Algebraic Cryptanalysis of Curry and Flurry using Correlated Messages

Jean-Charles Faugère and Ludovic Perret

SALSA Project
INRIA, Centre Paris-Rocquencourt
UPMC, Univ Paris 06, LIP6
CNRS, UMR 7606, LIP6
104, avenue du Président Kennedy
75016 Paris, France
jean-charles.faugere@inria.fr, ludovic.perret@lip6.fr

Abstract. In [10], Buchmann, Pyshkin and Weinmann have described two families of Feistel and SPN block ciphers called **Flurry** and **Curry** respectively. These two families of ciphers are fully parametrizable and have a sound design strategy against basic statistical attacks; i.e. linear and differential attacks. The encryption process can be easily described by a set of algebraic equations. These ciphers are then targets of choices for algebraic attacks. In particular, the key recovery problem has been reduced to changing the order of a Gröbner basis [10, 11]. This attack – although being more efficient than linear and differential attacks – remains quite limited. The purpose of this paper is to overcome this limitation by using a small number of suitably chosen pairs of message/ciphertext for improving algebraic attacks. It turns out that this approach permits to go one step further in the (algebraic) cryptanalysis of **Flurry** and **Curry**. To explain the behavior of our attack, we have established an interesting connection between algebraic attacks and high order differential cryptanalysis [21]. From extensive experiments, we estimate that our approach, that we can call "algebraic-high order differential" cryptanalysis, is polynomial when the Sbox is a power function. As a proof of concept, we have been able to break **Flurry**s – up to 8 rounds – in few hours.

1 Introduction

The basic principle of an *algebraic cryptanalysis* [13] is to model a cryptographic primitive by a set of algebraic equations. The system is constructed in such a way as to have a correspondence between the solutions of this system, and a secret information of the primitive. For block ciphers, the goal is usually to recover the secret key. It has to be noted that this line of research is somehow inspired by C.E. Shannon who stated that : *"Breaking a good cipher should require as much work as solving a system of simultaneous equations in a large number of unknowns of a complex type."* (Communication Theory of Secrecy Systems, 1949).

A cryptosystem can always be modeled by an algebraic set of equations. In practice, the same cryptographic primitive can even be described by several algebraic systems. Unfortunately, not all of these models yield "good" systems (from a system-solving point of view). One of the most crucial aspects here is to derive the best systems. In the context of block ciphers, the algebraic system is generated from the knowledge of one pair message/ciphertext. In this paper, we investigate the natural possibility [24, 5] of using a small number of suitably chosen pairs of message/ciphertext for improving algebraic attacks.

Once a model is chosen, the problem is to solve the algebraic system (or to evaluate the difficulty of solving such a system). Gröbner bases [8, 9] yield so far the most suitable algorithmic solution for solving algebraic systems of equations. To date, F_5 is the most efficient method for computing Gröbner bases [19].

The practical efficiency of algebraic attacks against modern block ciphers are quite difficult to predict; and remains so far quite limited. For this reason, Cid, Murphy, and Robshaw [14] described small scale variants of AES. The goal was to understand the behavior of algebraic attacks. In the same vain, Buchmann, Pyshkin and Weinmann [10] described two families of Feistel (**Flurry**) and SPN (**Curry**) block ciphers which are fully parametrizable and provide good resistance against linear and differential attacks. These ciphers are targets of choices for experimenting algebraic attacks. The authors of [10] proved that the key-recovery problem can be reduced to the problem of changing the order of a Gröbner basis. This permits to obtain a precise complexity for their approach; which is quite rare in algebraic cryptanalysis. In this paper, we focus our attention to these two families of ciphers.

1.1 Organization of the Paper. Main Results.

After this introduction, the paper is organized as follows. In Section 2, we introduce the families of Feistel and SPN block ciphers **Flurry** and **Curry** respectively [10, 11]. We briefly recall some security characteristics of the ciphers.

The basic problem of algebraic attacks is to find the zeros of a set of non-linear system of equations. In Section 3, we introduce the necessary mathematical tools (ideals, varieties and Gröbner bases), as well as the algorithmic tools (FGLM, and[1] F_4/F_5), allowing to address the problem of system solving. The reader already familiar with these objects can skip this part. However, we would like to emphasize that the material contained in this section is essential for understanding the practical behavior of the attacks presented in Section 4.

In this last section (Section 4), we will present results that we have obtained when mounting two refined algebraic attack strategies. First, we show that the use of a sparse version of FGLM permits to obtain a practical gain w.r.t. to the attack presented by Buchmann, Pyshkin and Weinmann in [10]. However, this attack remains limited since its theoretical complexity is exponential in the number of rounds and the size of the plaintext space. To overcome this limitation, we

[1] The description of F_4/F_5 is postponed in the Appendix.

have investigates the possibility of using a small amount of suitably chosen pairs $(message, ciphertext)$ to improve the efficiency of algebraic attacks. Precisely, we propose to use correlated messages. It appears that this approach permits to go one step further in the (algebraic) cryptanalysis of **Flurry** and **Curry**. To explain the behavior of our attack, we have established an interesting connection between algebraic attacks and high order differential cryptanalysis [21]. From our experiments, we estimate that if Sbox function if defined by a polynomial function f, then this last approach is polynomial in the degree of f. For instance, we have been able to break a **Flurry** up to 9 rounds in few hours. We also present experimental results for the inverse SBox.

2 The Flurry and Curry Block Ciphers

In this part, we describe the main concern of this paper, namely **Curry** and **Flurry** [10, 11]. We will also briefly recall the algebraic description of such ciphers, and highlight some security arguments of these block ciphers. In the following $\mathbb{K} = \mathbb{F}_2(\theta)$ is a finite field and $k = 2^n, n \in \{8, 16, 32, 64\}$, is the size of $\mathbb{K}$; $r \in \mathbb{N}$ is the number of rounds; $D \in \mathcal{M}_{m\times m}(\mathbb{K})$ is a matrix describing the linear diffusion mapping of the round function. The matrix D is also used in the key scheduling. Such matrices have been chosen to have an optimal diffusion strategy. We refer to [10] for the exact description of the matrices chosen. f is a non-linear function describing the Sbox. Here, we will consider :
– the power function $f(x) = f_p(x) : x \in \mathbb{K} \mapsto x^p \in \mathbb{K}$, with $p \in \{3, 5, 7\}$,
– or the inverse function $f(x) = f_{\text{inv}}(x) : x \in \mathbb{K} \mapsto x^{k-2} \in \mathbb{K}$.

2.1 The Feistel Case : Flurry

First we describe the r round Feistel cipher: **Flurry**(n, t, r, f, D) where $t \in \mathbb{N}$ is the size of a message block. We will also use the notation $m = \frac{t}{2}$ for the half-size of a block. We will denote by $L = (\ell_1, \ldots, \ell_m) \in \mathbb{K}^m$ $\big($resp. $R = (r_1, \ldots, r_m) \in \mathbb{K}^m$ $\big)$ the left (resp. right) part of the current state, and by $K = (k_1, \ldots, k_m) \in \mathbb{K}^m$ a key. The round function $T : \mathbb{K}^m \times \mathbb{K}^m \times \mathbb{K}^m \to \mathbb{K}^m \times \mathbb{K}^m$ is :

$$T(L, R, K) = \big(R, \big(f(r_1 + k_1), \ldots, f(r_m + k_m)\big) \cdot D\big).$$

Let $(K_0, K_1) \in \mathbb{K}^m \times \mathbb{K}^m$ be the initial key. The subkey used at round $i, 2 \le i \le r + 1$ is :

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

where $v_i = \big((\theta + 1)^i, (\theta + 1)^{i+1}, \ldots, (\theta + 1)^{i+m-1}\big) \in \mathbb{K}^m$. A message $m = (L_0, R_0) \in \mathbb{K}^m \times \mathbb{K}^m$ is encrypted into a ciphertext $c = (L_r, R_r) \in \mathbb{K}^m \times \mathbb{K}^m$ by iterating the round function T as follows :

$$\begin{aligned}(L_i, R_i) &= T(L_{i-1}, R_{i-1}, K_{i-1}), \text{ for all } i, 1 \le i \le r - 1,\\ c = (L_r, R_r) &= T(L_{r-1}, R_{r-1}, K_{r-1}) + (K_r, K_{r+1}).\end{aligned}$$

It is not difficult to describe the encryption process by a set of algebraic equations. To do so, we have to introduce new variables :
– $\{x_{i,j}\}_{1\leq i\leq (r-1)}^{1\leq j\leq t}$ corresponding to the internal states of the cipher,
– and $\{k_{i,j}\}_{1\leq i\leq r+1}^{1\leq j\leq m}$ corresponding to the initial/expanded key.
We will denote by $\mathcal{R}_{\text{Flurry}}$ the polynomial ring $\mathbb{K}\big[\{x_{i,j}\}_{1\leq i\leq (r-1)}^{1\leq j\leq t}, \{k_{i,j}\}_{1\leq i\leq r+1}^{1\leq j\leq m}\big]$. For a pair plaintext/ciphertext $(m,c)\in\mathbb{K}^t\times\mathbb{K}^t$, we will denote by :

$$\mathcal{P}_{\text{Flurry}}(m,c)\subset\mathcal{R}_{\text{Flurry}},$$

the set of all algebraic equations describing **Flurry**'s encryption process.

2.2 The SPN Case : Curry

The family of SPN ciphers **Curry**(n,m,r,f,D) is parameterized by $m\in\mathbb{N}$ is the plaintext space dimension; the ciphertext and secret key spaces are $\mathbb{K}^{m\times m}$. The round function $T:\mathbb{K}^{m\times m}\times\mathbb{K}^{m\times m}\to\mathbb{K}^{m\times m}$ of Curry is given by :

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

with $G: X=\{x_{i,j}\}\in\mathbb{K}^{m\times m}\to G(X)=\{f(x_{i,j})\}\in\mathbb{K}^{m\times m}$ being the parallel application of the SBox f to the components of a matrix X.

A plaintext $m=S_0\in\mathbb{K}^{m\times m}$ is encrypted into a ciphertext $c\in\mathbb{K}^{m\times m}$ by iterating the round function T exactly r times followed by a last key addition :

$$\begin{aligned} S_\ell &= T(S_{\ell-1},K_{\ell-1}),\ \text{for all } \ell, 1\leq\ell\leq r-1,\\ c=S_r &= T(S_{r-1},K_{r-1})+K_r. \end{aligned}$$

The master key $K_0\in K^{m\times m}$ is used at the first round; subsequent round keys $K_i, i\geq 1$ are computed using the formula : $K_i=K_{i-1}\cdot D+M_i, M_i=\{\theta^{i+(j-1)m+k}\}_{1\leq j,k\leq m}\in\mathbb{K}^{m\times m}$ being a round constant.

As for **Flurry**, the encryption process of **Curry** can be described symbolically; introduce new variables : $\{x_{i,j}^\ell\}_{1\leq i,j\leq m}^{1\leq\ell\leq(r-1)}$ corresponding to the internal states of the cipher, and $\{k_{i,j}^\ell\}_{1\leq\ell\leq r}^{1\leq i,j\leq m}$ corresponding to the initial/expanded key. Using an obvious notation, we will denote by $\mathcal{R}_{\text{Curry}}$ the polynomial ring $\mathbb{K}\left[\{x_{i,j}^\ell\}_{1\leq i,j\leq m}^{1\leq\ell\leq(r-1)}, \{k_{i,j}^\ell\}_{1\leq\ell\leq r}^{1\leq i,j\leq m}\right]$, and $\mathcal{P}_{\text{Curry}}(m,c)\subset\mathcal{R}_{\text{Curry}}$, the set of algebraic equations describing **Curry**.

2.3 Security Considerations

We emphasize that **Curry** and **Flurry** have a sound design strategy against linear [23] and differential [7] attacks. In particular, the Sboxes have been chosen to provide a good resistance against such attacks even for a small number of rounds [10, 11]. For a differential attack, one important parameter [16] is :

Definition 1. *Let $f : \mathbb{K}^m \mapsto \mathbb{K}$ be a mapping. We shall call δ-uniformity of f :*

$$\delta = \max_{(a,b)\in\mathbb{K}^*\times\mathbb{K}} \#\{x \in \mathbb{K} : f(x+a) + f(x) = b\}.$$

We shall then say that f is differentially δ-uniform.

This criterion permits to "measure" the resistance of a Sbox against differential cryptanalysis.

We recall that $\mathbb{K} \approx \mathbb{F}_{2^n} = \mathbb{F}_2(\theta)$. Let $(a, b) \in \mathbb{K} \times \mathbb{K}$, we denote by $\langle a, b\rangle = \sum_{i=0}^{n-1} a_i b_i$, with $a = \sum_{i=0}^{n-1} a_i\theta^i$, and $b = \sum_{i=0}^{n-1} b_i\theta^i$. For the resistance against linear cryptanalysis [23] :

Definition 2. *The non-linearity of a function $f : \mathbb{K}^m \mapsto \mathbb{K}$ is :*

$$\mathcal{NL}(f) = \min_{(a,b)\in\mathbb{K}\times\mathbb{K}^*} \#\{x \in \mathbb{K} : \langle x, a\rangle = \langle f(x), b\rangle\}.$$

The Sboxes selected in **Curry** and **Flurry** have the following properties [10, 11].

function	mapping	δ-uniformity	$NL(f)$
f_{-1}	$x \mapsto x^{-1}$	4	$2^{n-1}-2^{\frac{n}{2}}$
f_3	$x \mapsto x^3$	2	$\geq 2^{n-1}-2^{\frac{n}{2}+1}$
f_5	$x \mapsto x^5$	4	$\geq 2^{n-1}-2^{\frac{n}{2}+1}$
f_7	$x \mapsto x^7$	≥ 6	$\geq 2^{n-1}-3\cdot 2^{\frac{n}{2}}$

In [11], Buchmann, Pyshkin and Weinmann derive then upper bounds on the maximum differential and linear characteristic probability for **Curry** and **Flurry**. In turns out that such ciphers are immune against differential/linear attacks when the number of rounds is ≥ 4 (when the number of key bits is ≥ 128).

3 Gröbner Basics

In order to mount an algebraic key-recovery attack against the ciphers described previously, we have to address the problem of solving an algebraic system of equations. To date, Gröbner bases [8, 9] are the most efficient algorithmic solution for this problem. The description of F_4/F_5 [18, 19] is postponed in the Appendix.

3.1 Definition – Property

We start by defining two mathematical objects naturally associated with Gröbner bases [8, 9] : *ideals* and *varieties* [15]. We shall call *ideal generated* by $p_1, \ldots, p_s \in \mathbb{K}[x_1, \ldots, x_n]$ the set $\mathcal{I} = \langle p_1, \ldots, p_s\rangle = \left\{\sum_{k=1}^{s} p_k u_k : u_1, \ldots, u_k \in \mathbb{K}[x_1, \ldots, x_n]\right\}$. We will denote by $V_{\mathbb{K}}(\mathcal{I}) = \left\{\mathbf{z} \in \mathbb{K}^n : p_i(\mathbf{z}) = 0, \text{ for all } i, 1 \leq i \leq s\right\}$, the *variety associated* to $\mathcal{I}$, i.e. the common zeros over $\mathbb{K}$ of $p_1, \ldots, p_s$.

Gröbner bases offer an explicit method for describing varieties. Informally, a Gröbner basis of an ideal $\mathcal{I}$ is a generating set of $\mathcal{I}$ with "good" algorithmic properties. These bases are defined with respect to *monomial ordering*. For instance, the *lexicographical* (Lex) and *degree reverse lexicographical* (DRL) orderings which are widely used in practice are defined as follows :

Definition 1 *Let $\alpha = (\alpha_1, \ldots, \alpha_n)$ and $\beta = (\beta_1, \ldots, \beta_n) \in \mathbb{N}^n$. Then:*
– $x_1^{\alpha_1} \cdots x_n^{\alpha_n} \prec_{\text{Lex}} x_1^{\beta_1} \cdots x_n^{\beta_n}$ if the left-most nonzero entry of $\alpha - \beta$ is positive.
– $x_1^{\alpha_1} \cdots x_n^{\alpha_n} \prec_{\text{DRL}} x_1^{\beta_1} \cdots x_n^{\beta_n}$ if $\sum_{i=1}^n \alpha_i > \sum_{i=1}^n \beta_i$, or $\sum_{i=1}^n \alpha_i = \sum_{i=1}^n \beta_i$ and the right-most nonzero entry of $\alpha - \beta$ is negative.

Once a (total) monomial ordering is fixed, we define :

Definition 2 *We shall denote by $\mathrm{M}(n)$ the set of all monomials in n variables. We shall call total degree of a monomial $x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ the sum $\sum_{i=1}^n \alpha_i$. The leading monomial of $p \in \mathbb{K}[x_1, \ldots, x_n]$ is the largest monomial (w.r.t. some monomial ordering $\prec$) among the monomials of p. This leading monomial will be denoted by $\mathrm{LM}(p, \prec)$. The degree of p, denoted $deg(p)$, is the total degree of $\mathrm{LM}(p, \prec)$. Finally, the maximal total degree of p is the largest total degree among the monomials occurring in p.*

We are now in a position to define more precisely Gröbner bases.

Definition 3 *A set of polynomials $G \subset \mathbb{K}[x_1, \ldots, x_n]$ is a Gröbner basis – w.r.t. a monomial ordering $\prec$ – of an ideal $\mathcal{I}$ in $\mathbb{K}[x_1, \ldots, x_n]$ if, for all $p \in \mathcal{I}$, there exists $g \in G$ such that $LM(g, \prec)$ divides $LM(p, \prec)$.*

Gröbner bases computed for a lexicographical ordering (Lex-Gröbner bases) permit to efficiently compute varieties. A Lex-Gröbner basis of a *zero*-dimensional *system* (i.e. with a finite number of zeroes over the algebraic closure) is always as follows :

$$\{f_1(x_1) = 0, f_2(x_1, x_2) = 0, \ldots, f_{k_2}(x_1, x_2) = 0, f_{k_2+1}(x_1, x_2, x_3) = 0, \ldots\}$$

To compute the variety, we simply have to successively eliminate variables by computing zeroes of univariate polynomials and back-substituting the results. For a more thorough introduction to the topics of this part, we refer the reader to [1, 15].

3.2 The FGLM algorithm

Unfortunately, computing a Lex-Gröbner basis is much slower than computing a Gröbner basis w.r.t. any other monomial ordering. On the other hand, it is well known that computing degree reverse lexicographical Gröbner bases (DRL-Gröbner bases) is much faster. Algorithms changing the monomial ordering of a Gröbner basis permit to tackle this problem efficiently. In particular, the FLGM algorithm [17] permits – in the zero-dimensional case – to efficiently change the monomial ordering of a Gröbner basis. The complexity of this algorithm is polynomial in the number of solutions of the ideal.

Theorem 1. *Let $\mathcal{I}$ be a zero-dimensional ideal and $G_{\prec_{\text{old}}}$ be a $\prec_{\text{old}}$-Gröbner basis of $\mathcal{I}$ (w.r.t. to a monomial ordering $\prec_{\text{old}}$). FGLM [17] permits to compute a $\prec_{\text{old}}$-Gröbner basis $G_{\prec_{\text{new}}}$ of $\mathcal{I}$ knowing $G_{\prec_{\text{old}}}$ in $\mathcal{O}(n \cdot D^\omega)$, with $\omega, 2 < \omega \leq 3$ being the linear algebra constant, and D the dimension of the $\mathbb{K}$-vector space $\mathbb{K}[x_1, \ldots, x_n]/\mathcal{I}$ (which is equivalent to the number of zeroes counted with their multiplicities).*

We are not going to describe FGLM. However, we would like to mention that the performances of the version described in [17] can be improved using sparse linear algebra techniques. The most important operation of FGLM is to test linear dependencies of a set of D particular elements in the vector space $\mathbb{K}[x_1, \ldots, x_n]/\mathcal{I}$. Obviously, this can be done by performing a Gaussian elimination on a matrix M of size $D \times D$. If this matrix is sparse, we can use the Wiedemann's algorithm [25], whose complexity is $\mathcal{O}(D \cdot D')$, where D' is the number of non-trivial lines of the matrix M.

4 Improved Algebraic Attacks against Curry and Flurry

This section is divided into two parts. First, we show that the Buchmann, Pyshkin and Weinmann (BPW) attack against **Curry** and **Flurry** [10] can be improved using a "fast version" of FGLM. The goal is to illustrate the gain that we can obtain using sparse algebra techniques. This will permit to have a tight complexity estimates of the BPW attack, and then a good basis for comparing with the behavior of the new attack that we will present in the second part of this section.

4.1 Practical Improvements of the Buchmann, Pyshkin and Weinmann Attack

We start by recalling the (surprising) result of to Buchmann, Pyshkin and Weinmann [10]. They proved that for a well chosen order $\prec^*$ the polynomials of $\mathcal{P}_{\text{Flurry}}$ and $\mathcal{P}_{\text{Curry}}$ already form a Gröbner basis. It has to be noted that a similar result for AES-128 [12]. The key-recovery problem is then reduced to changing the order of a Gröbner basis. More precisely, solving $\mathcal{P}_{\text{Flurry}}$ (resp. $\mathcal{P}_{\text{Curry}}$) is equivalent to compute a Lex-Gröbner basis knowing a $\prec^*$–Gröbner basis. This can be done by using FGLM, and a precise complexity estimate can be given.

Lemma 1. *Let $\mathcal{I}_{\text{Flurry}}$ (resp. $\mathcal{I}_{\text{Curry}}$) be the ideal generated by the polynomials of $\mathcal{P}_{\text{Flurry}}$ (resp. $\mathcal{P}_{\text{Curry}}$). It holds that :*

$$\dim_{\mathbb{K}}\left(\mathcal{R}_{\text{Flurry}}/\mathcal{I}_{\text{Flurry}}\right) = \deg(f)^{m\cdot r}, \text{ for } \textbf{\textit{Flurry}}(n,m,r,f,D)$$
$$\dim_{\mathbb{K}}\left(\mathcal{R}_{\text{Curry}}/\mathcal{I}_{\text{Flurry}}\right) = \deg(f)^{m^2\cdot r}, \text{for } \textbf{\textit{Curry}}(n,m,r,f,D).$$

These results are not valid for the inverse function [10].

We recall that the complexity of FGLM is polynomial in the dimension of the quotient. As explained in 3.2, we can use sparse linear algebra techniques to decrease the exponent and obviously improving the efficiency of FGLM. To illustrate this fact, we will present experimental results. In the table below, we have quoted :
– The practical results obtained in [10] using FGLM. The authors have used the version available in Magma (version 2.11-8).
– the dimension $\dim_{\mathbb{K}}\left(\mathcal{R}/\mathcal{I}\right)$ of the quotient.
– The timings that we have obtained with our sparse version of FGLM.

		[10]	This paper
Flurry(n, m, r, f, D)	$\dim_{\mathbb{K}}(\mathcal{R}/\mathcal{I})$	F_4+FGLM (Magma)	"fast" FGLM (Fgb)
Flurry$(64, 1, 4, f_3, I_1)$	3^4	< 0.1 s.	< 0.1 s.
Flurry$(64, 1, 4, f_5, I_1)$	5^4	2.3 s.	< 0.1 s.
Flurry$(64, 1, 4, f_7, I_1)$	7^4	82.62 s.	19.4 s.
Flurry$(64, 1, 6, f_3, I_1)$	3^6	145.08 s.	2.1 s.

Interpretation of the Results. We observe that there is a non-negligible practical gain when using a sparse version of FGLM. Anyway, this approach becomes quickly impractical due to huge dimension of $\mathcal{R}_{\text{Flurry}}/\mathcal{I}_{\text{Flurry}}$ (resp. $\mathcal{R}_{\text{Curry}}/\mathcal{I}_{\text{Curry}}$). This is mainly due to the fact that the field equations are not included in $\mathcal{P}_{\text{Flurry}}$ (resp. $\mathcal{P}_{\text{Curry}}$). Therefore, the variety associated with these systems will mostly contain spurious solutions (solutions over the algebraic closure of $\mathbb{K}$). However, this is to our knowledge the best (algebraic) attacks proposed so far against **Flurry** and **Curry**. We will now present an alternative approach for attacking these ciphers.

Using Several Pairs of Plaintext/Ciphertext. The key recovery systems $\mathcal{P}_{\text{Flurry}}$ and $\mathcal{P}_{\text{Curry}}$ are constructed from only one pair plaintext/ciphertext (Section 2). In this part, we investigate the possibility of using few pairs of plaintext/ciphertext. Namely, we select $N > 1$ messages $m_1, \ldots, m_N$, request the corresponding ciphertexts $c_1, \ldots, c_N$ and try to solve a new key recovery system :

$$\mathcal{P}^N_{\text{Flurry}} = \bigcup_{i=1}^{N} \mathcal{P}_{\text{Flurry}}(m_i, c_i).$$

The set of equations $\mathcal{P}^N_{\text{Curry}}$ is defined similarly.
Note that for each pair (m_i, c_i), we have to introduce new variables corresponding to the internal states of the cipher. On the other hand, the variables corresponding to the key will remain the same for each pair (m_i, c_i). Again, we emphasize that the field equations are not included in the systems.

Remark that the result described previously (Sec. 4.1) for Flurry/Curry systems only holds when $N = 1$. As soon as $N > 1$, we have to follow a more classical approach to solve $\mathcal{P}^N_{\text{Flurry}}$ (resp. $\mathcal{P}^N_{\text{Curry}}$), i.e. the zero-dim strategy (Sec. 3.2).

Using N random messages. When we select *randomly* N messages $m_1, \ldots, m_N$, this approach will not lead to any improvement w.r.t. to the use of a single couple plaintext/ciphertext. Even worse, we have observed that the new systems are harder to solve in practice. To illustrate this, we have quoted few results that we have obtained for **Flurry**$(8, 2, 2, D_2, f_{\text{inv}})$ with different values of N. We also have included the time T necessary for solving the systems.

N	1	2	3
T	0.43 sec	25.8 sec	16 min 42 sec
Nb_{sol}	184	1	1

This is likely due to the fact that we have increased the number of equations/variables (corresponding to intermediate states), but the maximum degree reached during the Gröbner basis computation remains stable. Note anyway, that as soon as $N > 1$, the variety associated to $\mathcal{P}^N_{\text{Flurry}}$ (resp. $\mathcal{P}^N_{\text{Curry}}$) contains – most of the time – only one point corresponding to the secret key; thus only one direct Gröbner basis computation is needed. Hence the complexity of our attack is *linear* in the size of $\mathbb{K}$.

Using correlated messages. Algebraic-high order differential cryptanalysis. For taking advantage of multiple pairs, we have considered set of *correlated messages*. To explain the intuition behind our approach we have to introduce new definitions. The *derivative* (or *finite difference*) of a mapping $f : \mathbb{K}^n \mapsto \mathbb{K}^m$ at $r \in \mathbb{K}^n$ is defined by:

$$\Delta_r f(x) = f(x+r) - f(x).$$

Remark that if the components of f are of (total) degree d, then the components of $\Delta_r f(x)$ will be of total degree $\leq d$. To further decrease this degree, we can consider the ith derivative of f at the points $r_1, \ldots, r_i$ which is [21] as follows :

$$\Delta^{(i)}_{r_1,\ldots,r_i} f(x) = \Delta_{r_i}\left(\Delta^{(i-1)}_{r_1,\ldots,r_{i-1}} f(x)\right).$$

Now, let $L[r_1, \ldots, r_i]$ be the set of all binary linear combinations of the r_is. It is well known that :

$$\Delta^{(i)}_{r_1,\ldots,r_i} f(x) = \sum_{\delta \in L[r_1,\ldots,r_i]} f(x+\delta).$$

By the way, $\Delta^{(i)}_{r_1,\ldots,r_i} f(x)$ is equal [21] to zero if the r_i's are not linearly independent. We can now explain how we have generated the messages.

First, we will describe our approach on a small example, i.e. $N = 2$. We randomly select a message m_0, a difference r_1, and construct the key recovery system :

$$\mathcal{P}^N = \mathcal{P}(m_0, c) \cup \mathcal{P}(m_0 + r_1, c'), \text{ with } \mathcal{P}^N \in \left\{\mathcal{P}^N_{\text{Flurry}}, \mathcal{P}^N_{\text{Curry}}\right\}.$$

Now, let $T(\mathbf{X_i}, K_i) = T_{K_i}(\mathbf{X_i})$ be the round function[2] of **Flurry** or **Curry** (K_i is the subkey used at round i). For the first round, we have :

$$\mathbf{X_1}^{(0)} - T_{K_1}(m_0) = \mathbf{0} \in \mathcal{P}^2, \tag{1}$$

$$\mathbf{X_1}^{(1)} - T_{K_1}(m_0 + r_1) = \mathbf{0} \in \mathcal{P}^2. \tag{2}$$

The notation $\mathbf{X_1}^{(0)}$ (resp. $\mathbf{X_1}^{(1)}$) standing for the intermediate variables corresponding to m_0 (resp. $m_1 = m_0 + r_1$), a boldfaced letter will refer to a vector.
From (1) and (2), we deduce that the equations $\mathbf{X_1}^{(1)} - \mathbf{X_1}^{(0)} = \Delta_{r_1} T_{K_1}(m_0)$ are in the ideal generated by $\mathcal{P}^2$. Thus, by simply taking two pairs, we have created new equations relating the intermediates variables $\mathbf{X_1}^{(1)}, \mathbf{X_1}^{(0)}$, and the variables corresponding to K_1. The new equations are of degree strictly smaller than the initial equations of $\mathcal{P}^2$.

This can be viewed as an "algebraic-differential" style cryptanalysis. The idea of mixing differential and algebraic cryptanalysis has been proposed by Albrecht and Cid [2, 3]. They proposed to explicitly include new linear equations relating intermediates variables using the knowledge of a differential characteristic. This attack has been mounted against the cipher PRESENT [2, 3]. However, in our case, the equations derived from derivatives are automatically generated during the Gröbner basis computation.

We can iterate the process. Let $r_1, \ldots, r_N$ be a set of $N \geq 2$ linearly dependent vectors, and m_0 be a random message. We consider the system :

$$\mathcal{P}^N = \bigcup_{r \in L[r_1,\ldots,r_N]} \mathcal{P}(m_0 + r, c_r).$$

[2] For **Flurry** $X_i \in \mathbb{K}^t$, and for **Curry** $X_i \in \mathbb{K}^{m \times m}$.

We will denote by $\mathbf{X_i}^{(j)}$ the intermediates variables used at the ith round and corresponding to the jth message[3]. For the first round, we have that for all $k, 1 \leq k \leq \#L[r_1, \ldots, r_N]$:

$$\mathbf{X_1}^{(k)} - \mathbf{X_1}^{(0)} = \Delta_r T_{K_1}(m_0) \in \mathcal{P}^N, \text{ with } r \in L[r_1, \ldots, r_N].$$

As previously, we have created new equations of lower degree corresponding to derivatives. But, we will also create equations corresponding to the high order derivatives. For instance, let $r_1, r_2 \in L[r_1, \ldots, r_N]$. We have :

$$\mathbf{X_1}^{(0)} - T_{K_1}(m_0) = \mathbf{0} \in \mathcal{P}^N, \qquad \mathbf{X_1}^{(1)} - T_{K_1}(m_0 + r_1) = \mathbf{0} \in \mathcal{P}^N,$$
$$\mathbf{X_1}^{(2)} - T_{K_1}(m_0 + r_2) = \mathbf{0} \in \mathcal{P}^N, \qquad \mathbf{X_1}^{(3)} - T_{K_1}(m_0 + r_1 + r_2) = \mathbf{0} \in \mathcal{P}^N.$$

Therefore $\mathbf{X_1}^{(3)} - \sum_{k=0}^{2} \mathbf{X_1}^{(k)} = \Delta_{r_1,r_2} T_{K_1}(m_0) \in \mathcal{P}^N$. Moreover, r_1 and r_2 are not linearly independent, thus $\Delta_{r_1,r_2} T_{K_1}(m_0)$ is equal to zero. Thus, the ideal generated by $\mathcal{P}^N$ will now include linear relations between the intermediates variables $\mathbf{X_1}^{(j)}$. Then, such new linear equations will induce derivatives and high order derivatives in the subsequent rounds of the cipher. In our case, we know that $\mathbf{X_1}^{(3)} = \sum_{k=0}^{2} \mathbf{X_1}^{(k)}$. If we consider the second round, we have :

$$\mathbf{X_2}^{(0)} = T_{K_2}(\mathbf{X_1}^{(0)}) \in \mathcal{P}^N, \text{ and } \mathbf{X_2}^{(3)} = T_{K_2}(\mathbf{X_1}^{(0)} + \mathbf{X_1}^{(1)} + \mathbf{X_1}^{(2)}) \in \mathcal{P}^N.$$

The equations $\mathbf{X_2}^{(3)} - \mathbf{X_2}^{(0)} = \Delta_{\mathbf{X_1}^{(1)} + \mathbf{X_1}^{(2)}} T_{K_2}(\mathbf{X_1}^{(0)})$ is in the ideal generated by $\mathcal{P}^N$. This approach can be iterated for generating differentials of highest orders, and so new equations between the intermediates variables. Therefore, we have established here an interesting connection between algebraic attacks and high order differential cryptanalysis [21]; that we can call "algebraic-high order differential" cryptanalysis.

Experimental results. To summarize, our attack works as follows. Let $N > 1$ be an integer. We fix : $m_0 = (0, \ldots, 0)$, and $r_1 = (1, \ldots, 0)$. We constructe differences r_i, for all $i, 2 \leq i \leq N$, using the relation $r_{i+1} = \theta \cdot r_i$. After that, we have to solve the system :

$$\mathcal{P}^N = \bigcup_{r \in L[r_1, \ldots, r_N]} \mathcal{P}(m_0 + r, c_r).$$

As explained previously, the ideal generated $\mathcal{P}^N$ includes many new equations corresponding to all the derivatives of order less than $\lfloor \ln(N) \rfloor$. We expect that these new equations will allow the ease the Gröbner basis computation. We will see that this is indeed the case in practice. In next table, we have quoted the results we have obtained on **Flurry** and **Curry** with different values of N. Note that most of the parameters have been chosen for having a secret key of 128-bit. We have included :

T : total time of our attack.
$\mathrm{Nb_{op}}$: number of basic operations.
Mem : Maximum memory usage.
$D_{\max}$: the maximal reached during the Gröbner basis computation.
T_{BPW} : estimated complexity of the attack of [10].

[3] We supose w.l.o.g. an explicit order on the elements of $L[r_1, \ldots, r_N]$.

$\mathbf{Flurry}(n, m, r, f, D)$	T_{BPW}	N	$D_{\max}$	T	Nb_{op}	Mem
$\mathbf{Flurry}(16, 2, 4, f_3, D_2)$	$\approx 2^{38}$	3	3	0.0 s.	2^{10}	691 Mb.
$\mathbf{Flurry}(32, 2, 4, f_5, D_2)$	$\approx 2^{58}$	3	5	0.0 s.	$2^{15.5}$	691 Mb.
$\mathbf{Flurry}(32, 2, 4, f_7, D_2)$	$\approx 2^{70}$	3	7	0.04 s.	$2^{18.1}$	691 Mb.
$\mathbf{Flurry}(16, 2, 5, f_3, D_2)$	$\approx 2^{47}$	5	3	0.01 s.	2^{15}	1.2 Gb.
$\mathbf{Flurry}(16, 2, 5, f_5, D_2)$	$\approx 2^{70}$	9	5	0.15 s.	$2^{22.2}$	1.9 Gb.
$\mathbf{Flurry}(16, 2, 6, f_3, D_2)$	$\approx 2^{57}$	8	3	0.03 s.	2^{20}	2.1 Gb.
$\mathbf{Flurry}(16, 2, 6, f_5, D_2)$	$\approx 2^{84}$	14	5	543.7 s.	$2^{34.8}$	4.0 Gb.
$\mathbf{Flurry}(16, 2, 7, f_3, D_2)$	$\approx 2^{67}$	17	3	1.05 s.	$2^{25.8}$	4.9 Gb.
$\mathbf{Flurry}(32, 2, 7, f_3, D_2)$	$\approx 2^{67}$	17	3	1.9 s.	$2^{25.8}$	4.9 Gb.
$\mathbf{Flurry}(16, 2, 8, f_3, D_2)$	$\approx 2^{76}$	27	3	168.7 s.	$2^{33.5}$	8.7 Gb.
$\mathbf{Flurry}(16, 4, 4, f_3, D_2)$	$\approx 2^{76}$	3	3	0.01 s.	$2^{13.5}$	2 Gb.
$\mathbf{Flurry}(16, 4, 4, f_5, D_2)$	$\approx 2^{126}$	3	5	0.03 s.	$2^{18.1}$	2.2 Gb.
$\mathbf{Flurry}(16, 4, 4, f_7, D_2)$	$\approx 2^{177}$	3	7	0.33 s.	$2^{21.7}$	2 Gb.
$\mathbf{Flurry}(16, 4, 6, f_3, D_2)$	$\approx 2^{114}$	14	3	3.4 s.	$2^{27.4}$	1.3 Gb.
$\mathbf{Flurry}(16, 4, 8, f_3, D_2)$	$\approx 2^{152}$	**90**	3	1952 s.	$2^{36.1}$	117 Gb.
	$\approx 2^{152}$	100	3	2058 s.	$2^{36.2}$	130 Gb.
$\mathbf{Flurry}(16, 8, 6, f_3, D_4)$	$\approx 2^{228}$	20	3	35.8 s.	$2^{26.1}$	47 Gb.
$\mathbf{Curry}(32, 2, 3, f_3, D_2)$	$\approx 2^{57}$	2	10	0.01 s.	$2^{12.7}$	2.4 Gb.
	$\approx 2^{57}$	17	6	0.01 s.	$2^{14.5}$	18.8 Gb.
	$\approx 2^{57}$	**20**	3	0.01 s.	$2^{11.5}$	2.1 Gb.

Interpretation of the Results. We can observe that our approach is significantly faster than BPW attack [10] and that the complexity depends only linearly in, n, the size of the field $\mathbb{K}$. The most important is to observe that – in all cases – we have been able to find a number of pairs $N^* > 1$ such that the maximal degree reached during the Gröbner basis computation is equal to the degree d of the Sbox function. In practice, we have found this N^* by performing the following test incrementally on $N \geq 2$: we compute a DRL Gröbner basis of $\mathcal{P}^N_{\text{Flurry}}$ (resp. $\mathcal{P}^N_{\text{Curry}}$). If the maximal degree reached during this computation is greater than d then we stop the computation and set $N \leftarrow N + 1$, otherwise $N \leftarrow N^*$. On this basis, we can extrapolate the (experimental) complexity of our attack. Let b_{F} (resp. b_{C}) be the number of variables of the system $\mathcal{P}^{N^*}_{\text{Flurry}}$ (resp. $\mathcal{P}^{N^*}_{\text{Curry}}$). The complexity of our attack is :

$$\mathcal{O}\left(b_{\text{F}}^{\deg(f)\cdot\omega}\right), \text{for } \mathbf{Flurry}(n, m, r, f, D), \text{ and } \mathcal{O}\left(b_{\text{C}}^{\deg(f)\cdot\omega}\right), \text{ for } \mathbf{Curry}(n, m, r, f, D).$$

$\omega, 2 < \omega \leq 3$ being the linear algebra constant. We have then a polynomial time complexity for solving **Flurry** and **Curry** for power Sboxes.

These results are no longer valid for the inverse function. For this Sbox, we have observed a different behavior. Namely :

	N	$D_{\max}$	T	Nb_{op}	Mem
$\mathbf{Flurry}(n, m, r, f, D)$					
$\mathbf{Flurry}(16, 2, 6, f_{-1}, I_1)$	3	3	0.6 s.	2^{25}	1.8 Gb.
$\mathbf{Flurry}(16, 2, 7, f_{-1}, I_1)$	3	4	0.4 s.	2^{24}	1 Gb.
$\mathbf{Flurry}(16, 2, 8, f_{-1}, I_1)$	4	4	37.6 s.	2^{31}	1.4 Gb.
$\mathbf{Flurry}(16, 2, 9, f_{-1}, I_1)$	10	4	37296 s.	2^{41}	6.4 Gb.
$\mathbf{Flurry}(16, 4, 5, f_{-1}, I_1)$	2	4	0.5 s.	$2^{24.2}$	1.7 Gb.
$\mathbf{Flurry}(16, 4, 6, f_{-1}, I_1)$	4	4	810.3 s.	$2^{36.0}$	4.6 Gb.
$\mathbf{Flurry}(16, 8, 5, f_{-1}, I_1)$	3	4	3755.2 s.	$2^{37.5}$	5.4 Gb.

Once again, the use of correlated messages has permitted to go one step further in the algebraic cryptanalysis of **Flurry** with inverse SBox. However, it is not clear that there exists an optimal number of correlated messages N^* such that the maximal degree reached during the Gröbner basis computation is bounded by a constant. This deserves further investigations.

5 Conclusion

We propose a new algebraic cryptanalysis of **Curry** and **Flurry** using Gröbner bases techniques. We show that using correlated messages this technique can be effectively used to attack theses families of block ciphers. For choosing choose the correlated messages and to explain the behavior of this attack, we have established a connection between algebraic attacks and high order differential cryptanalysis. While our results have strong implications for the security of this kind of block ciphers when the power Sboxes is a power function (we have been able to break **Flurry**s – up to 9 rounds in less than 1 hour) it is more difficult to predict the complexity of the attack for the inverse Sbox.

References

1. W.W. Adams and P. Loustaunau. *An Introduction to Gröbner Bases.* Graduate Studies in Mathematics, Vol. 3, AMS, 1994.
2. M. Albrecht, C. Cid. *Algebraic Techniques in Differential Cryptanalysis.* Available at http://eprint.iacr.org/2007/137.
3. M. Albrecht, C. Cid. *Algebraic Techniques in Differential Cryptanalysis.* Proceedings of the First International Conference on Symbolic Computation and Cryptography, SCC 2008, Beijing, China, April 2008.
4. M. Bardet. *Étude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie.* Thèse de doctorat, Université de Paris VI, 2004.
5. B. Debraize. *Méthodes de cryptanalyse pour les schémas de chiffrement symétrique.* Thèse de doctorat, Université de Versailles, 2008.
6. M. Bardet, J-C. Faugère, B. Salvy and B-Y. Yang. *Asymptotic Behaviour of the Degree of Regularity of Semi-Regular Polynomial Systems.* In Proc. of MEGA 2005, Eighth International Symposium on Effective Methods in Algebraic Geometry, 2005.
7. E. Biham, A. Shamir. *Differential Cryptanalysis of DES-like Cryptosystems.* Advances in Cryptology - CRYPTO 1991, Lecture Notes in Computer Science, vol. 537, Springer–Verlag, pp. 2-21, 1991.
8. B. Buchberger, G.-E. Collins, and R. Loos. *Computer Algebra Symbolic and Algebraic Computation.* Springer-Verlag, second edition, 1982.
9. B. Buchberger. *Gröbner Bases : an Algorithmic Method in Polynomial Ideal Theory.* Recent trends in multidimensional systems theory. Reider ed. Bose, 1985.
10. J. Buchmann, A. Pyshkin, and R-P Weinmann *Block Ciphers Sensitive to Gröbner Basis Attacks.* Topics in Cryptology – CT RSA'06, Lecture Notes in Computer Science, vol. 3860, Springer–Verlag, pp. 313–331, 2006.
11. J. Buchmann, A. Pyshkin, and R-P Weinmann *Block Ciphers Sensitive to Gröbner Basis Attacks – Extended version.* http://eprint.iacr.org/2005/200.

12. J. Buchmann, A. Pyshkin, and R-P Weinmann *A Zero-Dimensional Gröbner Basis for AES*-128. Topics in Cryptology - CT RSA'06, Lecture Notes in Computer Science, vol. 4047, Springer–Verlag, pp. 78–88, 2006.
13. N. Courtois, and J. Pieprzyk. *Cryptanalysis of Block Ciphers with Overdefined Systems of Equations.* Advances in Cryptology – ASIACRYPT 2002, Lecture Notes in Computer Science, vol. 2501, pp. 267–287, 2002.
14. C. Cid, S. Murphy, M.J. B. Robshaw. *Small Scale Variants of the AES.* Fast Software Encryption (FSE 2005), Lecture Notes in Computer Science, vol. 3557, Springer–Verlag, pp. 267–287, 2005.
15. D. A. Cox, J.B. Little and D. O'Shea. *Ideals, Varieties, and algorithms: an Introduction to Computational Algebraic Geometry and Commutative algebra.* Undergraduate Texts in Mathematics. Springer-Verlag. New York, 1992.
16. J. Daemen, V. Rijmen. *The Design of Rijndael: The Wide Trail Strategy.* Springer-Verlag (2001).
17. J. C. Faugère, P. Gianni, D. Lazard, and T. Mora. *Efficient Computation of Zero-Dimensional Gröbner Bases by Change of Ordering.* Journal of Symbolic Computation, 16(4), pp. 329–344, 1993.
18. J.-C. Faugère. *A New Efficient Algorithm for Computing Gröbner Basis: F_4.* Journal of Pure and Applied Algebra, vol. 139, pp. 61–68, 1999.
19. J.-C. Faugère. *A New Efficient Algorithm for Computing Gröbner Basis without Reduction to Zero: F_5.* Proceedings of ISSAC, pp. 75–83. ACM press, July 2002.
20. J.-C. Faugère, and A. Joux. *Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems using Gröbner bases.* Advances in Cryptology - CRYPTO 2003, Lecture Notes in Computer Science, vol. 2729, Springer-Verlag, pp. 44–60, 2003.
21. X. Lai. *Higher Order Derivatives and Differential Cryptanalysis.* Communications and Cryptography, Kluwer Academic Publishers, pp. 227–233, 1994.
22. F. S. Macaulay. *The Algebraic Theory of Modular Systems.* Cambrige University Press, Cambrige, 1916.
23. M. Matsui. *Linear Cryptanalysis Method for DES Cipher.* Advances in Cryptology - Eurocrypt 1993, Lecture Notes in Computer Science, vol. 765, Springer–Verlag, pp. 386–397, 1993.
24. H. Raddum and I. Semaev. *New Technique for Solving Sparse Equation Systems.* Cryptology ePrint Archive: Report 2006/475. Available at http://eprint.iacr.org/2006/475.
25. D. H. Wiedemann. *Solving sparse linear equations over finite fields* IEEE Trans. Information Theory IT-32 (1986), 54–62.

The $\mathbf{F_4}$/$\mathbf{F_5}$ Algorithms.

The historical method for computing Gröbner bases is Buchberger's algorithm [8, 9]. Recently, more efficient algorithms have been proposed, namely the F_4 and F_5 algorithms [18, 19]. These algorithms are based on the intensive use of linear algebra techniques. Precisely, F_4 can be viewed as the "gentle" meeting of Buchberger's algorithm and Macaulay ideas [22]. In short, the arbitrary choices – limiting the practical efficiency of Buchberger's algorithm – are replaced in F_4 by computational strategies related to classical linear algebra problems (mainly the computation of a row echelon form).In [19], a new criterion (the so-called F_5 criterion) for detecting useless computations has been proposed. It is worth pointing out that Buchberger's algorithm spends 90% of its time to perform these useless computations. Under some regularity conditions, it has been proved that all useless computations can be detected and avoided.

A new algorithm, called F_5, has then been developed using this criterion and linear algebra methods. Briefly, F_5 (in its matrix form) constructs incrementally the following matrices in degree d :

$$A_d = \begin{array}{c} \\ t_1 f_1 \\ t_2 f_2 \\ t_3 f_3 \\ \dots \end{array} \begin{array}{c} m_1 \succ m_2 \succ m_3 \dots \\ \begin{bmatrix} \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \end{bmatrix} \end{array}$$

where the indices of the columns are monomials sorted w.r.t. $\prec$ and the rows are products of some polynomials f_i by some monomials t_j such that $\deg(t_j f_i) \leq d$. For a *regular system* [19] (resp. *semi-regular system* [4, 6]) the matrices A_d are of full rank. In a second step, row echelon forms of theses matrices are computed, i.e.

$$A'_d = \begin{array}{c} \\ t_1 f_1 \\ t_2 f_2 \\ t_3 f_3 \\ \dots \end{array} \begin{array}{c} m_1 \; m_2 \; m_3 \dots \\ \begin{bmatrix} 1 & 0 & 0 & \dots \\ 0 & 1 & 0 & \dots \\ 0 & 0 & 1 & \dots \\ 0 & 0 & 0 & \dots \end{bmatrix} \end{array}$$

For d sufficiently large, A'_d contains a Gröbner basis. An important parameter for evaluating the complexity of F_5 is the maximal degree d_{reg} occurring in the computation and the size $N_{d_{\text{reg}}}$ of the matrix $A_{d_{\text{reg}}}$. The overall complexity is dominated by the cost of computing the row echelon form of the last matrix $A_{d_{\text{reg}}}$. We have then a complexity of :

$$N_{d_{\text{reg}}}^{\omega} \approx \mathcal{O}(n^{\omega \cdot d_{\text{reg}}}),$$

with $\omega, 2 \leq \omega \leq 3$ being the linear algebra constant. For a semi-regular system of $\alpha \cdot n$ quadratic equations (over $\mathbb{F}_2$) in n variables, d_{reg} is asymptotically equivalent to :

$$d_{reg} \sim_{n \to \infty} \left(-\alpha + \frac{1}{2} + \frac{1}{2}\sqrt{2\alpha^2 - 10\alpha - 1 + 2(\alpha+2)\sqrt{\alpha(\alpha+2)}} \right) n.$$

More details on this complexity analysis, and further complexity results can be found in [4, 6]. Note that the algebraic systems arising in cryptography can behave very differently than a semi-regular system, leading then to a degree of regularity possibly much lower. A good example illustrating this fact is the cryptanalysis of HFE [20].