Block ciphers sensitive to Gröbner Basis Attacks

Johannes Buchmann, Andrei Pychkine, Ralf-Philipp Weinmann
{buchmann,pychkine,weinmann}@cdc.informatik.tu-darmstadt.de

Technische Universität Darmstadt

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 neglegible 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.

Keywords: secret–key cryptography, cryptanalysis, block ciphers, algebraic attacks, Gröbner bases

1 Introduction

Since the publication of Courtois' and Pieprzyk's XSL method [1] and Murphy and Robshaw's embedding of AES-128 [2], 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 Rijndael-128 and Serpent have thus far been highly controversial. Gröbner bases however are a proven tool for solving polynomial systems. Cid, Murphy and Robshaw [3] 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.

To this end we present two families of ciphers, Flurry, a Feistel network and Curry, a SPN construction. For both we specify suitable parameters and give estimates on the complexity of attacks using linear and differential cryptanalysis. We explain how to obtain polynomial equations describing the key recovery problem and how Gröbner bases can be used to solve them. Experimental results are given 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 $F := GF(2^k)$ with $k \in \{8, 16, 32, 64\}$. 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 eth 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 eth 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 a simple algebraic representations. For these we present parameters sets offering high resistance against differential and linear cryptanalysis and describe how to construct a system of polynomial equations.

2.1 The Feistel case: Flurry

We construct the family Flurry(k, m, r, f, D) of Feistel ciphers. We explain the parameters used in this family:

- $m \in \mathbb{N}$: the plaintext space, the ciphertext space and the cipher key space are F^{2m}.
- $r \in \mathbb{N}$: the number of rounds
- $f : F \to F$: a non-linear mapping that gives the S-Box of the round function
- $D = (d_{i,j}) \in F^{m\times m}$: this matrix describes the linear diffusion mapping of the round function.

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

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

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

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

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

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

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

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

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

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

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 linear over $GF(2^k)$. 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:

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

2.2 The SPN case: Curry

In this section we construct a family Curry(k, m, r, f, D) of ciphers similar to Square [4]. We explain the parameters used in this family:

- $m \in \mathbb{N}$: the plaintext space, the ciphertext space and the cipher key space are $F^{m \times m}$.
- $r \in \mathbb{N}$: the number of rounds
- $f : F \to F$: a bijective non-linear mapping that gives 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 $\rho : F^{m \times m} \times F^{m \times m} \to F^{m \times m}$ of a Curry cipher is defined as:

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

with $G : F^{m \times m} \to F^{m \times m}$ 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:

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

Analogously, using the inverse 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:

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

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 linear over $GF(2^k)$. 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 ciphers are the S-Boxes. In order to obtain a cipher with good resistance against differential and linear cryptanalysis even for a low number of rounds the S-Boxes must be chosen very carefully. Two important characteristics of a S-Box are its differential uniformity and its nonlinearity, i.e. the distance of the S-Box function to an affine function. These are defined as follows:

Definition 1. *Let $f : F \to F$ 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 map

$$F \to GF(2)^k, a = \sum_{i=0}^{k-1} \left(a_i \theta^i\right) \mapsto (a_0, \ldots, a_{k-1})$$

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

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

Definition 2. *The nonlinearity of a function $f : F \mapsto F$ 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 k-uniformity and the nonlinearity have been well studied in the literature [5,6,7]. 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.

Table 1. S-Box mappings

function	mapping	bijective over $GF(2^k)$	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	$2^{k-1} - 2^{\frac{k}{2}}$
f_3	$x \mapsto x^3$	no	2	$\geq 2^{k-1} - 2^{\frac{k}{2}}$
f_5	$x \mapsto x^5$	no	4	$\geq 2^{k-1} - 2^{\frac{k}{2}+1}$
f_7	$x \mapsto x^7$	yes	≤ 6	$\geq 2^{k-1} - 3 \cdot 2^{\frac{k}{2}}$

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. For $k = 4$, f_7 is 4-uniform.

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 [5].

Lemma 2. *1. The nonlinearity of f_{-1} is $2^{k-2} - 2^{\frac{k}{2}}$.*
2. For a polynomial function $f : F \mapsto F$ 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 [7], for claim 2 see [8].

The linear transformations We use matrices of Maximum Distance Separable codes – *MDS matrices* for short – for the matrix D in the linear layer. 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 matrix used in the `MixColumns` step of Rijndael, D_2 is equivalent to a Pseudo-Hadamard Transform.

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

Rijmen and Daemen introduced the notion of *branch number* of a linear transformation to measure the quality of the diffusion provided. For a F-vector $X := (x_1, \ldots, x_m)$ 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 [9]:

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

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

The linear branch number $\mathcal{B}_l(M)$ is defined as

$$\mathcal{B}_l(M) = \min_{\substack{X \in F^m \\ X \neq 0}} \left(w(X) + w(M^T X)\right)$$

For symmetric matrices such as D_2, the linear and the differential branch number clearly coincide. For D_4 the linear and differential branch number coincide for a different reason [9]. Thus in our case it suffices to speak of *the branch number* $\mathcal{B}(M)$ of a matrix M. In [9] it is shown that the branch number of MDS matrices is maximal, i.e. $\mathcal{B}(M) = n + 1$ with n 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 a branch number of $\mathcal{B}(I_1) = 2$.

Note that the diffusion matrices are also used in the key expansion process.

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 $\lfloor \frac{tr}{2} \rfloor$ 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 $\lfloor \frac{tr}{2} \rfloor$ 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 $\left(\frac{2^k-1}{2^k}\right)^{mr}$. The equations in this case are of a different form:

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

  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:

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

  Using the above equations, the polynomial system also does not hold with probability one but with with probability $\left(\frac{2^k-1}{2^k}\right)^{m^2 r}$.
  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 $v \in (\mathcal{X} \cup \mathcal{K})$ the relation $v^{2^k} + v = 0$ 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 [10]. This type of attack makes use of statistical weaknesses of the first order derivative of the cipher. Selecting carefully chosen plaintext pairs with specific differences, the 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 developed by Matsui [11] to attack the DES. For this attack to succeed, the cryptanalyst has to construct a 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 first introduced by Knudsen [12]. 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 [13] 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 $\rho : GF(2)^n \to GF(2)^n$ 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*

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

In the above definition, a is called *input mask* and b is called *output mask* of a round. A vector of masks $A = (a_1, \ldots, a_{r+1})$ 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*

$$\mathrm{DP}(\Delta x, \Delta y) = \mathrm{Pr}_X \{\rho(X) + \rho(X + \Delta x) = \Delta y\}$$

The value Δx is called *input difference* of a round, while Δy is called *output difference.* A vector of differences $A = (a_1, \ldots, a_{r+1})$ 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*

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

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

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

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 deduce the number of rounds required to make an instance practically secure against differential and linear cryptanalysis.

The maximum differential probability of a function $f : F \to F$ 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 $f : F \to F$ 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 both SPN ciphers and Feistel ciphers with an SP round function the MDCP is bounded by $p(f)^N$ and the MLCP is bounded by $q(f)^N$ [13].

According to Knudsen [12], 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 [14] as a counterpoint to the growing trend of using algebraic S-Boxes such as those proposed by Nyberg. 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 always 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 can thus be found. Moreover, should the size of a round key be smaller than the size of the cipher key, the cipher key can be recovered by applying the attack to all but the last round and verifying the guesses for the last round key using the resulting tuple of polynomials – in effect peeling off the final round. This obviously works for Feistel ciphers.

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 $\mathcal{PURE}$ presented in the original article is identical to the 64-bit cipher Flurry$(32, 1, I_1, f_3, r)$ sans key scheduling.

Jakobsen and Knudsen give upper bounds on the number of required pairs for known-plaintext interpolation attacks for selected examples. 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 this remains a fairly constant quantity.

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 small number of plaintext/ciphertext pairs faster than an 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 interested reader to [15] and [16]. In the following we will use the conventions of [15].

Definition 7 (Term order). *A term order $<$ is a linear order on the set of terms such that*

1. $1 \leq t$ *for all terms* t
2. *for all terms* s, t_1, t_2 *whenever* $t_1 < t_2$ *then* $st_1 < st_2$

If a term order has been fixed, we define $HT(f)$ to be the greatest term occuring in the polynomial $f \in R$ according to this order; this term is called the head term. Correspondingly $HM(f)$ 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 $t = v_1^{e_1} v_2^{e_2} \cdots v_n^{e_n}$ we define the *exponent vector* of t to be $\epsilon(t) = (e_1, e_2, \ldots, e_n) \in \mathbb{N}_0^n$. The total degree of the term t then is $\deg(t) = \sum_{i=1}^{n} e_i$.

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

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

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

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

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 uniquely defined if G is not 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 $G = \{g_1, \ldots, g_n\} \subset R$, if there exists a vector of polynomials $(m_1, \ldots, m_n)$ such that $f - \sum_{i=1}^{n} m_i g_i = 0$ with $HT(m_i g_i) \leq HT(f)$ for all $1 \leq i \leq n$.*

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

Let $\mathcal{P}$ be a set of multivariate polynomial equations $p_i = 0$. For the ideal $\mathfrak{I}$ 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. Several algorithms for this task are known; the two most prominent are the FGLM algorithm [17] and the Gröbner Walk [18]. 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

In general, estimating the time and space complexity of Gröbner basis algorithms is a hard problem. For the systems of equations coming from the cryptosystem HFE, Faugere and Joux were able to explain the good performance of the F5 algorithm [19] solving these systems [20]. 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 $\mathcal{G} = \{g_i = 0\}$:

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

   Let $\mathfrak{I}$ 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 $\mathfrak{I}$. 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 $\mathfrak{I}$ 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 still 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. Of course 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 standard algorithm for finding roots of univariate polynomials is the Berlekamp algorithm which has a run-time complexity of $O(d^3)$ with d being the degree of the polynomial; this degree of course is bounded by $2^k - 1$. The best algorithm for factoring polynomials is due to Kaltofen and Shoup [21] and has a complexity of $O(d^{1.815}k)$ field operations. 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 [22], version 2.11-8, on an AMD Athlon 64 3200+ Linux PC equipped with 1024 Megabytes of RAM. MAGMA implements Faugére's F4 algorithm [23] and is widely considered the best publicly available tool for computing Gröbner bases. We have chosen k 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. 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. We would like to point out that the 6, 8 and 10 round Flurry ciphers listed below are resistant to linear and differential cryptanalysis.

Table 2. Experimental results achieved with Magma

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

As mentioned in Section 2.4 we did not add the field equations to our polynomial systems. In our case, these equations are of extremely high degree. Generally it only makes sense to add these additional relations if the ground field is very small.

4.3 Gröbner bases without polynomial reductions

In some cases it is possible to directly 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 $0 \neq G$.

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

$$lcm(HT(f), HT(g)) = HT(f) \cdot HT(g)$$

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

Proposition 1 is the first Buchberger criterion. Together with the following theorem given in [16], 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 $spol(f, g) \rightarrow_G 0$ for all $f, g \in G$ with $f \neq g$.*

For the case of polynomial S-boxes, this enables us to compute degree-reverse lexographic Gröbner bases of key-recovery ideals of Flurry and Curry without performing a single polynomial reduction. Observe that for a total-degree order the head terms of all polynomials of $\mathfrak{I}$ are univariate. For each polynomial of round e, either a power of a state variable of the preceeding round or a power of a key variable of the current round occur as head term. However some head terms occur more than once.

By using an appropriate degree reverse lexicographical term order we can force the set of head terms of each round to be disjunct from the set of head terms of all other rounds:

- Curry
  To make the variable ordering more legible, we reduce the number of indexing of our variables: 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:

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

- 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:

$$\underbrace{x_0 < \ldots < x_{t-1}}_{\text{plaintext variables}} < \underbrace{x_{tr} < \ldots < x_{(t+1)r-1}}_{\text{ciphertext variables}} < \underbrace{x_{t(r-1)+m} < \ldots < x_{tr-1}}_{\substack{\text{state variables of the right Feistel}\\ \text{half of the second last round}}} < \underbrace{k_0 < \ldots < k_{m-1}}_{\substack{\text{key variables of}\\ \text{the first round}}} <$$

$$\underbrace{k_{m(r-1)} < \ldots < k_{mr-1}}_{\text{key variables of round } r} < \underbrace{k_m < \ldots < k_{m(r-1)-1} < k_{mr} < \ldots < k_{m(r+2)-1}}_{\text{all other key variables}} < \underbrace{x_t < \ldots < x_{t(r-1)+m-1}}_{\text{state variables of all other rounds}}$$

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. Observe that 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 round last key have the same head terms. To remedy this situation we need to rewrite the key schedule equations. We express all round keys but the last round key as a linear combination of the first two round keys and can then 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. The trick 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

For the FGLM algorithm, we will show how to estimate the complexity of a Gröbner basis conversion. The running time mainly depends on two parameters, the vector space dimension of the ideal $\mathfrak{I}$ generated by the Gröbner basis $G \subset R$ and the number of variables of the polynomial ring R.

Definition 11. *Let $R := F[X]$. For an ideal $\mathfrak{I} \subset R$ the dimension of the F-vector space of the ideal $R/\mathfrak{I}$ is denoted by* $\dim(R/\mathfrak{I})$.

The following theorem shows how this invariant of an ideal can be computed in general.

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

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

Corollary 1. *Let $G = \{g_1, \ldots, g_n\}$ be a Gröbner basis for the ideal $\mathfrak{I} \subset F[x_1, \ldots, x_n]$ with head terms $x_1^{d_1}, \ldots, x_n^{d_n}$. Then* $\dim(R/\mathfrak{I}) = d_1 \cdot \cdots \cdot d_n$.

Corollary 2. *Let $\mathfrak{I}$ 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. *$dim(R/\mathfrak{I}) = \deg(f)^{mr}$ for* $\text{Flurry}_{k,m,f,r}$.
2. *$dim(R/\mathfrak{I}) = \deg(f)^{m^2r}$ for* $\text{Curry}_{k,m,f,r}$.

We restate Theorem 5.1 of [17].

Theorem 3. *Let K be a finite field and $R = K[x_1, \ldots, x_n]$. Furthermore $G_1 \subset R$ is the Gröebner basis relative to a term order $<_1$ of an ideal I, and $D = dim(R/I)$. We can then convert G_1 into a Gröbner basis G_2 relative to a term order $<_2$ in $O(nD^3)$ field operations.*

In general we can assume the constant factor for the above estimate to be small. 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 $D \times nD$ matrix over F. Thus the memory usage of the algorithm is upper bounded by $\lceil (nD^2k)/8 \rceil$ bytes.

This allows us to estimate the maximum resistance of Flurry and Curry ciphers with polynomial S-Boxes against Gröbner basis attacks. Note that for the Curry cipher we need to use a bijective S-Box in the round function; the lowest degree S-Box function in our paper that is bijective is f_7.

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)
$\text{Flurry}_{32,2,4,f_3,D_2}$	8	$3^8 \approx 2^{12.68}$	$O(2^{41.0})$	$2^{30.4}$
$\text{Flurry}_{32,2,4,f_5,D_2}$	8	$5^8 \approx 2^{18.58}$	$O(2^{58.7})$	$2^{42.2}$
$\text{Flurry}_{32,2,4,f_7,D_2}$	8	$7^8 \approx 2^{22.46}$	$O(2^{70.4})$	$2^{49.9}$
$\text{Flurry}_{32,2,6,f_3,D_2}$	12	$3^{12} \approx 2^{19.02}$	$O(2^{60.6})$	$2^{43.2}$
$\text{Flurry}_{32,2,6,f_5,D_2}$	12	$5^{12} \approx 2^{27.86}$	$O(2^{87.2})$	$2^{61.3}$
$\text{Flurry}_{32,2,6,f_7,D_2}$	12	$7^{12} \approx 2^{33.69}$	$O(2^{104.7})$	$2^{73.0}$
$\text{Flurry}_{32,2,8,f_3,D_2}$	16	$3^{16} \approx 2^{25.36}$	$O(2^{80.0})$	$2^{56.7}$
$\text{Flurry}_{32,2,8,f_5,D_2}$	16	$5^{16} \approx 2^{37.15}$	$O(2^{115.5})$	$2^{80.3}$
$\text{Flurry}_{32,2,8,f_7,D_2}$	16	$7^{16} \approx 2^{44.92}$	$O(2^{138.8})$	$2^{95.8}$
$\text{Flurry}_{16,4,4,f_3,D_2}$	16	$3^{16} \approx 2^{25.36}$	$O(2^{80.0})$	$2^{55.7}$
$\text{Flurry}_{16,4,4,f_5,D_2}$	16	$5^{16} \approx 2^{37.15}$	$O(2^{115.5})$	$2^{79.3}$
$\text{Flurry}_{16,4,4,f_7,D_2}$	16	$7^{16} \approx 2^{44.92}$	$O(2^{138.8})$	$2^{94.8}$
$\text{Curry}_{32,2,3,f_7,D_2}$	12	$7^{12} \approx 2^{33.69}$	$O(2^{104.6})$	$2^{73.0}$

For the Gröbner Walk we are unable to theoretically gauge the complexity. The running time for this algorithm heavily depends on the source and target term order, whereas for FGLM it only depends on invariants of the ideal. Kalkbrenner gives degree bounds [24] that can be applied to the Gröbner Walk; these are very loose however. The best bound known a result about the growth of the degrees in each step; it can be shown to be at most quadratic. Estimating the global complexity for the Gröbner Walk seems to be a hard problem.

5 Conclusions

We have demonstrated that Gröbner basis algorithms can be used to successfully mount key-recovery attacks on algebraically simple block ciphers with large block and key size, even when they are practically secure against differential and linear cryptanalysis. This can be accomplished with a minimal number of known plaintext/ciphertext pairs. Furthermore we have demonstrated that Gröbner bases for ciphers that follow our construction principle can be calculated by hand. These Gröbner bases are relative to a total-degree order instead of a lexicographical order and thus do not give the solution to the polynomial system directly. However, 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 closed formula for the vector space dimension of the ideal of all inversion-free ciphers considered we were able to estimate the complexity of a Gröbner basis conversion using the FGLM algorithm. This allowed us to determine more instances of Flurry and Curry ciphers that are vulnerable against Gröbner basis attacks.

For the Gröbner Walk algorithm, it is an open problem to give bounds on its time and space complexity.

A Appendix

Below we give upper bounds for the maximum differential and linear characteristic probability for selected Curry and Flurry ciphers. Since we have fixed the diffusion matrices in Section 2.3, the actual matrix to be used only depends on the parameter m. For space reasons this is not explicitly listed in the table.

Table 4. MDCP for selected 128 bit Flurry ciphers

Flurry$_{k,f,m,r,D}$									
				$(64, f_{-1}, 1)$	$(32, f_{-1}, 2)$	$(16, f_{-1}, 4)$			
(k, f, m)	$(64, f_3, 1)$	$(32, f_3, 2)$	$(16, f_3, 4)$	$(64, f_5, 1)$	$(32, f_5, 2)$	$(16, f_5, 4)$	$(64, f_7, 1)$	$(32, f_7, 2)$	$(16, f_7, 4)$
p_s	2^{-63}	2^{-31}	2^{-15}	2^{-62}	2^{-30}	2^{-14}	$3 \cdot 2^{-63}$	$3 \cdot 2^{-31}$	$3 \cdot 2^{-15}$
r=4	2^{-126}	2^{-93}	2^{-75}	2^{-124}	2^{-90}	2^{-70}	$< 2^{-122}$	$< 2^{-88}$	$< 2^{-53}$
r=6	2^{-252}	2^{-155}	2^{-105}	2^{-248}	2^{-150}	2^{-98}	$< 2^{-245}$	$< 2^{-147}$	$< 2^{-80}$
r=8	2^{-315}	2^{-217}	2^{-165}	2^{-310}	2^{-210}	2^{-154}	$< 2^{-307}$	$< 2^{-205}$	$< 2^{-120}$

Table 5. MLCP for selected 128 bit Flurry ciphers

Flurry$_{k,f,m,r,D}$									
	$(f_{-1}, 1)$	$(f_{-1}, 2)$	$(f_{-1}, 4)$						
(k, f, m)	$(f_3, 1)$	$(f_3, 2)$	$(f_3, 4)$	$(f_5, 1)$	$(f_5, 2)$	$(f_5, 4)$	$(f_7, 1)$	$(f_7, 2)$	$(f_7, 4)$
q_s	2^{-62}	2^{-30}	2^{-14}	2^{-60}	2^{-28}	2^{-12}	$9 \cdot 2^{-62}$	$9 \cdot 2^{-30}$	$9 \cdot 2^{-14}$
r=4	2^{-124}	2^{-90}	2^{-56}	2^{-120}	2^{-84}	2^{-60}	$< 2^{-117}$	$< 2^{-80}$	$< 2^{-43}$
r=6	2^{-248}	2^{-150}	2^{-84}	2^{-240}	2^{-140}	2^{-84}	$< 2^{-235}$	$< 2^{-134}$	$< 2^{-64}$
r=8	2^{-310}	2^{-210}	2^{-126}	2^{-300}	2^{-196}	2^{-132}	$< 2^{-294}$	$< 2^{-187}$	$< 2^{-97}$

Table 6. MDCP for selected 128 bit Curry ciphers

Curry$_{k,f,m,r,D}$			
(k, f, m)	$(32, f_{-1}, 2)$	$(8, f_{-1}, 4)$	$(32, f_7, 2)$
p_s	2^{-30}	2^{-6}	$3 \cdot 2^{-31}$
r=4	2^{-180}	2^{-60}	$< 2^{-176}$
r=6	2^{-270}	2^{-90}	$< 2^{-264}$
r=8	2^{-360}	2^{-120}	$< 2^{-352}$

Table 7. MLCP for selected 128 bit Curry ciphers

Curry$_{k,f,m,r,D}$			
	$(32, f_{-1}, 2)$	$(8, f_{-1}, 4)$	
(k, f, m)	$(32, f_{-2}, 2)$	$(8, f_{-2}, 4)$	$(32, f_7, 2)$
q_s	2^{-30}	2^{-6}	$9 \cdot 2^{-30}$
r=4	2^{-180}	2^{-60}	$< 2^{-160}$
r=6	2^{-270}	2^{-90}	$< 2^{-241}$
r=8	2^{-360}	2^{-120}	$< 2^{-321}$

B Appendix

Example 3. The following sequence of polynomials G is a Gröbner basis for a $\textsc{Flurry}(32, 2, D_2, f_3, 4)$ w.r.t. the degree-reverse lexicographical order. Variables are ordered as follows:
$x_0 < x_1 < x_2 < x_3 < x_{16} < x_{17} < x_{18} < x_{19} < x_{14} < x_{15} < k_0 < k_1 < k_6 < k_7 < k_2 < k_3 < k_4 < k_5 < k_8 < k_9 < k_{10} < k_{11} < x_4 < x_5 < x_6 < x_7 < x_8 < x_9 < x_{10} < x_{11} < x_{12} < x_{13}$

$G = \{$

`plaintext:`

$x_0 + \theta^{31} + \theta^{29} + \theta^{27} + \theta^{24} + \theta^{22} + \theta^{21} + \theta^{19} + \theta^{13} + \theta^{11} + \theta^{8} + \theta^{7} + \theta^{6} + \theta^{4} + \theta^{3} + 1$

$x_1 + \theta^{31} + \theta^{30} + \theta^{29} + \theta^{22} + \theta^{21} + \theta^{15} + \theta^{14} + \theta^{11} + \theta^{10} + \theta^{7} + \theta^{6} + \theta^{5} + \theta^{3} + \theta^{2} + \theta$

$x_2 + \theta^{31} + \theta^{29} + \theta^{27} + \theta^{26} + \theta^{25} + \theta^{24} + \theta^{21} + \theta^{19} + \theta^{18} + \theta^{16} + \theta^{15} + \theta^{14} + \theta^{8} + \theta^{7} + \theta^{6} + \theta^{4} + \theta + 1$

$x_3 + \theta^{27} + \theta^{26} + \theta^{24} + \theta^{22} + \theta^{21} + \theta^{20} + \theta^{18} + \theta^{17} + \theta^{15} + \theta^{13} + \theta^{11} + \theta^{9} + \theta^{6} + \theta^{4} + \theta$

`ciphertext:`

$x_{16} + \theta^{31} + \theta^{29} + \theta^{28} + \theta^{27} + \theta^{26} + \theta^{24} + \theta^{21} + \theta^{19} + \theta^{18} + \theta^{16} + \theta^{15} + \theta^{14} + \theta^{12} + \theta^{4} + 1$

$x_{17} + \theta^{28} + \theta^{26} + \theta^{24} + \theta^{21} + \theta^{20} + \theta^{18} + \theta^{16} + \theta^{13} + \theta^{10} + \theta^{9} + \theta^{8} + \theta^{6} + \theta^{5} + \theta^{3} + \theta + 1$

$x_{18} + \theta^{29} + \theta^{25} + \theta^{21} + \theta^{20} + \theta^{19} + \theta^{17} + \theta^{16} + \theta^{15} + \theta^{14} + \theta^{13} + \theta^{10} + \theta^{9} + \theta^{8} + \theta^{7} + \theta^{6} + \theta^{5} + \theta^{3}$

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

`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 $C_2 = 1 + (\theta + 1)^{-1}$

References

1. Courtois, N., Pieprzyk, J.: Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. In Zheng, Y., ed.: ASIACRYPT. Volume 2501 of Lecture Notes in Computer Science., Springer (2002) 267 – 287
2. Murphy, S., Robshaw, M.J.: Essential Algebraic Structure within the AES. In Yung, M., ed.: CRYPTO. Volume 2442 of Lecture Notes in Computer Science., Springer (2002) pp. 1 – 16
3. Cid, C., Murphy, S., Robshaw, M.: Small Scale Variants of the AES. In: Fast Software Encryption, 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005. Lecture Notes in Computer Science, Springer (2005)
4. Daemen, J., Knudsen, L., Rijmen, V.: The block cipher Square. [25] 149 – 165
5. Nyberg, K.: Differentially Uniform Mappings for Cryptography. [26] 55–64
6. Beth, T., Ding, C.: On Almost Perfect Nonlinear Permutations. [26] 65–76
7. Dobbertin, H.: One-to-One Highly Nonlinear Power Functions on $GF(2^n)$. Applicable Algebra in Engineering, Communication and Computing **9** (1998) 139–152
8. Cheon, J.H., Chee, S., Park, C.: S-boxes with Controllable Nonlinearity. In Stern, J., ed.: EUROCRYPT. Volume 1592 of Lecture Notes in Computer Science., Springer (1999) 286–294
9. Daemen, J., Rijmen, V.: The Design of Rijndael: The Wide Trail Strategy. Springer-Verlag (2001)
10. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. In Menezes, A., Vanstone, S.A., eds.: CRYPTO. Volume 537 of Lecture Notes in Computer Science., Springer (1991) 2 – 21
11. Matsui, M.: Linear Cryptanalysis Method for DES Cipher. In Stinson, D.R., ed.: CRYPTO. Volume 773 of Lecture Notes in Computer Science., Springer (1994) 386 – 387
12. Knudsen, L.R.: Practically Secure Feistel Ciphers. In Anderson, R.J., ed.: FSE. Volume 809 of Lecture Notes in Computer Science., Springer (1994) 211–221
13. Kanda, M.: Practical Security Evaluation against Differential and Linear Cryptanalyses for Feistel Ciphers with SPN Round Function. In Stinson, D.R., Tavares, S.E., eds.: Selected Areas in Cryptography. Volume 2012 of Lecture Notes in Computer Science., Springer (2001) 324–338
14. Jakobsen, T., Knudsen, L.: The Interpolation Attack on Block Ciphers. [25] 28–40
15. Becker, T., Weispfenning, V.: Gröbner Bases – A Computational Approach to Commutative Algebra. Springer-Verlag (1991)
16. Cox, D.A., Little, J.B., O'Shea, D.: Ideals, Varieties, and Algorithms. 2nd edn. Springer-Verlag, NY (1996) 536 pages.
17. Faugère, J.C., Gianni, P., Lazard, D., Mora, T.: Efficient Computation of Zero-Dimensional Gröbner Bases by Change of Ordering. Journal of Symbolic Computation **16** (1993) 329–344
18. Collart, S., Kalkbrener, M., Mall, D.: Converting Bases with the Gröbner Walk. Journal of Symbolic Computation **24** (1997) 465–469
19. Faugère, J.C.: A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In: ISSAC, ACM (2002) pp. 75–83
20. Faugère, J.C., Joux, A.: Algebraic Cryptanalysis of Hidden Field Cryptosystems Using Gröbner Bases. In Boneh, D., ed.: CRYPTO. Volume 2729 of Lecture Notes in Computer Science., Springer (2003) pp. 44–60
21. Kaltofen, E., Shoup, V.: Subquadratic-time Factoring of Polynomials over Finite FIelds. Mathematics of Computation **67** (1998) 1179–1197
22. University of Sydney Computational Algebra Group: The Magma Computational Algebra System (2004) http://magma.maths.usyd.edu.au/magma/.
23. Faugère, J.C.: A New Efficient Algorithm for Computing Gröbner bases (F4). Journal of Pure and Applied Algebra **139** (1999) 61–88
24. Aoki, K.: Efficient Evaluation of Security against Generalized Interpolation Attack. In Heys, H.M., Adams, C.M., eds.: Selected Areas in Cryptography. Volume 1758 of Lecture Notes in Computer Science., Springer (2000) 135–146
25. Biham, E., ed.: Fast Software Encryption, 4th International Workshop, FSE '97, Haifa, Israel, January 20-22, 1997, Proceedings. In Biham, E., ed.: FSE. Volume 1267 of Lecture Notes in Computer Science., Springer (1997)
26. Helleseth, T., ed.: Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings. In Helleseth, T., ed.: EUROCRYPT. Volume 765 of Lecture Notes in Computer Science., Springer (1994)