Hidden Field Equations and Gröbner basis cryptanalysis

Agathe Blanvillain[1], Kevin Tran [2]

[1]agathe.blanvillain.1@etudiant.univ-rennes1.fr
[2]kevin.tran@etudiant.univ-rennes1.fr

Contents

Introduction

In the Crypto '95 workshops, Patarin present a cryptanalysis of a new asymmetric algorithm [PAT] based on multivariate polynomials of degree two over a finite field described by T. Matsumoto and H. Imai. To repair the cryptosystem, Patarin propose a new cryptosystem also based on the same problem named Polynomial System Solving (PoSSo) which is NP-complete in $\mathbf{F}_2$ (proved by Fraenkel & Yesha 1979; Fraenkel & Yesha 1979). This new algorithm lie on the observation that Matsumoto & Imai algorithm choose weak key, HFE solves this problem with a larger set of key.

In 2003, Antoine Joux and Jean Charles Faugère implements an effective attack on HFE [FJG], and win the first HFE Challenge with a prize of 500$. This attack is based, on the improvement of the computation of Gröbner basis. The effectiveness of this computation is due to Faugère's works on family of algorithm named F4 & F5, and FGLM.

In this report, we will present the "Basic" HFE and our implementation, it will brief due to the fact that our main target was to study the cryptanalysis. We will introduce the division in $\mathbf{K}[x_1, \cdots, x_n]$, S-polynomial, Gröbner basis, Buchberger's algorithm and the explanation of Faugère's cryptanalysis. Some sections are remainder which can be skip, but it's necessary to understand Faugère's attack. Finally we will present you our implementations of two algorithms ([FGLM] and [F4]). To complete, our study we will introduce you differential attack, this chapter as the goal to prevent the choice of weak keys against differential attack.

All implementations will be done in Sagemath on Jupyter Notebook.

Chapter 1

"Basic HFE"

The Hidden Field Equations scheme was first presented by Patarin in [PAT]. In this section, we will first introduce the mathematical background of "basic" HFE cryptosystem. Then, we will explain the encryption/decryption process. Finally, we are going to introduce our implementation.

1.1 Mathematical background

To explain "basic" HFE, we first need to give mathematical properties.
Let $q = p^m$ be a prime power and n an integer.
Let $\mathbb{F}_q$ be a finite field and let $\mathbb{K}$ be a degree n extension of $\mathbb{F}_q$.
Finally, let f be a quadratic function of $\mathbb{F}_{q^n}[x]$ of degree d:

$$f(x) = \sum_{\substack{i \leqslant j \\ q^i + q^j < d}} \alpha_{i,j} x^{q^i + q^j} + \sum_{i, q^i < d} \beta_i x^{q^i} + \gamma$$

with $\alpha_{i,j}, \beta_i, \gamma \in \mathbb{K}$
Here we have $\mathbb{F}_{q^n} \simeq (\mathbb{F}_q)^n$, so the elements of $\mathbb{F}_{q^n}$ can be represented as n-tuples of elements of $\mathbb{F}_q$ and

$$\mathbb{F}_{q^n}[x]/(x^{q^n} - x) \simeq (\mathbb{F}_q[x_1, ..., x_n]/(x_1^q - x_1, ..., x_n^q - x_n))^n$$

Thus, we can represent $f(x)$ as a n-tuple of polynoms of n variables in $\mathbb{F}_q$:

$$f(x_1, ...x_n) = (p_1(x_1, ...x_n), ..., p_n(x_1, ...x_n))$$

The following theorem will be the heart of the HFE algorithm:

Theorem 1 *Let $\mathbb{K}$ be a finite field, with $|\mathbb{K}| = q^n$ with q and n "not too large" (for example $q \leqslant 54$ and $n \leqslant 1024$).*
Let $f(x)$ be a given polynomial in x in a finnite field $\mathbb{K}$, with a degree d "not too large" (for example $d \leqslant 1024$).
Let a be an element of $\mathbb{K}$.
Then it is always possible on a computer to find all the roots of the equation $f(x) = a$.

1.2 Encryption/Decryption

HFE encryption starts with a message M, represented as a n-tuple of $\mathbb{F}_q$(if $p = 2$, then each message can be represented by nm bits).

Then we include redudancy in the representation of each message M (by using correcting codes or hash function for example). That way, we can obtain the representation x of the message M.
Next, we define two affine bijections s and t of $(\mathbb{F}_q)^n \to (\mathbb{F}_q)^n$. As we did in the previous section, we can also represent s and t as n-tuples of linear polynoms in $(x_1, ..., x_n)$ on $\mathbb{F}_q$ linearly independants.
We will also use the function f as described in the previous section.
Thusly, we can introduce $(p_1, ..., p_n)$, n polynoms in n variables as:

$$(p_1, ..., p_n)(x) = t(f(s(x)))$$

The secret key is composed of :
⋆ the n-degree extension $\mathbb{K}$ of $\mathbb{F}_q$,
⋆ the function f,
⋆ the two affine bijections s ant t.

The public key is composed of :
⋆ a finite field $\mathbb{F}_q$ of $q = p^m$ elements,
⋆ an integer n,
⋆ n polynoms $(p_1, ..., p_n)$ in n variables in $\mathbb{F}_q$ (see construction above)
⋆ We can also explain the principle of redudancy (how to obtain x when we have M).

The encryption scheme is described in the figure bellow

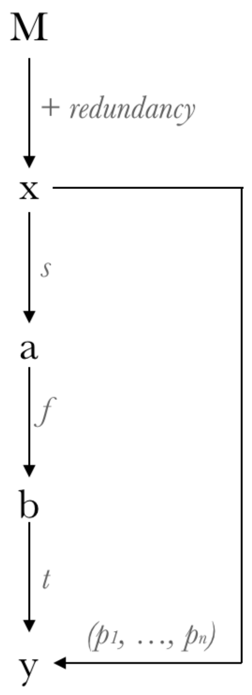


Figure 1.1: "basic" HFE for encryption

Now that we have encrypted our message, let's see the decryption scheme.

To decrypt the message y recieved, we are going to use the private key $(\mathbb{K}, f, s, t)$.
We know that $y = s(f(t(x)))$
s and t are invertible. We can calculate $s^{-1}(y) = f(t(x))$
The **Theorem 1** presented before permits us to find all the roots of f and to find a group S of solutions for $t(x)$ to the equation $s^{-1}(y) = f(t(x))$.
Then, we will find the group X of solutions for x by applying $t^{-1}(x)$ to each value of S.
Finally, we can find the group of possible M by inverting the redundancy and applying it to each value of X.

Let's explain how HFE is used for communication.
Alice wants to send a message M to Bob.
She takes Bob's public key. As we explained it above, she will apply redundancy to her message M to get the n-tuple $x = (x_1, ...x_n)$ and she computes the ciphertext $y = (p_1(x_1, ...x_n), ..., p_n(x_1, ...x_n))$.
Then she sends y to Bob.
When Bob recieves y, he will use his private key. As we also explained it above, he will first find all solutions z to the equation $f(z) = t^{-1}(y)$. Next, he computes all t^{-1}'s and finally he will use the redundancy to find M from these.

1.3 HFE implementation

Now that we have explained the principle of "basic" HFE, we will implement it in SageMath. The implementation is made in two major parts. First, we need to introduce basic elements for HFE and then in a second part, we can implement the cryptosystem of HFE.

1.3.1 Mathematical structure of this implementation

The first thing we have to do is to define all the basic parameters we will need : p, n, q
Then, we define our finite field $\mathbb{F}_q$ and its extension $\mathbb{F}_{q^n}$.
We will need a function that represents a function $f(x)$ of $\mathbb{F}_{q^n}[x]$ who satisfies the conditions of HFE as a n-tuple of polynomials in $\mathbb{F}_q[x_0, \cdots, x_{n-1}]$. For that, we begin by looking at f in the ring $\mathbb{F}_{q^n}[x_0, ..., x_{n-1}][X]$. Then, we evaluate f in $x_0 + x_1X + ... + x_{n-1}X^{n-1}$ and we reduce it by the irreducible polynomial of the extension (the modulus). This gives us a polynomials of degree $< n$ in X, so it gives the $p_i s \in \mathbb{F}_q[x_0, \cdots, x_{n-1}]$.

$$f \equiv \sum_{i=0}^{n-1} p_{i+1}(x_0, x_1, \cdots, x_{n-1})X^i \qquad [P(X), (x_i^q - x_i)_{i \in \{1, \cdots, n\}}]$$

To improve the algorithm, we could do a variant of the algorithm square and multiply. If you look at our implementation it can be seen as follow:

$$\begin{aligned}
f(Y) &= \sum_{i=0}^{d} \alpha_i Y^i \\
f(x_0 + x_1 X + \cdots x_{n-1} X^{n-1}) &= \sum_{i=0}^{d} \alpha_i (\sum_{j=0}^{n-1} x_j \cdot X^j)^i \\
&= \sum_{i=0}^{(n-1)\cdot d} \gamma_i(x_0, \cdots, x_{n-1}) X^i \\
&\equiv \sum_{i=0}^{n-1} \beta_i(x_0, \cdots, x_{n-1}) \overline{X}^i \qquad [P(X)] \\
&\equiv \sum_{i=0}^{n-1} p_i(x_0, \cdots, x_{n-1}) \overline{X}^i \qquad [P(X), (x_k^q - x_k)_k]
\end{aligned}$$

1.3.2 Possibles improvements

It's worked but as in the Integer Ring, we compute faster a^e mod n when we reduce at each exponentiation than when we compute a^e and then reduce.
We will not do this implementation, our main study was the Gröbner attack of HFE.

We also had a great problem to put the $(p_i)_i$ in $\mathbb{F}_q[x_0, \cdots, x_{n-1}]$. We had to solve many discret logarithm. To go through this problem, we have to change the all structure of vector space. In SageMath, we can't (or we didn't find the command) see $\mathbb{K}$ as an $\mathbb{F}_q$-vector space.

1.3.3 Our implementation

```
P = L_N._modulus()
x = ['x'+str(k) for k in range(N)]
M = PolynomialRing(L_N,N,x)
I = M.ideal([M(x[i])^q - M(x[i]) for i in range(N)]) # we add field's equations
Q = M.quotient(I)

R.<X> = PolynomialRing(M)
S = R.quotient(R(P))

variable = sum([M(x[i])*X^i for i in range(N)])

def HFE_representation_polynome(f):
    f_prime = R(f)
    f_prime = S(f_prime(variable)) # f_prime is look in K[x_1,...,x_n][X]/<P(X)>
    g = M(f_prime.lift()(a))

    f = 0
    for i in g.monomials():
        f+= (X^L_N(g.monomial_coefficient(i)).log(a)*i)
    p_list = [Q(pi).lift() for pi in list(f_prime)]
    # the p_is is now look in K[x_1,...x_n]/<(x_i^q-x_i)_i>
    return p_list
```

The implementation of the cryptosystem of HFE starts by the generation of the keys. First, we need to implement the HFE function $f(x)$ we presented in the first section. With the degree d (a sum of two powers of q) and the number of coefficients we want for $f(x)$, we generate the HFE function. The last parameter is not in the basic algorithm but as our implementation is slow, we have to study sparse polynomial.

```
def HFE_function(d, number_coeff):
    coeff = [L_N.random_element() for i in range(number_coeff)]
    f = 0
    lim = abs(floor(log(d,q)-1))
    for i in range(len(coeff)-2):
        theta = randint(0,lim)
        phi = randint(0,lim)
        f += coeff[i] * X^(q^theta + q^phi)

    mu_0 = L_N.random_element()
    f += mu_0 + coeff[-1] * X^d
    return f
```

The next step is to generate all the keys. We still need to have the d and number of coefficients we want. In this function, we first calculate f, the HFE function and we find s and t the two affine bijections. The last step in this function is to find the n polynomials in n variables $(p_1, ..., p_n)$ so we compute s(f(t)) and we use the function *HFE representation polynome* to convert this polynom of $\mathbb{F}_{q^n}[x]$ into n polynomials of $\mathbb{F}_q[x_0, \cdots, x_{n-1}]$.

```
def HFE_generates_keys(d, number_coeff):
    f = HFE_function(d, number_coeff)

    T.<Y> = PolynomialRing(K)

    s = R(T.random_element(1))
    t = R(T.random_element(1))

    p_list = HFE_representation_polynome(s(f(t)))
    private_key = (L_N,f,s,t)
    public_key = (K, p_list)
    return public_key, private_key
```

Now that we have our keys, we are now able to implement the HFE encryption function. As argumments, we need our message and the public key of generated above. To avoid implementing redundancy, we suppose m is an element of $(\mathbb{F}_q)^m$. The principle of the encryption is to evaluate all the m with the public key $s(f(t))$. We obtain the encrypted message y.

```
def HFE_encryption(m, public_key):
    K, p_list = public_key
    m = tuple(m)
    return [p(m) for p in p_list]
```

The last step of this implementation is to create the function decryption. As arguments, we need the encrypted message recieved and our private key generated previousely.

```
def HFE_decryption(y, private_key):
    L_N, f, s, t = private_key
```

```
a = L_N([0,1])
y = sum([y[i]*a**i for i in range(N)])

list_s = list(s)
s_inv = list_s[1]**(-1) * (X - list_s[0])
list_t = list(t)
t_inv = list_t[1]**(-1) * (X - list_t[0])

ft = s_inv(y)

v = tuple((L_N^N).zero())

roots = (f-ft).roots(multiplicities = False)
m = [L_N.vector_space(map = false)((t_inv(r))(v)) for r in roots]

return m
```

Chapter 2

Theory behind Faugère's Attack

HFE is based on multivariate polynomials, to study the cryptanalysis we have to study the structure of the ring $\mathbb{K}[x_1, \cdots, x_n]$. In this area, the reference is the book of Cox, Little, O'Shea [CLS]. In this section, we will introduce the definitions, propositions and theorems mostly directly from the Cox.

2.1 Reproduce the division and Gaussian's elimination in $\mathbf{K}[x_1, \cdots, x_n]$

To solve a multivariate polynomials system, we can be inspire by the linear case, were we know an fast algorithm named the Gaussian elimination.

Example 1

$$(S)\begin{cases} \lambda_{1,1}x_1 + \cdots + \lambda_{1,n}x_n & = & 0 \\ \vdots & & \\ \lambda_{r,1}x_1 + \cdots + \lambda_{r,n}x_n & = & 0 \end{cases}$$

We start to create a triangular system (elimination), then we find a solution from the last line and we expand this solution to the upper line (extension).

$$(S)\begin{cases} \lambda_{1,1}x_1 + & & \cdots + & \lambda_{1,n}x_n & = & 0 \\ & \lambda_{2,1}x_2 + & \cdots + & \lambda_{1,n}x_n & = & 0 \\ & \ddots & & & & \\ & & \lambda_{r,r}x_r + \cdots + & \lambda_{r,n}x_n & = & 0 \end{cases}$$

To reproduce that we will need a monomial ordering, to create an euclidean division.

Definition 1 An ideal I of a ring $(R, +, \cdot)$ as 3 properties:

1. $I \neq \emptyset$
2. $(I, +)$ is a subgroup of $(R, +)$
3. $\forall a \in R, \forall g \in I, a \cdot g \in I$

Definition 2 Let $\mathbb{K}$ a field, $f_1, \cdots, f_s \in \mathbf{K}[x_1, \cdots, x_n]$, then we set

$$V(f_1, \cdots, f_s) = [(a_1, \cdots, a_n) \in \mathbf{K}^n | f_i(a_1, \cdots, a_n) = 0 \text{ for all } 1 \leq i \leq s]$$

We call $V(f_1, \cdots, f_s)$ the affine variety of $f_1, \cdots, f_n$

To solve a multivariate polynomials system S we can see the solutions of S, the affine variety of S.

In order to do that, we need to learn more about the structure of the ideal $< S >$. We know that $\mathbb{K}[x_1, \cdots, x_n]$ is not an euclidean domain if $n > 1$, and in the most case an ideal are not principal, but we have:

Definition 3 Let a ring R, R is a notherian ring if it respects this two equivalent properties:

1. For all ideal I of R, I are generated by a finite number of element from R (i.e I can be write, $I =< e_1, \cdots, e_s >$ with $e_i \in R$)
2. Let
$$I_1 \subseteq I_2 \subseteq \cdots$$
be an ascending chain in R. Then there exists an $N \geq 1$ such that
$$I_N = I_{N+1} = \cdots$$

The second definition are named the Ascending Chain Condition (or ACC).

We know that if $\mathbf{K}$ is a field, $\mathbf{K}[x_1, \cdots, x_n]$ is a Notherian ring (theorem of Hilbert and Nother). So even if $I =< S >$ is an infinite set, $V(I)$ make sense, in fact if $I =< g_1, \cdots, g_s >$ then $V(I) = V(g_1, \cdots, g_s)$.

2.1.1 Monomial ordering

Definition 4 A monomial ordering $>$ on $\mathbf{K}[x_1, \cdots, x_n]$ is a relation on $\mathbf{Z^n_{\geq 0}}$ satisfying:

1. $>$ is a total ordering
2. if $\alpha > \beta$ and $\gamma \in \mathbf{Z^n_{\geq 0}}$ then $\alpha + \gamma > \beta + \gamma$
3. $\alpha \geq 0$ for all $\alpha \in \mathbf{Z^n_{\geq 0}}$

Definition 5 Let $f = \sum_\alpha a_\alpha x^\alpha$ be a nonzero polynomial in $\mathbf{K}[x_1, \cdots, x_n]$ and let $>$ be a monomial order. We define:

1. The **multidegree** of f is
$$\text{multideg}(f) = \max(\alpha \in \mathbf{Z}^n_{\geq 0} | a_\alpha \neq 0)$$
2. The **leading coefficient** of f is
$$\text{LC}(f) = a_{\text{multideg}(f)}$$
3. The **leading monomial** of f is
$$\text{LM}(f) = x^{\text{multideg}(f)}$$
4. The **leading term** of f is
$$\text{LT}(f) = \text{LC}(f) \cdot \text{LM}(f)$$

NB: In some books, we can see that the definition of leading term is switched with the definition of leading monomial

In an univariate polynomial ring $\mathbf{K}[x]$, we have an implicit ordering of monomial, for a $n \in \mathbb{N}$

$$x^n > x^{n-1} > \cdots > x > 1$$

.

In a linear polynomial see in the first example an implicite ordering:

$$x_1 > x_2 > \cdots > x_n$$

We give two important examples of well-ordering that we will use

Example 2 (Lexicographic Order) Let $\alpha = (\alpha_1, \cdots, \alpha_n)$ and $\beta = (\beta_1, \cdots, \beta_2)$.

$$x^\alpha >_{lex} x^\beta \Leftrightarrow \exists j \begin{cases} \alpha_i = \beta_i & \forall i < j \\ \alpha_j > \beta_j \end{cases}$$

Example 3 (Graded Reverse Lexicographic Order) Let $\alpha = (\alpha_1, \cdots, \alpha_n)$ and $\beta = (\beta_1, \cdots, \beta_2)$.

$$x^\alpha >_{grevlex} x^\beta \Leftrightarrow |\alpha| > |\beta| \text{ or } \exists j \begin{cases} \alpha_i = \beta_i & \forall i > j \\ \alpha_j < \beta_j \end{cases}$$

With this structure, we can create a sort of division (not euclidean).

Theorem 1 *Let $>$ be a monomial order and let $F = (f_1, \cdots, f_s)$ be an s-tuple of polynomials in $\mathbf{K}[x_1, \cdots, x_n]$ then every $f \in \mathbf{K}[x_1, \cdots, x_n]$ can be written as*

$$f = q_1 f_1 + \cdots + q_s f_s + r$$

where $q_i, r \in \mathbf{K}[x_1, \cdots, x_n]$, and either $r = 0$ or r is a linear combinaition, with coefficients in $\mathbf{K}$, of monomials, none of which is divisible by any of $LT(f_1), \cdots, LT(f_s)$.

The algorithm can be find in the [CLS] page 65. This algorithm have two main problem:

- The remainder depends of the order of the tuple F
- If the remainder of the algorithme for f is nonzero we can't conclude that $f \notin < F >$

Example 4 Let $f_1 = xy - 1, f_2 = y^2 - 1 \in k[x, y]$ with the lex order. Dividing $f = xy^2 - x$ by $F = (f_1, f_2)$, the result is

$$xy^2 - x = y(xy - 1) + 0(y^2 - 1) + (-x + y)$$

but for $F = (f_2, f_1)$ we have the result

$$xy^2 - x = x(y^2 - 1) + 0(xy - 1) + 0$$

2.2 Gröbner Basis

Definition 6 Let I an ideal of $\mathbf{K}[x_1, \cdots, x_n]$, $G = \{g_1, \cdots, g_s\}$ is a Gröbner basis of I, if

$$< LT(I) >=< LT(g_1), \cdots, LT(g_s) >$$

where $LT(I) = \{cx^\alpha |$ there exists $f \in I \setminus \{0\}$ with $LT(f) = cx^\alpha\}$

If we are given a finite generating set for I, say $I =< f_1, \cdots, f_s >$, then $< LT(f_1), \cdots, LT(f_s) >$ and $< LT(I) >$ may be different ideals.

Example 5 Let $I =< f1, f2 >$, where $f_1 = x^3 - 2xy$ and $f_2 = x^2y - 2y^2 + x$, and use graded reverse lexicographic ordering in $k[x, y]$. Then

$$x(x^2y - 2y^2 + x) - y(x^3 - 2xy) = x^2$$

so that $x^2 \in I$. Thus $x^2 = LT(x^2) \in< LT(I) >$. However $x^2 \notin< LT(f_1), LT(f_2) >$.

Theorem 2 *Fix a monomial order. Then every ideal $I \subseteq \mathbf{K}[x_1, \cdots, x_n]$ has a Gröbner basis.*

This theorem can be proved with the Dickson's theorem apply on $< LT(I) >$. The details can be found in [CLS]

2.2.1 Buchberger's algorithm

Definition 7 Let $\mathbf{K}[x_1, \cdots, x_n]$with a fix monomial order, $f, g \in \mathbf{K}[x_1, \cdots, x_n]$, we define the S-polynomial of f and g (or critical pair):

$$S(f, g) = \frac{x^\gamma}{LT(f)} \cdot f - \frac{x^\gamma}{LT(g)} \cdot g$$

where $x^\gamma = LCM(LM(f), LM(g))$

Proposition 1 (Buchberger criterion) *Let $\mathbf{K}[x_1, \cdots, x_n]$with a fix monomial order, I an ideal of $\mathbf{K}[x_1, \cdots, x_n]$, $G = \{g_1, \cdots, g_s\}$ is a Gröbner basis for I if and only if $S(g_i, g_j) \equiv 0[G]$*

This criterion permits us to have an easy way to compute a Gröbner basis.

```
def Buchberger(F):
    G = F.copy()
    P = list(Subsets(F,2))
    while P != []:
        f,g = P.pop()
        h = division(S(f,g),G)
        if (h != 0):
            P += [{h,f} for f in G]
            G.append(h)
    return G
```

This algorithm terminates due to the Ascending chain condition, the ideal of leading terms of G is growing after when P grows, after some step we know that $< LT(G) >$ will be equal to $< LT(I) >$ after that P will be strictly decrease to the empty list.

Definition 8 (Reduced Gröbner basis) Let I an ideal of $\mathbf{K}[x_1, \cdots, x_n]$, G is a reduced Gröbner basis of I if and only if $\forall g \in G$

1. The leading terms of g is equal to 1;
2. All terms of g is not reducible by $G \setminus \{g\}$

G is the unique reduced Gröbner basis of I for a fixed monomial ordering.

2.3 Solve polynomial system and HFE analysis

2.3.1 Shape position [Fhdr]

Definition 9 For an affine variety V of an ideal $I \subseteq \mathbf{K}[x_1, \cdots, x_n]$, if V is a finite set, then I is a 0-dimensional ideal of $\mathbf{K}[x_1, \cdots, x_n]$

Definition 10 A Gröbner basis $G = \{g_1, \cdots, g_m\}$, of a 0-dimensional ideal I of $\mathbf{K}[x_1, \cdots, x_n]$, is in shape position if:

- n = m, the number of equation is equal to the number of variables;
- $\mathbf{K}[x_1, \cdots, x_n]$is ordering with the lexicographic ordering
- we can write G as follow:
$$\begin{cases} g_n & = & P_n(x_n) \\ g_{n-1} & = & x_{n-1} - P_{n-1}(x_n) \\ \vdots & & \vdots \\ g_1 & = & x_1 - P_1(x_n) \end{cases}$$
Where $\forall 1 \leq i \leq n, P_i \in \mathbf{K}[x_n]$

2.3.2 Basic HFE Gröbner attack [FJG]

Suppose we have received a cypher text $c = (c_1, \cdots, c_n)$ encrypt with the Basic HFE cryptosystem, the public key $(\mathbf{F}_q, (p_1(x_1, \cdots, x_n), \cdots, p_n(x_1, \cdots, x_n)))$.

We have an explicit polynomial system define by:

$$(S) \begin{cases} p_1(x_1, \cdots, x_n) & = & c_1 \\ & \vdots & \\ p_n(x_1, \cdots, x_n) & = & c_n \end{cases}$$

The p_is are in $\mathbf{F}_q[x_1, \cdots, x_n]$, so we can add in the system S the field's equations (it will accelerate the compute of the Gröbner basis). So we redefine S:

$$(S) \begin{cases} p_1(x_1, \cdots, x_n) - c_1 & = & 0 \\ \vdots & & \\ p_n(x_1, \cdots, x_n) - c_n & = & 0 \\ x_1^q - x_1 & = & 0 \\ \vdots & & \\ x_n^q - x_n & = & 0 \end{cases}$$

To find the message $m = (m_1, \cdots, m_n)$ which verify $HFE(m, publickey) = c$, we have to solve S. We first have to define the ideal I $= < p_1(x_1, \cdots, x_n) - c_1, \cdots, p_n(x_1, \cdots, x_n) - c_n, x_1^q - x_1, \cdots, x_n^q - x_n > \subseteq \mathbf{F}_q[x_1, \cdots, x_n]$ and we compute with lex ordering a Gröbner basis of I. We often have G in shape position when the characteristic $\mathbf{K}$ is large, otherwise we can use the elimination's/extension's theory. The solutions of S will have the form:

$$Sol(S) = \{(P_1(\alpha), \cdots, P_{n-1}(\alpha), \alpha);\ \text{where } \alpha \text{ is a root of } P_n(x_n)\}$$

Experimentally, we found that the system S has the same solutions as the solutions of the function HFE decryption.

The cryptanalysis with Gröbner basis is due to the improvement in the algorithms to compute a Gröbner basis. The challenge proposed by Patarin was for HFE(D=96,n=80) where D is the degree of the secret polynomial f and n the number of variables. The polynomial system solving with this parameter was supposed to be untractable, we will see in the next chapter how works the algorithms proposed by Faugère.

Chapter 3

Strategy to compute a Gröbner basis

There are many way to compute a Gröbner basis with different complexity. First, we have the choice of algorithm, Buchberger, F4, F5, etc... Then, the choice of ordering. Many parameter but which is the most efficient?

3.1 Complexity and result

3.1.1 Notations

If we compute a Gröbner basis for a polynomial system S, we define:

- n, is the number of variables;
- d, the maximal degree of <S> that we will evaluate (cf. Macaulay matrix or F4);
- $\tilde{O}(\cdot)$, asymptotic complexity where we neglict polynomials in n.

3.1.2 Buchberger's algorithm

The problems with this algorithm are that we don't have a clear stopping condition, we know that the algorithm will end due to the ACC, but the choice of critical pair may dramatically change the computation time, in [GTime] there are some examples where the complexity of the computation is $\tilde{O}(d^{2^{\Omega(n)}})$

3.1.3 Computation of a Gröbner basis [FGLM]

For the computation of the Gröbner basis with the DRL order the complexity is $\tilde{O}(d^{O(n^2)})$ if the solutions are finite in number (Caniglia et al., 1988 and 1991), this complexity decreased to $\tilde{O}(d^{O(n)})$ the solutions at infinity is also finite in number (Lazard, 1983). The computation for the LEX order leads to a complexity of $\tilde{O}(d^{O(n^3)})$.

For our uses, we can suppose that computing a Gröbner basis costs $\tilde{O}(d^{O(n)})$ for DRL order.

To improve and regulate the complexity of the computation we have Faugère's algorithm [F4], which have theoretically the same asymptotic complexity as Buchberger algorithm, but in reality it's more efficient. (Table 1 from [FJG])

The stategy for compute a LEX Gröbner basis is:

1. Compute a Gröbner basis for DRL order $\sim \tilde{O}(d^{O(n)})$
2. Change order to LEX $\sim \tilde{O}(d^{O(n^2)})$

Table 1. Comparison of various algorithms and implementations (PC PIII 1 Ghz)

Algorithm	Buchberger				F_4	F_5
System	Maple	Magma	Macaulay	Singular	FGb	FGb
CPU time < **10m**	12	17	18	19	22	**35**
CPU time < **2h**	14	19	20	21	28	**45**

This stategy permits to compute LEX Gröbner basis with a complexity of $\tilde{O}(d^{O(n^2)})$ instead of $\tilde{O}(d^{O(n^3)})$

3.2 FGLM

The FGLM (for Faugère, Gianni, Lazard and Mora) algorithm [FGLM] is a algorithm of changing order, which takes in entry a reduced Gröbner basis for an old ordering (often grevlex) and transform to a reduced Gröbner basis with respect of the new ordering (often lex).

3.2.1 Multiplication matrix

We want to compute the multiplications matrix to bring the changing order algorithm to a problem of linear algebra where complexity and algorithm is well-known.

Definitions

First, we have to define the multiplication matrix

Definition 11 Let I an ideal of $\mathbf{K}[x_1, \cdots, x_n]$, $1 \leq i \leq n$, we define the application

$$\begin{array}{rcl} \phi_i : \mathbf{K}[x_1, \cdots, x_n]/I & \rightarrow & \mathbf{K}[x_1, \cdots, x_n]/I \\ p & \mapsto & p \cdot x_i \end{array}$$

If I is 0-dimensional, we can find a basis $\mathcal{B}$ of $\mathbf{K}[x_1, \cdots, x_n]/I$ and give a representation matrix M of the application ϕ_i:

$$M = \mathcal{M}at_{\mathcal{B} \leftarrow \mathcal{B}}(\phi_i)$$

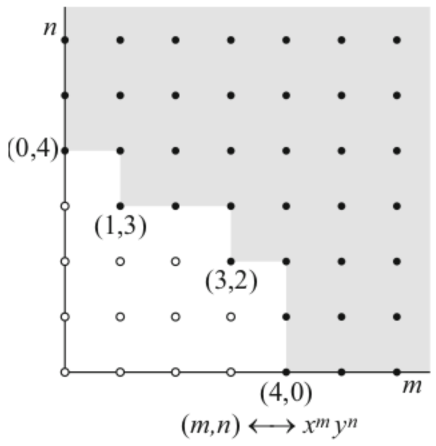
M is a multiplication matrix.

Definition 12 Let I an ideal of $\mathbf{K}[x_1, \cdots, x_n]$, I is zero-dimensional if I verify one of this equivalent properties:

1. $\mathbf{K}[x_1, \cdots, x_n]/I$ is a finite $\mathbf{K}$-vector space;
2. $V(I)$ is a finite set;
3. Let G a Gröbner basis of I for a fix monomial ordering, $\forall 1 \leq i \leq n, \exists k \in \mathbb{N}^*, x_i^k \in G$

FGLM works only for zero-dimensional ideal of $\mathbf{K}[x_1, \cdots, x_n]$. It will often be our case in cryptography.

We have a representation of terms which is not in I (i.e a representation of the complement of $< LT(I) >$):



Example 6 Let I an ideal of $\mathbb{Q}[x, y]$, and G the reduced Gröbner basis with $LT(G) =< x^4, x^3y^2, xy^3, y^4 >$, this ideal is zero-dimensional, the shadder region in $\mathbb{Z}^2$ of the figure represent $< LT(I) >$ The open circles are not in $< LT(I) >$.

This example allow us to present the notion of "staircase".

Definition 13 Let I a zero-dimensional ideal of $\mathbf{K}[x_1, \cdots, x_n]$and G a Gröbner basis for a monomial ordering.

$$\mathcal{E}(G) = \{t \in T|\ \text{t is not reducible by G}\}$$

We sort this set and define $deg(I) = dim(\mathbf{K}[x_1, \cdots, x_n]/I$.

$$\mathcal{E}(G) = \{e_1 = 1 < e_2 < \cdots < e_{deg(I)}\}$$

In the example, A vector space basis of $\mathbf{K}[x_1, \cdots, x_n]/I$ is $\{1, x, x^2, x^3, y, xy, x^2y, x^3y, y^2, xy^2, x^2y^2, y^3\}$ this basis is also equal to $\mathcal{E}(G)$ sort with the ordering lex with $x < y$

Definition 14 Let $\mathcal{E}(G)$ be the natural basis of $\mathbf{K}[x_1, \cdots, x_n]/I$, let

$$\mathcal{F}(G) = \{x_i e;\ \text{such that } e \in \mathcal{E}(G) \text{ and } x_i e \notin \mathcal{E}(G)\}$$

be the bordering of G.

In the figure example, $\mathcal{F}(G) = \{x^4, x^4y, x^4y^2, x^3y^2, x^3y^3, x^2y^3, xy^3, xy^4, y^4\}$, we can see that $\mathcal{F}(G) \setminus LT(G) = \{x^4y, x^4y^2, x^3y^3, x^2y^3, xy^4\}$.

Proposition 2 *Let I a zero-dimensional ideal, $(G, <)$ a reduced Gröbner basis, and $\mathcal{E}(G)$ the natural basis of $\mathbf{K}[x_1, \cdots, x_n]$, then for every element $t \in \mathcal{F}(G)$ exactly one of the following condition holds:*

1. *$\exists g \in G, t = LT(g)$;*
2. *$\exists t' \in \mathcal{F}(G), i \in \mathbb{N}^*, t = x_i t'$*

Proof Let $A = \{1 \leq j \leq n;\ \text{such that } x_j|t \text{ and } \dfrac{t}{x_j} \notin \mathcal{E}(G)\}$

We can study two cases:

1. First $A = \emptyset$, then $t \notin \mathcal{E}(G), \exists g \in G$ such that $LT(g)|t$, let $u = \dfrac{t}{LT(g)}$. Here if $u = 1$, we prove the first part of the proposition, otherwise $\exists 1 \leq j \leq n, x_j|u$, let $v = \dfrac{u}{x_j}$. We would have $\dfrac{t}{x_j} = vLT(g) \notin \mathcal{E}(G)$ and so $j \in A$ which is absurd. So $u = 1$ and $LT(g) = t$;

2. If $A \neq \emptyset$, $\exists j, x_j|t$ and $t' = \dfrac{t}{x_j} \notin \mathcal{E}(G)$. As $t \in \mathcal{F}(G) : x_j t' = t = x_i e$ with $e \in \mathcal{E}(G)$. The equality $i = j$ is impossible because this would imply that $t' = e \in \mathcal{E}(G)$. So $i \neq j$ and $x_j|e$; Plus we can prove that if $e \in \mathcal{E}(G) \Rightarrow \dfrac{e}{x_j} \in \mathcal{E}(G)$. So $t' = x_i \cdot (\dfrac{e}{x_j}) \in \mathcal{F}(G)$

Computation

We will first define the **parameters**:

1. G, is the reduced Gröbner basis;

2. EG, is the staircase of G: $\mathcal{E}(G)$, his structure is a dictionnary from python with the value:

$$EG = \{1 : 1, e_1 : e_1, \cdots, e_{deg(I)} : e_{deg(I)}\}$$

   All the keys and values are in $\mathbf{K}[x_1, \cdots, x_n]$

The function will **returns**:

1. M, a list of n element, $M^{(i)}$ will be the representation of the map ϕ_i;

2. N, a dictionnary, Normal Form for elements of $\mathcal{E}(G)$ or elements of $\mathcal{F}(G)$

We will explain our implementation step-by-step.

The Proposition 1 allows us to determine 3 cases for the computation of our mutiplication matrix: Let $k \in \{1, \cdots, n\}, j \in \{1, \cdots, deg(I)\}$

1. $x_k e_j \in E(G)$ so it exists $i \in \{1, \cdots, deg(I)\}$ such that $e_i = x_k e_j$ then we can put on $M^{(k)}[i, j] = 1$ and for $l \neq i, M^{(k)}[l, j] = 0$

```
for i, key in enumerate(EG):
    ei = EG[key]
    N[ei] = ei  # add on the dictionnary an element of EG
    for k in range(n):
        if (ei % x[k] == 0):
            ej = ei//x[k]
            if (ej in list_EG_values):
                j = list_EG_values.index(ej)
                M[k][i,j] = 1
```

2. $x_k e_j = LT(g)$ for a $g \in G$, will implies $LT(t - g) \in \mathcal{E}(G)$, $t - g$ can be write in the basis $\mathcal{E}(G)$

```
N[t] = t - g
dic_t = dict([(i,j) for j,i in list(N[t])]) # write N[t] in E(G) basis
for k in range(n):
    if (t % x[k] == 0):
        ej = t//x[k]
```

```
            if (ej in list_EG_values):
                j = list_EG_values.index(ej)
                for i in range(degI):
                    ei = list_EG_values[i]
                    if (ei in dic_t):
                        M[k][i,j] = dic_t[ei]
```

3. $t' \in \mathcal{F}(G)$ and $j \in \{1, \cdots, n\}$ such that $t = x_j t'$, this implies that if we already compute $N[t']$ we can compute $N[t]$, if we suppose that $N[t'] = \sum_{i=1}^{deg(I)} \mu_i e_i$, we will have $N[t] = N[x_j t'] = \sum_{i=1}^{deg(I)} \mu_i N[x_j e_i]$ we can rewrite that in $\mathcal{E}(G)$.

```
if (t % x[j] == 0):
    t_prime = t//x[j]
    if (t_prime in N):
        if(t_prime not in EG.keys()):   # implies that t' is in F(G)
            col = parent(M[0][:,0])([0]*degI)
            for m,r in list(N[t_prime]):
               ind = list_EG_values.index(r)
               col += m*M[j][:,ind]
            N[t] = 0
            for i,j in enumerate(EG):
               N[t] += col[i][0]*EG[j]
           for k in range(n):
               if (t%x[k] == 0 and t//x[k] in list_EG_values):
                    i = list_EG_values.index(t//x[k])
                    M[k][:,i] = col
```

This last case works because we sort $\mathcal{F}(G)$ before going in the loop, so if we choose $t \in \mathcal{F}(G)$ and $t = x_j t'$ like in 3., then $t' \in \mathcal{F}(G)$ will be in the dictionnary N.

Proposition 3 *We can compute Matrix multiplication with a complexity of* $O(ndeg(I)^3)$

3.2.2 FGLM algorithm

We want to contruct from the reduced Gröbner basis for a old monomial order $(G, <_{old})$ to the reduced Gröbner basis for a new monomial order $(G', <_{new})$. FGLM takes in entry M, N the multiplication matrix and the N we compute with the last algorithm. For FGLM, we first compute $\mathcal{E}(G')$ with the multiplication matrix, to do that we need a transition matrix P from $\mathcal{E}(G)$ to $\mathcal{E}(G')$. If we note $\mathcal{E}(G) = \{e_1 = 1 < e_2 < \cdots < e_{deg(I}\}$ and $\mathcal{E}(G') = S = \{\epsilon_1 < \cdots < \epsilon_{deg(I)}\}$, we have:

$$S = P \cdot \mathcal{E}(G)$$

We will construct incremently S and P; First we take $S_1 = \{\epsilon_1 = e_1\}$ $(S_{degI} = S)$.
We compute $\phi(x_k \cdot e_i) = M^{(k)} \cdot w$ with $w \in \mathcal{E}(G)$ we can write the product in $\mathcal{E}(G)$:

$$v = M^{(k)} w = \sum_{i=1}^{deg(I)} v_i e_i$$

To know if v is in S_s for some s. We compute

$$\lambda = P \cdot v = \sum_{i=1}^{deg(I)} \lambda_i \epsilon_i$$

There we have 2 cases:

1. if $\lambda_{s+1} = \cdots = \lambda_{deg(I)} = 0$ then $v \in Span(S_s)$
2. if $\exists k > l$ such that $\lambda_k \neq 0$ then $\epsilon_{s+1} = \lambda$. We have to update P such that:

$$P \cdot v = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \epsilon_{s+1}$$

The algorithm to update is given in the jupyter notebook.

We also need a function to convert a element of $\mathbf{K}[x_1, \cdots, x_n]/I$ in a $\mathbf{K}$-vector space in basis $\mathcal{E}(G)$, we call this function NF.

We will now present the FGLM algorithm, step-by-step:

```
S = [K(1)]
V = [NF(1)]
L = [(i,0) for i in range(1,n)] #(k,l) such that x_kS[l]
G = []
t = (0,0)
M_space = parent(M[0])
P = M_space.identity_matrix()
while(True):
    s = len(S)-1
    k,l = t
    v = M[k]*V[l]
    lamb = P*v
    if lamb[s+1:].is_zero(): # lambda is already in S_l
    # we're doing the opposite operation as for the multiplication matrix, when t = LT(g)
        g = x[k]*S[l]-sum([lamb[i]*S[i] for i in range(s+1)])
        if (g != 0):
            G += [g]
    else: # lambda not in S_l
        P = Update_mat_passage(s,lamb,P)
        S += [x[k]*S[l]]
        V += [v]
        L += [(i,s+1) for i in range(n)]
        L = list(set(L))
        L.sort()
        L = [(k,l) for k,l in L if all([x[k]*S[l]%g.lm() != 0 for g in G])]
    if L == []:
        return G
    t = L[0]
    L.remove(t)
```

Proposition 4 *This algorithm can be compute with a complexity of $O(ndeg(I)^2)$.*

The proof of the correctness is in [Fhdr]

It gives us a algorithm of changing order for a complexity of $\tilde{O}(d^{O(n^2)})$ due to Bézout's bound $(deg(I) \leq d^n)$ and the increased of the coefficients [FGLM]

3.3 F4

In this section, we will give the great lines of the algorithm and theorems on which it is based.

As we said earlier, the Buchberger's algorithm is really expensive, even with improvement for the selection of critical pairs, the algorithm pass 90 % of his times to compute critical pair to 0 [F4]. The idea of F4 is instead of picking and reducing critical pair one by one, we can reduce many at the same time. F4 will do as FGLM, transpose our polynomial problem to a linear algebra problem.

3.3.1 Macaulay's matrix [MB]

The idea is to look for a finite **K**-vector space.

Let $I =< f_1, \cdots, f_m >\subseteq \mathbf{K}[x_1, \cdots, x_n]$, we define $I_d = \{f \in I, deg(f) = d\}$ is a **K**-vector subspace of $S_d = \{f \in \mathbf{K}[x_1, \cdots, x_n]; deg(f) = d\}$; I_d is a finite vector space because S_d is.

Definition 15 Let a basis of S_d, $\mathcal{B}_d = \{\omega_1^{(d)} > \cdots > \omega_\mu^{(d)}\}$, for all $f_i \in I$, we consider all product with a monomial t such that $t \cdot f_i \in I_d$.

$$\mathcal{M}_{d,m}^{acaulay} = \begin{array}{cc} & \text{monomials from the basis } \mathcal{B}_d \\ tf_i & \begin{pmatrix} \\ \cdots \quad \text{coefficient of } f_i \text{ in } \mathcal{B}_d \quad \cdots \\ \\ \end{pmatrix} \end{array}$$

It's a matrix from a generator family $\{tf_i; \forall f_i \in I \text{ and } \forall t \in \text{ terms of degree } d - deg(f_i)\}$ to the basis B_d. With this matrix we can write all element of I_d.

The next proposition come from [Laz83] and [Laz01]

Proposition 5 *Let $\mathcal{M}_{d,m}^{acaulay}$, I_d we can compute the Gröbner basis of I_d by echelonize the matrix of Macaulay $\widetilde{\mathcal{M}_{d,m}^{acaulay}}$ for the ideal I of degree d and picking polynoms from the rows of $\widetilde{\mathcal{M}_{d,m}^{acaulay}}$ where the leading terms is absent from $\mathcal{M}_{d,m}^{acaulay}$*

Proposition 6 *Let I an ideal, there are an integer D_{max} such that $\widetilde{\mathcal{M}_{D_{max},m}^{acaulay}}$ can define a Gröbner basis of I.*

$D_{max} \leq \sum_{i=1}^{n}(d_i - 1) + 1$ this sum is called the Macaulay's bound.

This propositions gives us a way to compute a Gröbner basis with linear algebra methods. This computation can be improved because the rank of the Macaulay's matrix for the degree D_{max} is much lower than the rows of the matrix.

This is the main idea of F4, you can read the algorithm of Faugère and his correctness proof in [F4]. We also have implement this algorithm on Sagemath. In F5, Faugère constructs a matrix of full rank to improve F4.

3.3.2 Distinguish a HFE system

In the thesis of Magali Bardet, we can find a algorithm to distinguish a polynomial system generates by HFE and an random polynomial system. This distinguisher comes from empirical results:

d is the degree in the parameter of F4 present in the *Sel* function, D is the secret polynom f in HFE and n is the number of variables in the polynomial ring (or the length of the ciphertext).

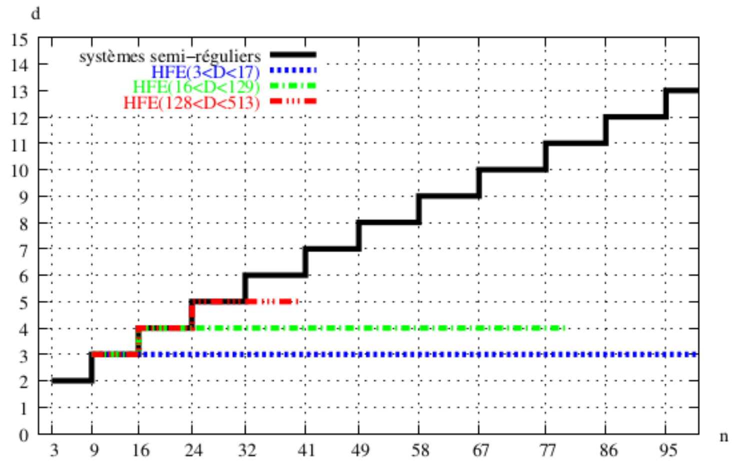


FIG. 5.3 – d-Régularité des systèmes HFE

HFE(D)	$3 \leq D \leq 16$	$17 \leq D \leq 128$	$129 \leq D \leq 512$	$513 \leq D \leq 1280$
d-régularité	3	4	5	6

TAB. 5.1 – d-régularité des systèmes HFE

We can see with this graphic and this table, for a secret polynomial of degree D we can decrypt the message, only by launch F4 or F5 restrict with the degree d. We deduce from this observation an algorithm to distinguish HFE:

```
Entrée : { $F = (f_1, \ldots, f_n)$ un système d'équations quadratiques,
         { $D$ un entier
Sortie : { oui  si $F$ provient d'un système HFE($D$)
         { non  sinon
Calculer $d$ la $d$-régularité d'un système HFE($D$) (en utilisant la table 5.1),
Exécuter l'algorithme F5-matriciel jusqu'au degré $d$,
Si en degré $d$ on observe de nombreuses chutes de degré,
  alors Retourner oui
  sinon Retourner non.
```

FIG. 5.4 – Un distingueur pour les systèmes HFE

Chapter 4

Differential cryptanalysis

HFE cryptosystem can also be attacked with differential cryptanalysis. The idea behind this is to see if we can build an HFE system which resists 2 forms of differential attacks. Fot this, we invertigate HFE differential properties. The reference for this section will be the article of T. Daniels and D. Smith-Tone [DST].
First, what is a differential attack?

Definition 1 [BS] Differential cryptanalysis is a method which analyzes the effect of particular differences in plaintext pairs on the differences of the resultant ciphertext pairs. These differences can be used to assign probabilities to the possible keys and to locate the most probable key. This method usually works on many pairs of plaintexts with the same particular difference using the resultant ciphertext pairs. The goal is to reduce the amount of time needed to find the key. It's what is called a chosen plaintext attack; the attacker has access to plaintext and corresponding ciphertext.

We are interested in this because several other algebra based systems like oil and vinegar and SFLASH have been broken by certain differential attacks, where an opponent will exploit non trivial differential structure in a core map.
So we want to find of provable "Differential Security Metric". There is actually precedent for this: It was shown by Perlner and Smith-Tone [PST] that systems C* and pC* have no non trivial differential invariants. So an opponant who wants to exploit such a structure has nothing to go on.

In this section, we are going to see the restrictions we can put on an HFE system to make it secure against some differential attacks.
First, we will look at the basic definions we will need.
Then we will explain how to protect HFE from differential symmetries.
Finally, we will work on how we can protect HFE cryptosystem against differential invariants.

4.1 Basic definitions

Definition 2 The Discrete Differential of a fiels map f is defined as $DF(a, x) = f(a + x) - f(a) - f(x) + f(0)$ (a, x are vectors in an arbitrary field). It is a difference operator that is normalized.

The "Differential Adversary" is an opponent who wants to gain information about an unknown field map by discovering properties of the map's Discrete Differential. He will try to calculate the two following things about our core map:

$\star$ a linear map M such that $DP(My,x) + DP(y,Mx) = \Lambda_M \circ DP(y,x)$
$\star$ subspaces V, W (of the relevant field) with $dim(V) + dim(W) \geqslant n$, where n is the number of variables, such that $\forall x \in V, y \in W, DP(y,x) = 0$
We can recall the formal definition of the core map of HFE given before as:

If we take the differential of the core map $f(x)$ given in the first part , we see that all the affine terms get removed. We finally get:

$$Df(y,x) = \sum_{\substack{i \leqslant j \\ q^i+q^j<D}} \alpha_{i,j}(y^{q^i}x^{q^j} + y^{q^j}x^{q^i})$$

The differential satisfies the properties of bilinearity.

4.2 Protection against differential attacks

4.2.1 Differential Symmetries

We would like to prevent the existence of a linear map $M : \mathbb{K} \to \mathbb{K}$ such that

$$DP(My,x) + DP(y,Mx) = \Lambda_M \circ DP(y,x)$$

$$Mx = m_0 x + ... + m_{n-1}x^{q^{n-1}}$$

There will be maps that satisfy this equation but we want to avoid the ones that are in our context non trivial.

We write M in a generalized polynomial form, we pick a vector representation x for $\mathbb{K}$ to write M in the following representation. We can say

$$x = \begin{bmatrix} x \\ x^q \\ \vdots \\ x^{q^{n-1}} \end{bmatrix} \qquad M = \begin{bmatrix} m_0 & m_1 & \cdots & m_{n-1} \\ m_{n-1}^q & m_0^q & \cdots & m_{n-2}^q \\ \vdots & \vdots & \ddots & \vdots \\ m_1^{q^{n-1}} & m_2^{q^{n-1}} & \cdots & m_0^{q^{n-1}} \end{bmatrix}$$

We take the left hand side of the equation above, to rewrite it in a matrix form:

$$DP(My,x) + DP(y,Mx) = y(M^\top DP + DPM)x$$

Here, DP is the matrix representing the action of the discrete differential on the input variables. Now, representing

$$DP(My,x) + DP(y,Mx) = \Lambda_M \circ DP(y,x)$$

in a matrix form gives:

$$y(M^\top DP + DPM)x = y(\Lambda_M \circ DP)x$$

so we set

$$M^\top DP + DPM = \Lambda_M \circ DP$$

and look for solutions.

We now try to get a visual argument.

We try to investigate a sample core term from our core map:

$$f(x) = \alpha_{i,j}x^{qi+q^j} + \alpha_{r,s}x^{q^r+q^s}$$

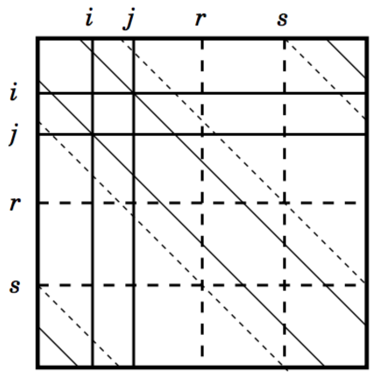


Figure 4.1: graphical representation of the equation $M^\top DP + DPM = \Lambda_M \circ DP$ for the HFE polynomial $f(x)$

On the diagram, horizontal and vertical lines represent nonzero entries in $M^\top DP + DPM$ while diagonal lines represent nonzero entries in $\Lambda_M \circ DP$. Solid lines correspond to the (i, j) monomial while dotted lines correspond to the (r, s) monomial. The idea is that if we can make all of these lines miss each other almost everywere, then we can severely reduce the number of non zero coefficients for that linear map M and maybe make the except one zero, which makes a multiplication map. It's actually quite easy to do that. When you build your HFE system, all you need to do is pick exponents for your coremap such that none of them are repeated and the difference between all of them is unique. Then, f will have no non trivial linear map satisfying the equation told earlier.

Theorem 2 *Let f(x) be an HFE polynomial (in particular f is not a monomial function. Suppose that f has the following properties.*
$\star$ *no power of q is repeated among the exponents of f*
$\star$ *the difference of powers of q in each exponent is unique*
Then f has no non-trivial differential symmetry

Proof $\star$ Given the assumptions and the idea of the picture, nearly all potential non-zero terms, that is the m_i "miss each other"
$\star$ $\forall i \neq 0, m_i = 0$
$\star$ We have $Mx = m_0 x$ for $m_0 \in \mathbb{F}_q$ (Smith-Tone 2010) So m is just a multiplication by a scalar (a trivial linear map)
$\star$ f has no non-trivial symmetry.

(See complete proof in Annexe 1)

4.2.2 Differential Invariants

Differential invariants is the other type of structure we'd like to avoid. There are even easier to avoid than differential symmetries.

Definition 3 [AP] A differential invariant define an induction principle for differential equations and which can be checked for invariance along a differential equation just by using their differential structure, without having to solve them.

Definition 4 Let $f : \mathbb{K} \to \mathbb{K}$ be a function. A differential invariant of f is a subspace $V \subseteq \mathbb{K}$ with the property that there is a subspace $W \subseteq \mathbb{K}$ such that $dim(W) \leqslant dim(V)$ and $\forall A \in Span_{\mathbb{F}_q}(Df_i), AV \subseteq W$.

f is a subspace of $\mathbb{K}$ such that under the action of all the differential coordinate forms and al linear combination of those, this subspace V stays inside W.

Definition 5 We can define a corresponding subspace $V^{\perp}$ as the set of all elements $x \in \mathbb{K}$ such that the dot product $\langle x, Av \rangle = 0 \;\; \forall v \in V, \forall A \in Span_{\mathbb{F}_q}(Df_i)$.

These two subspaces will be what we focus on to try to avoid our differential invariants.

Proposition 1 *Suppose that V is a subspace of $\mathbb{K}$ and $V^{\perp}$ is as above. If $M : \mathbb{K} \to V$ and $M^{\perp} : \mathbb{K} \to V^{\perp}$, then:*

$$\forall y, x \in \mathbb{K}, Df(M^{\perp}y, Mx) = 0$$
$$\Leftrightarrow Df(M^{\perp}\mathbb{K}, M\mathbb{K}) = 0$$

Now, we are going to take our space V, make assumptions about it, and then attempt to caracterize the space $V^{\perp}$.
We are employing a result from linear algebra:

Proposition 2 *If A, B are two $m \times n$ matrices, then $rank(A) = rank(B)$ if and only if there exist nonsingular matrices C, D, such that $A = CBD$.*

Given the definitions and the previous two propositions, we can write $M^{\perp} = SMT$, where S may be singular and T non-singular.

Now by making assumptions about V and M we are going to see how we can caracterize S and even make it a zero map on V.
The ideas are:
Given V and $V^{\perp}$, we can show that $dim(V) + dim(V^{\perp}) \geqslant n$, implying that either $dim(V^{\perp}) \geqslant \dfrac{n}{2}$ or $dim(V) \geqslant \dfrac{n}{2}$
Then, we assume V is a non-trivial differential invariant (i.e $V \neq 0$ and $V \neq \mathbb{K}$) Without loss of generality, we assume $dim(V) \geqslant \dfrac{n}{2}$
We now take M the map to V. We will attempt to characterize $M^{\perp} = SMT$ by characterizing S.
If S is the zero map on V, then $V^{\perp} = 0$, contradicting the non-triviality of (V, W).

When you build your coremap, not all the exponents have to be unique, but if you have one that is unique, it can be shown that there are no non-trivial differential invariant to our core map.

Theorem 3 *Let f be an HFE polynomial with degree bound $D < q^{\frac{n}{2}}$. If there is a power of q which is unique, f has no non-trivial invariant structure.*

The uniqueness of an exponent will force that S must map all V elements to 0 if they have to satisfy the equation

$$Df(SMTy, Mx) = 0$$

4.3 Question raised and conclusion

With the restriction we put on f, a question can be raised:
If we put restriction on the exponents of our HFE maps will we diminish the key space too much?

We don't disminish it a lot. In fact it is easy to check, given a specific f, that the core map f lacks non-trivial differential structure. We have restricted the key space quite a bit but we can still have a rich key space. In addition, these conditions are sufficient but not necessary conditions so it is possible to have systems without these structures even if they don't satisfy all the theorems exposed before.

With a simple choice of parameters we can probably eliminate non-trivial differential symmetric and invariant structure while maintaining security against attacks exploiting a diminished private key space.
As an extension to this part, we can add that it is possible to extend HFE differential properties to HFE- with very few changes.

Annexes

1 - Proof Theorem 4.2.2 [DST]

First consider computing DfM. From the condition on the monomials of f, Df has at most a single nonzero entry in any row or column. Therefore each row of DfM is a multiple of a row in M. In particular, if $\alpha_{i,j}x^{q^i+q^j}$ is a monomial of f, then the ith row of DfM is :

$$[\alpha_{i,j}m_{-j}^{q^j}\alpha_{i,j}m_{1-j}^{q^j}...\alpha_{i,j}m_{-1-j}^{q^j}]$$

and the jth row is:

$$[\alpha_{i,j}m_{-i}^{q^i}\alpha_{i,j}m_{1-i}^{q^i}...\alpha_{i,j}m_{-1-i}^{q^i}]$$

Consider the ith row of $M^TDf + DfM$. For all k not occuring as a power of q in f, the (i,k)th entry is $\alpha_{i,j}m_{k-j}^{q^j}$. Consider the (i,k)th entry of $M^TDf + DfM$, this quantity is the sum of the (i,j)th entry of DfM and the (j,i)th entry, specifically $\alpha_{i,j}(m_0^{q^i} + m_0^{q^j})$. Let $\alpha_{r,s}x^{q^r+q^s}$ be another monomial of f. Then the (i,r)th entry of $M^TDf + DfM$ is $\alpha_{i,j}m_{r-j}^{q^j} + \alpha_{r,s}m_{i-s}^{q^s}$ and the (i,s)th entry is $\alpha_{i,j}m_{s-j}^{q^j} + \alpha_{r,s}m_{i-r}^{q^r}$.

In $\Lambda_M Df$, for all $\alpha_{i,j}x^{q^i+q^j}$ a monomial in f, the $(i+k, j+k)$th entry is equal to the $(j+k, i+k)$th entry and takes the value $\alpha_{i,j}^{q^k}\lambda_k$ while all other entries are zero.

Therefore, consider the elements in the ith row of the equation $M^TDf + DfM = \Lambda_M Df$. For every monomial $\alpha_{r,s}x^{q^r+q^s}$ in f, we have that the $s-r+i$th element and the $r-s+i$th element of row i in $\Lambda_M Df$ are nonzerp. All other entries of that row are zero. Therefore, for all k not occuring as a power of q in f or as a difference of the powers of q in an exponent of a monomial in f plus i, $m_{k-j} = 0$. Given the condition that the differences of powers of q in the exponents are unique, and the equations $m_{k-t} = 0$ for all other t occuring as powers of q, we obtain $m_i = 0$ for all $i \neq 0$. Therefore M is a multiplication map. But if $m_0 \notin \mathbb{F}_q$ this implies that the polynomial is a C^* monomial, a contradiction. Thus, M is simply multiplying a scalar which induces a symmetry for every map $g : \mathbb{K} \to \mathbb{K}$. Thus f has no nontrivial differential symmetry.

Bibliography

[CLS] D.Cox; Little; O'Shea (2010), "Ideals, Varieties, and Algorithms", Springer 4^{th} edition

[PAT] J. Patarin (1996), "Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms", Eurocrypt '96

[GTime] Mayr, Ernst W; Meyer, Albert R (1982). "The complexity of the word problems for commutative semigroups and polynomial ideals".

[F4] J.C Faugère (June 1999), "A new efficient algorithm for computing Gröbner bases (F4)", Journal of Pure and Applied Algebra

[FGLM] J.C. Faugère; P. Gianni; D. Lazard; T. Mora (1993),"Efficient Computation of Zero-dimensional Gröbner Bases by Change of Ordering", Journal of Symbolic Computation

[FJG] A. Joux; J.C Faugère , "Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems Using Grobner Bases"

[MB] M. Turrel Bardet (2001-2004), "Etude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie."

[DST] T. Daniels; D. Smith-Tone, "Differential Properties of the HFE Cryptosystem"

[Fhdr] J.C. Faugère (Feb 9 2007), "Calcul efficace des bases de Gröbner et Applications"

[BS] E. Biham; A. Samir (Sep 7 2009), "Differential Cryptanalysis of the Data Encryption Standard" (Introduction to differential cryptanalysis)

[PST] R. Perlner; D. Smith-Tone, "A Classification of Differential Invariants for Multivariate Post-Quantum Cryptosystems"

[Laz83] D. Lazard. Gröbner bases, Gaussian elimination and resolution of sys- tems of algebraic equations. In Computer algebra (London, 1983), volume 162 of Lecture Notes in Comput. Sci., pages 146–156. Springer, Berlin, 1983.

[Laz01] D. Lazard. Solving systems of algebraic equations. ACM SIGSAM Bulletin, 35(3) :11–37, Septembre 2001.

[AP] A.Platzer, " The structure of differential invariants and differential cut elimination"