ciphers against all known attacks. Our ciphers may very well be vulnerable against one or several advanced attacks even if they resist standard linear and differential cryptanalysis. Indeed, as an example we argue that the choices we have made for the S-Boxes are very weak against interpolation attacks.

3.1 Estimating the Resistance against Differential and Linear Cryptanalysis

A fundamental parameter that influences the complexity of differential and linear attacks is the minimum number of active S-Boxes N over consecutive rounds of the cipher. Kanda [18] gives useful results on both SPN ciphers and Feistel ciphers with a SP round function; from these we derive the following lemma:

Lemma 3. *The minimum number of active S-boxes in 4, 6, 8 consecutive rounds of a Feistel cipher with SP round function is lower bounded by $\mathcal{B}(D)$, $\mathcal{B}(D) + 2$ and $2\mathcal{B}(D) + 1$ respectively. For an SPN cipher the minimum number of active S-Boxes for $2r$ consecutive rounds is lower bounded by $r\mathcal{B}(D)$.*

In the following X denotes a uniformly distributed random variable in $GF(2)^n$ and $\rho : GF(2)^n \rightarrow GF(2)^n$ a function for which we wish to compute the linear and differential probability.

Definition 4. *The linear probability for a pair $(a, b) \in GF(2)^n \times GF(2)^n$ with $a \neq 0$ is defined as*

$$LP(a, b) = (2 \cdot \Pr_X \{ \langle a, X \rangle = \langle b, \rho(X) \rangle \} - 1)^2$$

In the above definition, a is called *input mask* and b is called *output mask* of a round. A vector of masks $A = (a_1, \dots, a_{r+1})$ with $a_i \neq 0$ for all $1 \leq i \leq r$ is called *linear characteristic* of a cipher.

Definition 5. *The differential probability for a pair $(\Delta x, \Delta y) \in GF(2)^n \times GF(2)^n$ with $\Delta x \neq 0$ is defined as*

$$DP(\Delta x, \Delta y) = \Pr_X \{ \rho(X) + \rho(X + \Delta x) = \Delta y \}$$

The value Δx is called *input difference* of a round, while Δy is called *output difference*. A vector of differences $A = (a_1, \dots, a_{r+1})$ with $a_i \neq 0$ for all $1 \leq i \leq r$ is called *differential characteristic* of a cipher.

Definition 6. *Let Ω_L be the set of all linear characteristics and Ω_D the set of all differential characteristics of a cipher C . The maximum linear characteristic probability (MLCP) of C is*

$$MLCP(C) = \max_{A \in \Omega_L} \prod_{i=1}^r LP(a_i, a_{i+1})$$

Analogously the maximum differential characteristic probability (MDCP) of C is

$$MDCP(C) = \max_{A \in \Omega_D} \prod_{i=1}^r DP(a_i, a_{i+1})$$

3.2 Differential and Linear Cryptanalysis of FLURRY and CURRY

In this section we show how to compute upper bounds of MLCP and the MDCP of ciphers of the FLURRY and CURRY family. From these bounds we can deduce the number of rounds required to make an instance practically secure against differential and linear cryptanalysis.

The maximum differential probability of a function $f : F \rightarrow F$ can be calculated from δ as $p(f) = \frac{\delta}{\#F}$ where δ is according to Definition 1. The maximum linear probability of a mapping $f : F \rightarrow F$ can be computed as

$$q(f) = \left(1 - \frac{2\mathcal{N}(f)}{\#F} \right)^2$$

where $\mathcal{N}(f)$ is defined as in Section 2.3. For SPN ciphers and Feistel ciphers with a SP round function the MDCP is bounded by $p(f)^N$ while the MLCP is bounded by $q(f)^N$ [18], where N is the minimum number of active S-Boxes.

According to Knudsen [19], a block cipher with dependent round keys is practically secure against differential and linear cryptanalysis if the MLCP and the MDCP is too low for an attack to work under the assumption of independent round keys. Note however that for both r -round Feistel and r -round SPN ciphers, we need to consider the MLCP and MDCP of $r - 2$ rounds because of attacks that guess bits of the first and the last round key, so-called 2R attacks.

3.3 Interpolation Attacks

Jakobsen and Knudsen presented interpolation attacks in [16] as a counterpoint to the growing trend of using algebraic S-Boxes such as those proposed by Nyberg [22]. In fact, interpolation attacks can be seen as the first algebraic attacks on block ciphers. The underlying intuition of this attack is that the relationship between plaintext and ciphertext can be expressed as a tuple of polynomial expressions. If the degree of these polynomials is low enough, the coefficients of the polynomials can be interpolated from a number of plaintext/ciphertext pairs. A key-dependent equivalent of the encryption or the decryption algorithm has then been determined. In [16] upper bounds on the number of required pairs for known-plaintext interpolation attacks for selected examples are given. In general this number increases exponentially with the degree of the polynomial function describing the S-Box, the number of rounds and the number of elements in the internal state, while for the attacks we present in the next section it remains a constant quantity.

Courtois later improved on the work of Jakobsen and Knudsen and introduced an attack called General Linear Cryptanalysis [8]. In the same paper he also gives several examples of insecure ciphers based on inversion based S-Boxes that resist differential and linear cryptanalysis. His approach and his goals are quite different from ours however.

FLURRY and CURRY quite naturally are susceptible to interpolation attacks – their clean structure and the monomial S-boxes make them textbook examples. As a matter of fact, the cipher *PURE* presented in the original article is identical to the 64-bit cipher FLURRY(32, 1, r , f_3 , I_1) sans key scheduling.

4 Attacks Using Gröbner Bases

Gröbner bases are standard bases of polynomial ideals that can be used for solving systems of polynomial equations. What Gaussian elimination does for systems of linear equations, Gröbner basis algorithms try to emulate for polynomial systems. Unfortunately the computational complexity of Gröbner basis algorithms for nonlinear systems is no longer polynomial. In this paper we restrict ourselves to known-plaintext *Gröbner Basis attacks* that recover a secret key of a block cipher from a minimum number of plaintext/ciphertext pairs faster

than a sequential exhaustive search of the key space – by computing Gröbner Bases.

We will briefly introduce the concepts necessary to explain our results. For a more thorough introduction to Gröbner bases we refer the reader to [1] and [10]. In the following we adopt the conventions of [1].

Definition 7 (Term order). *A term order \leq is a linear order on the set of terms $\mathcal{T}(R)$ such that*

1. $1 \leq t$ for all terms $t \in \mathcal{T}(R)$
2. for all terms $s, t_1, t_2 \in \mathcal{T}(R)$ whenever $t_1 \leq t_2$ then $st_1 \leq st_2$

If a term order has been fixed, we define $\text{HT}(f)$ to be the greatest term occurring in the polynomial $f \in R$ according to this order; this term is called the head term. Correspondingly $\text{HM}(f)$ is the head monomial, i.e. the head term of f multiplied with the matching coefficient.

We will now introduce two useful and widely used term orders. To accomplish this we first need to define some technicalities: For a term $t = \mathbf{v}_1^{e_1} \mathbf{v}_2^{e_2} \cdots \mathbf{v}_k^{e_k} \in \mathcal{T}(R)$ we define the *exponent vector* of t to be $\epsilon(t) = (e_1, e_2, \dots, e_k) \in \mathbb{N}_0^k$. The total degree of the term t then is $\text{deg}(t) = \sum_{i=1}^k e_i$.

Example 1 (Lexicographic term order). For terms s, t we define $s <_{lex} t$ iff there exists an i with $1 \leq i \leq k$ such that the first $i - 1$ components of $\epsilon(s)$ and $\epsilon(t)$ are equal but the i th component of $\epsilon(s)$ is smaller than the i th component of $\epsilon(t)$.

Example 2 (Degree reverse lexicographic term order). For terms s, t we define $s <_{DRL} t$ iff either $\text{deg}(s) < \text{deg}(t)$ or if $\text{deg}(s) = \text{deg}(t)$ and $s <_{lex} t$.

Definition 8 (Syzygy polynomial). *The syzygy polynomial of two polynomials f, g is defined as*

$$\text{spol}(f, g) = \frac{\text{lcm}(\text{HM}(f), \text{HM}(g))}{\text{HM}(f)} f - \frac{\text{lcm}(\text{HM}(f), \text{HM}(g))}{\text{HM}(g)} g$$

For a set of polynomials $G \subset R$ we can define the reduction of a polynomial $f \in R$ to a remainder r which we will denote by $f \rightarrow_G r$. The result of this operation may not be uniquely defined unless G is a Gröbner basis. In the following we will only be interested in polynomial divisions that leave no remainder.

Definition 9 (Reduction to zero). *A polynomial $f \in R$ reduces to zero modulo a set $G = \{g_1, \dots, g_k\} \subset R$, if there exists a vector of polynomials (m_1, \dots, m_k) such that $f - \sum_{i=1}^k m_i g_i = 0$ with $\text{HT}(m_i g_i) \leq \text{HT}(f)$ for all $1 \leq i \leq k$.*

Definition 10 (Gröbner basis). *Let \mathfrak{J} be an ideal of R . A finite set of polynomials $G \subset \mathfrak{J}$ is a Gröbner basis of \mathfrak{J} if $f \rightarrow_G 0$ holds for every $f \in \mathfrak{J}$.*

Let \mathcal{P} be a set of multivariate polynomial equations $p_i = 0$. For the ideal \mathfrak{J} generated by the set $P = \{p_i\}$ computing the Gröbner basis relative to an appropriate term order, e.g the lexicographical term order enables us to solve the system \mathcal{P} .

Computing a Gröbner basis relative to a total-degree order however usually is faster than computing a lexicographical Gröbner basis of the same ideal. This was the reason for the development of algorithms that change the term order of a Gröbner basis. The two most prominent are the FGLM algorithm [15] and the Gröbner Walk [7]. While the FGLM algorithm as originally described only works for zero-dimensional ideals, i.e. when the number of solutions of \mathcal{P} in the closure of F is finite, the Gröbner Walk does not have this restriction.

4.1 Key Recovery Using Gröbner Bases

Estimating the time and space complexity of Gröbner basis algorithms is no easy feat. For polynomial systems induced by block ciphers, no theoretical works estimating the performance of Gröbner basis algorithms are currently known. We therefore carried out experiments to study the resistance of our ciphers against Gröbner Basis attacks. Results of these experiments are presented and analysed in section 4.2.

The Gröbner basis attack we have successfully used on instances of FLURRY and CURRY to determine the secret key from a small number of plaintext/ciphertext pairs entailed the following steps:

1. Set up a polynomial system $\mathcal{P} = \{p_i = 0\}$ for the cipher in question with $p_i \in R$ as described in Section 2.4. The system \mathcal{P} consists of both cipher and key schedule equations.
2. Request a plaintext/ciphertext pair $((P_1, \dots, P_t), (C_1, \dots, C_t))$. This gives rise to the following additional system of linear equations $\mathcal{G} = \{g_i = 0\}$:

$$\begin{array}{ll} x_1^{(0)} + P_1 = 0 & x_1^{(r)} + C_1 = 0 \\ \vdots & \vdots \\ x_t^{(0)} + P_t = 0 & x_t^{(r)} + C_t = 0 \end{array}$$

Let \mathfrak{J} be the ideal generated by the set of polynomials $\mathcal{L} = (\bigcup_i \{p_i\}) \cup (\bigcup_i \{g_i\})$. We call this ideal the *key recovery ideal*.

3. Compute a degree-reverse lexicographic Gröbner basis G_{DRL} of \mathfrak{J} . For ciphers using a multiplicative inverse as S-Box function, the system may be inconsistent, resulting in $G_{DRL} = 1$.
4. If $G_{DRL} = 1$ go to Step 2, otherwise proceed.
5. Use a Gröbner basis conversion algorithm to obtain a lexicographical Gröbner basis G_{lex} from G_{DRL} . The variable ordering should be such that the key variables of the first round are the least elements.
6. Compute the variety Z of \mathfrak{J} using the Gröbner basis G_{lex} .

7. Request another plaintext/ciphertext pair (P', C') .
8. Try all elements $k \in Z$ as key candidates to encrypt P' . If k does not encrypt P' to C' , remove k from Z , otherwise retain.
9. If Z contains more than one element, go to step 7.
10. Terminate

Considerable complexity is hidden in step 6. To compute the variety of an ideal using a lexicographical Gröbner basis, we need to successively eliminate variables by computing zeroes of univariate polynomials and back-substituting results. The complexity of this depends on the number of solutions of the polynomial system (zeroes of the ideal) and the complexity of the algorithm for finding roots of univariate polynomials. The best algorithm for factoring polynomials is due to Kaltofen and Shoup [17] and has a complexity of $O(d^{1.815}n)$ field operations, where d is the degree of the polynomial. This degree is bounded by $2^n - 1$. The number zeroes is equivalent to the number of distinct keys encrypting the plaintext to a ciphertext. In general we can expect this number to be small.

4.2 Experimental Results

We have performed experiments to analyze the resistance of FLURRY and CURRY using the computer algebra system MAGMA [23], version 2.11-8, on an AMD Athlon 64 3200+ equipped with 1024 Megabytes of RAM running Linux. MAGMA implements Faugère's F4 algorithm [14] and is widely considered the best publicly available tool for computing Gröbner bases. We have chosen n and m such that the ciphers evaluated are 128-bit block ciphers.

Table 2 lists a number of instantiations of FLURRY and CURRY ciphers for which we were able to successfully recover the secret key; the 6, 8 and 10 round FLURRY ciphers are resistant to linear and differential cryptanalysis. We see that ciphers with inversion-based S-boxes are easier to break than ciphers which use a monomial S-box, even if the monomial is of very low degree. Furthermore we were unable to determine an a priori indicator for selecting the most efficient Gröbner basis conversion algorithm – in some cases FGLM was faster, in other cases the Gröbner walk; the same holds for the memory consumption. As mentioned in Section 2.4 we did not add the field equations to our polynomial systems.

4.3 Gröbner Bases without Polynomial Reductions

Sometimes one can determine whether a set of polynomials forms a Gröbner basis without computing normal forms. In the following let be $G \subset R$ be a finite set of polynomials with $0 \notin G$.

Proposition 1 (First Buchberger criterion). *Suppose that we have $f, g \in G$ such that*

$$\text{lcm}(HT(f), HT(g)) = HT(f) \cdot HT(g)$$

i.e the head terms of f and g are pairwise prime. Then $\text{spol}(f, g) \rightarrow_G 0$.

Table 2. Experimental results obtained with MAGMA

| cipher | conversion | CPU time | memory used |
|-----------------------------------|------------|-------------|---------------|
| FLURRY(64, 1, 4, f_{-1}, I_1) | Walk | 0.011 s | 3.48 MBytes |
| FLURRY(64, 1, 4, f_{-1}, I_1) | FGLM | 0.011 s | 3.48 MBytes |
| FLURRY(64, 1, 4, f_3, I_1) | Walk | 0.04 s | 3.48 MBytes |
| FLURRY(64, 1, 4, f_3, I_1) | FGLM | 0.029 s | 3.58 MBytes |
| FLURRY(64, 1, 4, f_5, I_1) | Walk | 1.28 s | 3.97 MBytes |
| FLURRY(64, 1, 4, f_5, I_1) | FGLM | 2.3 s | 6.36 MBytes |
| FLURRY(64, 1, 4, f_7, I_1) | Walk | 13.61 s | 6.22 MBytes |
| FLURRY(64, 1, 4, f_7, I_1) | FGLM | 82.62 s | 33.4 MBytes |
| FLURRY(64, 1, 6, f_{-1}, I_1) | Walk | 0.15 s | 3.58 MBytes |
| FLURRY(64, 1, 6, f_{-1}, I_1) | FGLM | 0.059 s | 3.58 MBytes |
| FLURRY(64, 1, 6, f_3, I_1) | Walk | 59.91 s | 10.63 MBytes |
| FLURRY(64, 1, 6, f_3, I_1) | FGLM | 145.08 s | 193.24 MBytes |
| FLURRY(64, 1, 8, f_{-1}, I_1) | Walk | 3.43 s | 4.51 MBytes |
| FLURRY(64, 1, 8, f_{-1}, I_1) | FGLM | 1.46 s | 4.46 MBytes |
| FLURRY(64, 1, 10, f_{-1}, I_1) | Walk | 115.44 s | 14.74 MBytes |
| FLURRY(64, 1, 10, f_{-1}, I_1) | FGLM | 60.61 s | 12.39 MBytes |
| FLURRY(64, 1, 12, f_{-1}, I_1) | Walk | 4194.28 s | 99.97 MBytes |
| FLURRY(64, 1, 12, f_{-1}, I_1) | FGLM | 2064 s | 142.90 MBytes |
| FLURRY(32, 2, 4, f_{-1}, D_2) | Walk | 216.53 s | 25.58 MBytes |
| FLURRY(32, 2, 4, f_{-1}, D_2) | FGLM | 65.78 s | 41.62 MBytes |
| FLURRY(16, 4, 2, f_{-1}, D_4) | Walk | 264 s | 37.13 MBytes |
| FLURRY(16, 4, 2, f_{-1}, D_4) | FGLM | 26.119 s | 18.56 MBytes |
| CURRY(32, 2, 3, f_{-1}, D_2) | Walk | 1750.87 sec | 138.77 MBytes |
| CURRY(32, 2, 3, f_{-1}, D_2) | FGLM | 3676.26 sec | 107.54 MBytes |

Proposition 1 is the first Buchberger criterion. Together with the following theorem given in [10], we can decide whether a sequence of polynomials is a Gröbner basis from looking at the head terms alone.

Theorem 1. *The set G is a Gröbner basis iff $\text{spol}(f, g) \rightarrow_G 0$ for all $f, g \in G$ with $f \neq g$.*

When using polynomial S-boxes, this enables us to compute a degree-reverse lexicographic Gröbner bases of the key-recovery ideals of FLURRY and CURRY without performing polynomial reductions; the head terms of all polynomials of \mathfrak{J} are univariate. For each polynomial of round e , either a power of a state variable of the preceding round or a power of a key variable of the current round occur as head term. Some head terms however occur more than once.

By using an appropriate variable order we can force the set of head terms of each round to be disjoint from the set of head terms of all other rounds:

– CURRY

For better legibility, we identify $x_{i,j}^{(e)}$ with $x_{et+im+j}$ and $k_{i,j}^{(e)}$ with $k_{et+im+j}$.

We then fix the following variable order:

$$\underbrace{x_0 < \dots < x_{t-1}}_{\text{plaintext variables}} < \underbrace{x_{tr} < \dots < x_{t(r+1)-1}}_{\text{ciphertext variables}} < \underbrace{k_0 < \dots < k_{t(r+1)-1}}_{\text{key variables}} < \underbrace{x_t < \dots < x_{tr-1}}_{\text{internal state variables}}$$

– FLURRY

Again we decrease the number of indexes: we identify $x_i^{(e)}$ with x_{et+i} and $k_i^{(e)}$ with k_{et+i} . We then fix the following variable order:

$$\begin{array}{c} \underbrace{x_0 < \dots < x_{t-1}}_{\text{plaintext variables}} < \underbrace{x_{tr} < \dots < x_{(t+1)r-1}}_{\text{ciphertext variables}} < \underbrace{x_{t(r-1)+m} < \dots < x_{tr-1}}_{\substack{\text{state variables of the right} \\ \text{half of the second last round}}} < \\ & \underbrace{k_0 < \dots < k_{m-1}}_{\substack{\text{key variables of} \\ \text{the first round}}} < \underbrace{k_{m(r-1)} < \dots < k_{mr-1}}_{\text{key variables of round } r} < \\ & \underbrace{k_m < \dots < k_{m(r-1)-1} < k_{mr} < \dots < k_{m(r+2)-1}}_{\text{remaining key variables}} < \underbrace{x_t < \dots < x_{t(r-1)+m-1}}_{\text{remaining state variables}} \end{array}$$

To make the following linear transformation easier to describe we use a vectorial representation for FLURRY and a matrix representation for CURRY. The entries in the vector and matrix of each round are the left-hand side polynomials of the nonlinear cipher equations.

We can multiply the vectors respectively matrices of all rounds by D^{-1} to obtain pairwise prime head terms within each and across rounds. For CURRY this is sufficient. For FLURRY we also need to adjust the key schedule equations. The nonlinear polynomials of the first and the last round have powers of key variables as head terms. These key variables are of the first and the last round respectively. For the first round this poses no problem. However for the last round the key schedule polynomials that produce the last round key have the same head terms. Thus we rewrite the key schedule equations. We express all round keys except for the last round key as a linear combination of the first two round keys. Then we write the second round key as a linear combination of the first and the last round key. This results in all head terms being pairwise prime. In order for this to work for FLURRY, the order of the matrix used in the key schedule needs to be greater than the number of rounds.

We have shown how to make the head terms of all polynomials pairwise prime. Hence by Theorem 1, we have obtained a Gröbner basis. This strategy however does not work FLURRY and CURRY instances with inversion S-Boxes, as the head terms in these cases are never univariate.

4.4 Complexity Of Gröbner Basis Conversions Using FGLM

The complexity of the FGLM algorithm hinges on two parameters of the input G : the number of variables of the polynomial ring R and the vector space dimension

of the residue class ring R/\mathfrak{J} , where \mathfrak{J} is the ideal generated by the Gröbner basis $G \subset R$. The following theorem [1] shows how this invariant of an ideal can be computed.

Theorem 2. *Let G be a Gröbner basis of the ideal \mathfrak{J} . Then*

$$\dim(R/\mathfrak{J}) = \#\{t \in \mathcal{T}(R) : HT(f) \nmid t \text{ for all } f \in G\}$$

Corollary 1. *Let $G = \{g_1, \dots, g_k\}$ be a Gröbner basis for the ideal $\mathfrak{J} \subset F[x_1, \dots, x_k]$ with head terms $x_1^{d_1}, \dots, x_k^{d_k}$. Then $\dim(R/\mathfrak{J}) = \prod_{i=1}^k d_i$.*

Corollary 2. *Let \mathfrak{J} be an ideal of an instantiation of either a FLURRY or a CURRY cipher as described in Section 2.4 and f a polynomial function. Then the following holds:*

1. $\dim(R/\mathfrak{J}) = \deg(f)^{mr}$ for FLURRY(n, m, r, f, D).
2. $\dim(R/\mathfrak{J}) = \deg(f)^{m^2r}$ for CURRY(n, m, r, f, D).

We restate Theorem 5.1 of [15].

Theorem 3. *Let K be a finite field and $R = K[x_1, \dots, x_k]$. Furthermore $G_1 \subset R$ is the Gröbner basis relative to a term order $<_1$ of an ideal \mathfrak{J} , and $d = \dim(R/\mathfrak{J})$. We can then convert G_1 into a Gröbner basis G_2 relative to a term order $<_2$ in $O(kd^3)$ field operations.*

We conjecture the constant factor in the above estimate to be approximately one cipher operation. For the space complexity of the algorithm, no bound is given in the original paper. We note that the dominant memory requirement of the FGLM algorithm is a $d \times kd$ matrix over F . Thus the memory usage of the algorithm is upper bounded by $\lceil (kd^2n)/8 \rceil + o(1)$ bytes.

This allows us to estimate the maximum resistance of FLURRY and CURRY ciphers with polynomial S-Boxes against Gröbner basis attacks (see Table 3). Note that for the CURRY cipher we need to use a bijective S-Box in the round function; the lowest degree S-Box function that is bijective is f_7 .

5 Conclusions

We have demonstrated that Gröbner basis algorithms can be used to successfully mount key-recovery attacks on algebraically simple block ciphers with a large block and key size; even when these ciphers are practically secure against differential and linear cryptanalysis. Key recovery can be accomplished with a minimal number of known plaintext/ciphertext pairs. Degree-reverse lexicographical Gröbner bases for our ciphers can be calculated by hand. These however do not give the solution to the polynomial system directly. Our contribution shows that the problem of recovering a key for these block ciphers can be reduced to a Gröbner basis conversion. By giving a formula for the vector space dimension of the polynomial ring modulo the key recovery ideal for all inversion-free ciphers

Table 3. Upper bounds on the complexity of breaking 128-bit FLURRY and CURRY ciphers with FGLM

| cipher | n | $\dim(R/I)$ | # of operations | memory required (bytes) |
|-------------------------------|-----|----------------------------|-----------------|-------------------------|
| FLURRY(32, 2, 4, f_3, D_2) | 8 | $3^8 \approx 2^{12.68}$ | $O(2^{41.0})$ | $2^{30.4}$ |
| FLURRY(32, 2, 4, f_5, D_2) | 8 | $5^8 \approx 2^{18.58}$ | $O(2^{58.7})$ | $2^{42.2}$ |
| FLURRY(32, 2, 4, f_7, D_2) | 8 | $7^8 \approx 2^{22.46}$ | $O(2^{70.4})$ | $2^{49.9}$ |
| FLURRY(32, 2, 6, f_3, D_2) | 12 | $3^{12} \approx 2^{19.02}$ | $O(2^{60.6})$ | $2^{43.2}$ |
| FLURRY(32, 2, 6, f_5, D_2) | 12 | $5^{12} \approx 2^{27.86}$ | $O(2^{87.2})$ | $2^{61.3}$ |
| FLURRY(32, 2, 6, f_7, D_2) | 12 | $7^{12} \approx 2^{33.69}$ | $O(2^{104.7})$ | $2^{73.0}$ |
| FLURRY(32, 2, 8, f_3, D_2) | 16 | $3^{16} \approx 2^{25.36}$ | $O(2^{80.0})$ | $2^{56.7}$ |
| FLURRY(32, 2, 8, f_5, D_2) | 16 | $5^{16} \approx 2^{37.15}$ | $O(2^{115.5})$ | $2^{80.3}$ |
| FLURRY(32, 2, 8, f_7, D_2) | 16 | $7^{16} \approx 2^{44.92}$ | $O(2^{138.8})$ | $2^{95.8}$ |
| FLURRY(16, 4, 4, f_3, D_2) | 16 | $3^{16} \approx 2^{25.36}$ | $O(2^{80.0})$ | $2^{55.7}$ |
| FLURRY(16, 4, 4, f_5, D_2) | 16 | $5^{16} \approx 2^{37.15}$ | $O(2^{115.5})$ | $2^{79.3}$ |
| FLURRY(16, 4, 4, f_7, D_2) | 16 | $7^{16} \approx 2^{44.92}$ | $O(2^{138.8})$ | $2^{94.8}$ |
| CURRY(32, 2, 3, f_7, D_2) | 12 | $7^{12} \approx 2^{33.69}$ | $O(2^{104.6})$ | $2^{73.0}$ |

considered we were able to estimate the complexity of a Gröbner basis conversion using the FGLM algorithm.

Acknowledgments. The authors would like to thank the anonymous referees for their comments. The third author acknowledges several fruitful discussions with Frederik Armknecht and Stefan Lucks.

References

1. Thomas Becker and Volker Weispfenning. *Gröbner Bases – A Computational Approach to Commutative Algebra*. Springer-Verlag, 1991.
2. Thomas Beth and Cunsheng Ding. On Almost Perfect Nonlinear Permutations. In Tor Helleseeth, editor, *Advances in Cryptology – EUROCRYPT ’93*, volume 765 of *Lecture Notes in Computer Science*, pages 65–76. Springer-Verlag, 1994.
3. Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. In Alfred Menezes and Scott A. Vanstone, editors, *Advances in Cryptology – CRYPTO ’90*, volume 537 of *Lecture Notes in Computer Science*, pages 2–21. Springer-Verlag, 1991.
4. Jung Hee Cheon, Seongtaek Chee, and Choonsik Park. S-boxes with Controllable Nonlinearity. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT ’99*, volume 1592 of *Lecture Notes in Computer Science*, pages 286–294. Springer-Verlag, 1999.
5. Carlos Cid and Gaëtan Laurent. An Analysis of the XSL Algorithm. In C. Pandu Rangan, editor, *Advances in Cryptology – ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 333–353. Springer-Verlag, 2005.
6. Carlos Cid, Sean Murphy, and Matt Robshaw. Small Scale Variants of the AES. In Helena Handschuh and Henri Gilbert, editors, *Fast Software Encryption – FSE 2005*, *Lecture Notes in Computer Science*, pages 145–162. Springer-Verlag, 2005.
7. Stéphane Collart, Michael Kalkbrener, and Daniel Mall. Converting Bases with the Gröbner Walk. *Journal of Symbolic Computation*, 24(3/4):465–469, 1997.

8. Nicolas Courtois. The Inverse S-box, Non-linear Polynomial Relations and Cryptanalysis of Block Ciphers. In Hans Dobbertin, Vincent Rijmen, and Aleksandra Sowa, editors, *AES 4 Conference*, volume 3373 of *Lecture Notes in Computer Science*, pages 170–188. Springer–Verlag, 2005.
9. 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–Verlag, 2002.
10. David A. Cox, John B. Little, and Don O’Shea. *Ideals, Varieties, and Algorithms*. Springer–Verlag, NY, 2nd edition, 1996. 536 pages.
11. Joan Daemen, Lars Knudsen, and Vincent Rijmen. The block cipher Square. In Eli Biham, editor, *Fast Software Encryption – FSE 1997*, volume 1267 of *Lecture Notes in Computer Science*, pages 149–165. Springer–Verlag, 1997.
12. Joan Daemen and Vincent Rijmen. *The Design of Rijndael: The Wide Trail Strategy*. Springer–Verlag, 2001.
13. Hans Dobbertin. One-to-One Highly Nonlinear Power Functions on $GF(2^n)$. *Applicable Algebra in Engineering, Communication and Computing*, 9(2):139–152, 1998.
14. Jean-Charles Faugère. A New Efficient Algorithm for Computing Gröbner bases (F4). *Journal of Pure and Applied Algebra*, 139(1-3):61–88, June 1999.
15. Jean-Charles Faugère, P. Gianni, Daniel Lazard, and Teo Mora. Efficient Computation of Zero-Dimensional Gröbner Bases by Change of Ordering. *Journal of Symbolic Computation*, 16(4):329–344, 1993.
16. Thomas Jakobsen and Lars Knudsen. The Interpolation Attack on Block Ciphers. In Eli Biham, editor, *Fast Software Encryption – FSE 1997*, volume 1267 of *Lecture Notes in Computer Science*, pages 28–40. Springer–Verlag, 1997.
17. Erich Kaltofen and Victor Shoup. Subquadratic-time Factoring of Polynomials over Finite Fields. *Mathematics of Computation*, 67(223):1179–1197, 1998.
18. Masayuki Kanda. Practical Security Evaluation against Differential and Linear Cryptanalyses for Feistel Ciphers with SPN Round Function. In Douglas R. Stinson and Stafford E. Tavares, editors, *Selected Areas in Cryptography – SAC 2000*, volume 2012 of *Lecture Notes in Computer Science*, pages 324–338. Springer–Verlag, 2001.
19. Lars R. Knudsen. Practically Secure Feistel Ciphers. In Ross J. Anderson, editor, *Fast Software Encryption – FSE 1993*, volume 809 of *Lecture Notes in Computer Science*, pages 211–221. Springer–Verlag, 1994.
20. M. Matsui. Linear Cryptanalysis Method for DES Cipher. In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO ’93*, volume 773 of *Lecture Notes in Computer Science*, pages 386–387. Springer–Verlag, 1994.
21. Sean Murphy and Matthew J.B. Robshaw. Essential Algebraic Structure within the AES. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 1–16. Springer–Verlag, 2002.
22. Kaisa Nyberg. Differentially Uniform Mappings for Cryptography. In Tor Helleseth, editor, *Advances in Cryptology – EUROCRYPT ’93*, volume 765 of *Lecture Notes in Computer Science*, pages 55–64. Springer–Verlag, 1994.
23. University of Sydney Computational Algebra Group. The Magma Computational Algebra System, 2004. <http://magma.maths.usyd.edu.au/magma/>.

A A DRL Gröbner basis for FLURRY(32, 2, 4, f_3 , D_2)

The following sequence of polynomials G is a degree-reverse lexicographic Gröbner basis for a FLURRY(32, 2, 4, f_3 , D_2) for the following variable ordering:

$$x_0 < x_1 < x_2 < x_3 < x_{16} < x_{17} < x_{18} < x_{19} < x_{14} < x_{15} < k_0 < k_1 < k_6 < k_7 < k_2 < k_3 < k_4 < k_5 < k_8 < k_9 < k_{10} < k_{11} < x_4 < x_5 < x_6 < x_7 < x_8 < x_9 < x_{10} < x_{11} < x_{12} < x_{13}$$

$$G = \{$$

plaintext:

$$x_0 + \theta^{31} + \theta^{29} + \theta^{27} + \theta^{24} + \theta^{22} + \theta^{21} + \theta^{19} + \theta^{13} + \theta^{11} + \theta^8 + \theta^7 + \theta^6 + \theta^4 + 1$$

$$x_1 + \theta^{31} + \theta^{30} + \theta^{29} + \theta^{22} + \theta^{21} + \theta^{15} + \theta^{14} + \theta^{11} + \theta^{10} + \theta^7 + \theta^6 + \theta^5 + \theta^3 + \theta$$

$$x_2 + \theta^{26} + \theta^{25} + \theta^{24} + \theta^{21} + \theta^{19} + \theta^{18} + \theta^{16} + \theta^{14} + \theta^8 + \theta^7 + \theta^6 + \theta^4 + \theta + 1$$

$$x_3 + \theta^{27} + \theta^{26} + \theta^{24} + \theta^{21} + \theta^{17} + \theta^{15} + \theta^{13} + \theta^{11} + \theta^9 + \theta^6 + \theta^4 + \theta$$

ciphertext:

$$x_{16} + \theta^{31} + \theta^{29} + \theta^{21} + \theta^{19} + \theta^{18} + \theta^{16} + \theta^{15} + \theta^{14} + \theta^{12} + \theta^4 + 1$$

$$x_{17} + \theta^{24} + \theta^{21} + \theta^{20} + \theta^{18} + \theta^{16} + \theta^{13} + \theta^{10} + \theta^9 + \theta^8 + \theta^6 + \theta^5 + \theta^3 + \theta + 1$$

$$x_{18} + \theta^{29} + \theta^{25} + \theta^{21} + \theta^{20} + \theta^{19} + \theta^{13} + \theta^{10} + \theta^9 + \theta^8 + \theta^7 + \theta^6 + \theta^5 + \theta^3$$

$$x_{19} + \theta^{29} + \theta^{27} + \theta^{26} + \theta^{20} + \theta^{13} + \theta^{10} + \theta^8 + \theta^5 + \theta^2$$

round 1:

$$x_4 + x_2$$

$$x_5 + x_3$$

$$k_0^3 + k_0^2 x_2 + k_0 x_2^2 + x_2^3 + C_1 x_7 + C_1 x_6 + C_1 x_1 + C_1 x_0$$

$$k_1^3 + k_1^2 x_3 + k_1 x_3^2 + x_3^3 + C_2 x_7 + C_1 x_6 + C_2 x_1 + C_1 x_0$$

round 2:

$$x_8 + x_6$$

$$x_9 + x_7$$

$$x_6^3 + x_6^2 k_2 + x_6 k_2^2 + k_2^3 + C_1 x_{11} + C_1 x_{10} + C_1 x_5 + C_1 x_4$$

$$x_7^3 + x_7^2 k_3 + x_7 k_3^2 + k_3^3 + C_2 x_{11} + C_1 x_{10} + C_2 x_5 + C_1 x_4$$

round 3:

$$x_{12} + x_{10}$$

$$x_{13} + x_{11}$$

$$x_{10}^3 + x_{10}^2 k_4 + x_{10} k_4^2 + k_4^3 + C_1 x_9 + C_1 x_8 + C_1 k_9 + C_1 k_8 + C_1 x_{15} + C_1 x_{14}$$

$$x_{11}^3 + x_{11}^2 k_5 + x_{11} k_5^2 + k_5^3 + C_2 x_9 + C_1 x_8 + C_2 k_9 + C_1 k_8 + C_2 x_{15} + C_1 x_{14}$$

round 4:

$$x_{14} + x_{16}$$

$$x_{15} + x_{17}$$

$$k_6^3 + k_6^2 x_{14} + k_6 x_{14}^2 + x_{14}^3 + C_1 x_{13} + C_1 x_{12} + C_1 k_{11} + C_1 k_{10} + C_1 x_{19} + C_1 x_{18}$$

$$k_7^3 + k_7^2 x_{15} + k_7 x_{15}^2 + x_{15}^3 + C_2 x_{13} + C_1 x_{12} + C_2 k_{11} + C_1 k_{10} + C_2 x_{19} + C_1 x_{18}$$

key expansion:

$$k_{11} + \theta^2 k_7 + (\theta^2 + \theta + 1)k_1 + \theta k_0 + \theta^4 + \theta^2$$

$$k_{10} + \theta^2 k_6 + \theta k_1 + k_0 + \theta^3 + \theta$$

$$k_9 + (\theta^2 + \theta)k_7 + (\theta + 1)k_6 + \theta^2 k_1 + (\theta + 1)k_0 + \theta^6 + \theta^5 + \theta^3 + 1$$

$$k_8 + (\theta + 1)k_7 + (\theta + 1)k_6 + (\theta + 1)k_1 + k_0 + \theta^5 + \theta^3$$

$$k_5 + (\theta^2 + \theta + 1)k_7 + \theta k_6 + \theta^2 k_1 + (\theta + 1)k_0 + \theta^6 + \theta^4 + \theta^3 + \theta$$

$$k_4 + \theta k_7 + k_6 + (\theta + 1)k_1 + k_0 + \theta^5 + \theta^4 + \theta^3 + 1$$

$$k_3 + \theta^2 k_7 + (\theta + 1)k_6 + (\theta^2 + \theta + 1)k_1 + \theta k_0 + \theta^6 + \theta^5 + \theta^4 + \theta$$

$$k_2 + (\theta + 1)k_7 + k_6 + \theta k_1 + k_0 + \theta^5 + \theta^2 + \theta + 1$$

}

with $C_1 = (\theta + 1)^{-1}$ and $C_2 = 1 + (\theta + 1)^{-1}$