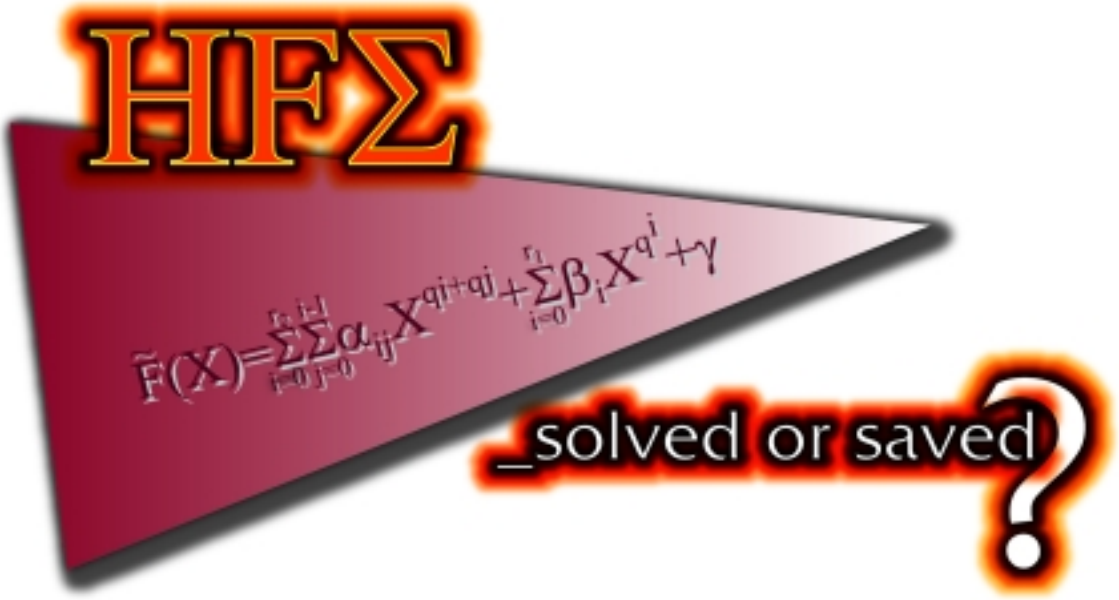
Bachelor Thesis



– by **Fabian Werner** –

TU Darmstadt

University:	Darmstadt University of Technology
Department:	Computer Science
Subject group:	Theoretical computer science - Cryptography and Computeralgebra
Supervisor:	Prof. Dr. Jintai Ding

Contents

0 Notations **5**

1 Introduction **7**
1.1 Multivariate cryptosystems 7

2 HFE **8**
2.1 Description of HFE 8
2.1.1 $\tilde{F}(X)$ produces at most quadratic polynomials 9
2.1.2 Encryption 10
2.1.3 Decryption 10
2.2 Example of an HFE system 11
2.3 MQ-GF(2) is $\mathcal{NP}$-hard 12

3 Gröbner Basis Theory **14**
3.1 Why are Gröbner Bases interesting? 14
3.1.1 Elimination of variables 15
3.1.2 Successive completion 16
3.2 Example for breaking HFE 16
3.3 Existence of Gröbner Bases 17
3.3.1 Dicksons Lemma 17
3.3.2 Hilberts Basis Theorem 18
3.4 Estimating Gröbner Bases 19
3.4.1 The Buchberger criterion 19
3.4.2 Buchbergers algorithm 21
3.5 The Faugère F_4 algorithm 23

4 How to save HFE **27**

5 Saving HFE by adapting the characteristic of k **27**
5.1 The role of the Field Equations 27
5.1.1 Inserting the field equations 27
5.1.2 Leaving out the field equations 28
5.2 Computer experiments 29
5.2.1 Experiment on the number of solutions 29
5.2.2 Experiment on time and memory usage 29

6 Internal perturbation of HFE (IPHFE) **32**
6.1 Finding a k-linear map of dimension IP_r 33
6.2 Encryption and decryption 33
6.3 Toy example for IPHFE 34
6.4 Computer experiments on IPHFE 35
6.5 The algebraic attack revisited 35
6.5.1 Results for computing Gröbner Bases *with* the HFE parameter . . 35

6.5.2 Results for computing Gröbner Bases *without* using the HFE parameter 37

7 MAGMA bugs **38**

A Time- and memory-tables **39**
A.1 Tables for $\mathbb{F}_3$ 39
A.2 Tables for $\mathbb{F}_5$ 40
A.3 Tables for $\mathbb{F}_7$ 41
A.4 Tables for $\mathbb{F}_{11}$ and $\mathbb{F}_2$ for comparison 42

Erklärung an Eides Statt

Hiermit versichere ich feierlich, dass ich die vorliegende Arbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt habe. Ich habe alle Stellen, die ich aus den Quellen wörtlich oder inhaltlich entnommen habe, als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 30. August 2007

0 Notations

$*$	We begin by defining the usual multiplication: "$*$". Sometimes "$\cdot$" is used.
k	k is defined as an arbitrary finite field. In specific, let $GF(q) = \mathbb{F}_q$ denote the finite field with q elements.
term	A term is a product of x_i's: x^α where $x = x_1x_2x_3...x_n$, $\alpha \in \mathbb{N}^n$, and $x^\alpha = x_1x_2...x_n^{(\alpha_1,\alpha_2,...,\alpha_n)} := x_1^{\alpha_1}x_2^{\alpha_2}...x_n^{\alpha_n}$
$\mathbb{T}^n$	For the variables $x_1, ..., x_n$, we define the set of terms as $\mathbb{T}^n := \{x_1^{\alpha_1}x_2^{\alpha_2}...n_n^{\alpha_n} \mid \alpha_i \in \mathbb{N}\}$
monomial	A monomial is a product $c * t$ where c is chosen from the respective underlying structure and t is a term.
polynomial	A polynomial is a sum of monomials: $p := \sum_i c_i * t_i$, where $c_i \in k, t_i \in \mathbb{T}^n$.
k_x	$k_x := k[x_1, ..., x_n]$ is the polynomial ring over k in n variables, $x_1, ..., x_n$, so it is a commutative ring with neutral 1-element.
K	$K = k[t]/g(t)$ is a level n algebraic extension of the ground field k, the polynomial ring k_x respectively, so it as a quotient of $k[t]$ modulo some irreducible polynomial $g(t)$ which has degree n.
$AC(K)$	$AC(K)$ is defined as the algebraic closure (i.e. the set of all algebraic numbers in K).
$T(m)$	For a monomial $m = c * x^\alpha$, we define $T(m) := x^\alpha$.
$deg(m)$	For a monomial $m = c * x^\alpha$, we define $deg(m) := \alpha$.
$T(p)$	For a polynomial, $\sum_i c_i * t_i$, we define the terms of the polynomal as $T(p) := \{t_i \mid c_i \neq 0\}$.
$T(M)$	For a set M consisting of monomials or polynomials, we define $T(M) := \{T(m) \mid m \in M\}$.
ideal	For a set of polynomials $G = \{g_1, ..., g_t\}$, we define the "ideal" of this set as $\langle G\rangle = \langle g_1, ..., g_t\rangle = \{c_1g_1 + ... + c_tg_t \mid c_i \in k_x\}$. In particular, $0 \in \langle G\rangle$. Actually, our definition is just one special case for an ideal and there are more general defintions of an ideal, but we will stick to this special definition, since all ideals which will be considered are of this structure.
$a\|b$	We define the relation "$\|$", "divides" as follows: $a\|b \iff \exists h \in k_x : b = h \cdot a$.
$S_1 \subset S_2$	We define the "subset" relation "$\subset$" of two sets S_1 and S_2 as $S_1 \subset S_2 \iff S_1 = S_2$ or $\forall x : (x \in S_1) \Rightarrow x \in S_2$.

A relation $<_\sigma$ on $\mathbb{N}^n$ is called a term ordering iff.

(i) $<_\sigma$ is a total ordering on $\mathbb{N}^n$.

(ii) $\alpha <_\sigma \beta, \gamma \in \mathbb{N}^n \Rightarrow \alpha + \gamma <_\sigma \beta + \gamma$.

(iii) The monomial 1 is the smalles: $\forall x^\alpha \neq 1 : 1 <_\sigma x^\alpha$.

We extend this definition on monomials in the following way: $x^\alpha <_\sigma x^\beta$ iff. $\alpha <_\sigma \beta$,

We define well known examples of term orderings:

- $\sigma = lex : \alpha <_{lex} \beta \iff$ the first nonzero entry of $\alpha - \beta \in \mathbb{Z}^n$ is negative.
- $\sigma = grlex : \alpha <_{grlex} \beta \iff |\alpha| = \sum_{i=1}^{n} \alpha_i < \sum_{i=1}^{n} \beta_i = |\beta|$ or $|\alpha| = |\beta|$ and $\alpha <_{lex} \beta$.

For a polynomial $p = \sum_i c_i x^{\alpha_i}$, we define:

The "**L**eading **T**erm"	$LT(p) := \max\{x_i^\alpha \mid c_i \neq 0\}$.
The "**L**eading **C**oefficient"	$LC(p) := \max\{c_i\}$.
The "**L**eading **M**onomial"	$LM(p) := LC(p) * LT(p)$.
The "**M**aximal **D**egree"	$MD(p) := deg(LT(p))$.

Normally, the leading term and monomial of a polynomial is defined with respect to a given term ordering σ. For the rest of this document, we assume that an arbitrary but fixed term ordering σ is given. This means, that it is chosen arbitrarily (with respect to the rules given above), but it must not change during a proof or algorithm.

Given a set (in particular an ideal) I, we define $LT(I) := \{LT(f) \mid f \in I\}$ as the set of leading terms.

1 Introduction

Due to improvements of Gröbner Basis-Algorithms such as the Faugère F_4/F_5-Algorithm, multivariate cryptosystems face new challenges in terms of algebraic attacks. This thesis will further explore questions related to these attacks, in particular on the HFE family. We will do an in-depth study of the complexity of the algebraic attack and adapt the system to see whether we can improve the original HFE-system such that it can resists the attack that makes use of the F_4/F_5-algorithm.

The thesis is divided into four parts: in the first section, a brief overview over multivariate public key cryptosystems (MVPKCS) will be given. In the second section, the design of the original HFE system will be introduced. In the third section, we will deal with the basic theory behind the general attack on MVPKCS, the Gröbner Basis theory. The Faugère F_4 algorithm will be introduced. In the last section, we will present our main results including the theoretical argumentation and experimental results adapting HFE. We will therefor show that over finite field having a sufficiently large characteristic, HFE can defeat the algebraic attacks.

1.1 Multivariate cryptosystems

Since the invention of quantum computers and algorithms, cryptographic researchers spared no effort on constructing cryptosystems for the post-quantum era. The security of HFE and multivariate cryptosystems in general relies on the structure of the underlying problem (called MQ-GF(2)) which has to be solved for breaking these systems. As we will see in section 2.3, MQ-GF(2) is $\mathcal{NP}$-hard. These systems are called multivariate because the public key and the underlying problem consist of a set of multivariate (in more than one variable) polynomials over a finite field. Although it is in principle $\mathcal{NP}$-hard to solve these systems as randomly chosen polynomials, it is sometimes possible to break the system using special algorithms which make use of the underlying structures.

The first "real" multivariate cryposystem was proposed by Matsumoto and Imai in [MI88]. The basic idea met two conditions:

- The system was able to create polynomials which were quadratic.
- While creating these polynomials in order to achieve $\mathcal{NP}$-hardness, decryption was still efficient by using a special trapdoor.

In [Pat95], Patarin has shown that the Matsumoto-Imai cryptosystem is insecure by a special algebraic attack using linearization equations. Later he proposed a related MVPKCS, the HFE cryptosystem, which he considered to be more secure. In 1999, Jean-Charles Faugère proposed an algorithm called F_4 [Fa99, Fa02] which can be used for attacking MVPKCS. In 2002, the first HFE challanges with relatively realistic parameters has been broken [FJo03] by the usage of F_4 in nearly two days.

2 HFE

In this section, a brief introduction to HFE will be given. HFE is an abbreviation for "**H**idden **F**ield **E**quation". In the following part of this section, a HFE cryptosystem as proposed by Patarin will be described. Afterwards, we will briefly present the proof for the $\mathcal{NP}$-hardness of MQ-GF(2).

2.1 Description of HFE

We begin by decsribing the HFE Cryptosystem. As in most public key cryptosystem, the overall process has to meet two conditions:

(1) Create a fast encryption function with a secret which is hard to invert.

(2) Create a private key from which the original plaintext can be restored relatively easy.

In this case, "inverting" the secret can be done by solving "MQ-GF(2)". It will consist of solving a system of polynomials which are at **M**ost **Q**adratic over **GF(2)**.

The whole system begins at a multivariate polynomialring over a (usually) small finite field k, having q elements. The polynomial ring will be called $k_x := k[x_1, ..., x_n]$. Then, k is extended using a new variable t. This extension is done by constructing the quotient concerning an irreducible polynomial $g(t)$ of dergee n over k. This means nothing else but doing a usual univariate polynomial division concerning the variable t. We will name the algebraic extension $K := k[t]/g(t)$. We can identify K as a n-dimensional vector space over k. The mapping is defined as $\phi : K \to (k)^n$ where

$$\phi(a_0 + a_1 t + ... + a_{n-1} t^{n-1}) \to (a_0, a_1, ..., a_{n-1}). \tag{2.1}$$

Eventually, there will be created a third level, a polynomial ring over K in the variable X called $K[X]$.

The key pair will be created as follows: First, two random invertible affine linear transofrmations $L_1(x_1, ..., x_n)$ and $L_2(x_1, ..., x_n)$ are chosen, which will each consist of a Matrix $A_i \in k^{n \times n}$ and a vector $b_i \in k^n$ for $i = 1, 2$. Afterwards, the following polynomial in $K[X]$ is created:

$$\tilde{F}(X) = \sum_{i=0}^{r_2} \sum_{j=0}^{i-1} \alpha_{ij} X^{q^i + q^j} + \sum_{i=0}^{r_1} \beta_i X^{q^i} + \gamma, \tag{2.2}$$

where the coefficients, respectively the constant $\alpha_{ij}, \beta_i, \gamma$ are randomly chosen elements in $(k[t]/g(t))$, so all coefficients will just consist of t-polynomials, not of x_i's. The private key wil be $\{L_1, L_2, \tilde{F}(X)\}$. The public key is calculated by mapping a vector in k_x^n through $\tilde{F}$ by interpreting it as a function. So the public key is created by computing

$$F(x_1, ..., x_n) = (L_1 \circ \phi \circ \tilde{F} \circ \phi^{-1} \circ L_2)(x_1, ..., x_n), \tag{2.3}$$

which produces n polynomials in k_x called $f_1, ..., f_n$. Please note, that "computing" $F(x_1, ..., x_n)$ means evaluating all the functions for **general** x_i's, so they all do not have a concrete value.

2.1.1 $\tilde{F}(X)$ produces at most quadratic polynomials

Consider the first sum of $\tilde{F}(X)$, namely $\sum_{i=0}^{r_2}\sum_{j=0}^{i-1}\alpha_{ij}X^{q^i+q^j}$. When creating the public key, each substitutions for X will be an element in $k[t]/g(t)$. It follows that every $(\phi^{-1}\circ L_2)(x_1,...,x_n)$ will consist of linear combinations of x_i's (coming from the first affine linear transformation) as coefficients for the extension variable t. These substitutions of X will always look like

$$p := l_0 + l_1 * t + l_2 * t^2 + ... + l_{n-1} * t^{n-1},$$

where $l_i := \alpha_1(i)x_1 + ... + \alpha_n(i)x_n$, where $\alpha_i(j) \in k$. Note that taking a polynomial such as p to the power of q is a **linear function** in terms of finite fields. This works due to facts which will be proven now.

Theorem. *Let k be a finite field with characteristic q, then*

$$(x+y)^q = x^q + y^q, \text{for } x, y \in k. \tag{2.4}$$

Proof. From the Pascals triangle (from the binomial coefficients respectively), we know that $(x+y)^q = \sum_{k=0}^{q}\binom{q}{k}x^{q-k}y^k$. We will show that $q \mid \binom{q}{k}$ for $1 \le k \le n-1$, since then $\binom{q}{k} = 0$ for $1 \le k \le n-1$. So consider such an $\binom{q}{k} = \frac{q!}{k!(q-k)!}$. Using the prime factors of the nominator and the denominator, this can be rewritten as $\binom{q}{k} = \frac{q\cdot q_1\cdot ...\cdot q_i}{p_1\cdot ...\cdot p_j}$, where all p_i, q_i are prime numbers and $p_i < q$ for all p_i. Since we know, that $\binom{q}{k} \in \mathbb{N}$ and q is a prime number and $p_i < q$ (so $gcd(q, p_1 \cdot ... \cdot p_j) = 1$), it follows that $h = \frac{q_1\cdot ...\cdot q_i}{p_1\cdot ...\cdot p_j} \in \mathbb{N}$ (just leaving out q in the numerator). We have found a natural number h, for which $\binom{q}{k} = h * q$ holds. This means, that $q \mid \binom{q}{k}$ and therefor $\binom{q}{k} = 0$ for $1 \le k \le n-1$. Consequently,

$$\begin{aligned}(x+y)^q &= \sum_{k=0}^{q}\binom{q}{k}x^{q-k}y^k \\ &= \binom{q}{0}x^{q-0}y^0 + \binom{q}{1}x^{q-1}y^1 + ... + \binom{q}{q-1}x^{q-(q-1)}y^{q-1} + \binom{q}{q}x^{q-q}y^q \\ &= x^q + 0 + ... + 0 + y^q \\ &= x^q + y^q.\end{aligned}$$

□

It is important to note that not only k, the base field, has characteristic q, but all structures which are based on k, such as k_x, $k[t]/g(t)$, etc. Inductively, we can apply this argumentation on bigger sums and on exponents which are a multiple of q, so $(s_1 + s_2 + ...s_k)^{q^i} = (s_1 + [s_2 + ... + s_k])^{q^i} = s_1^{q^i} + [s_2 + ... + s_k]^{q^i} = ... = s_1^{q^i} + ... + s_k^{q^i}$.

Furthermore, we know that

$$l_i^q = l_i^{(q \mod (q-1))} = l_i. \tag{2.5}$$

This holds, since all $l_i \in k_x$ and k (the underlying structure on which k_x is based on) has exactly q elements. Hence, it follows that $k \setminus \{0\}$ forms a group which has cardinality $q-1$. From the theorems of Lagrange and Euler [Buc04], we know that in all groups G, for all $g \in G : g^{|G|} = 1$. This means that for all $i \in \mathbb{N}$:

$$\begin{aligned} p^{q^i} &= (l_0 + l_1 * t + l_2 * t^2 + ... + l_{n-1} * t^{n-1})^{q^i} \\ &= l_0^{q^i} + (l_1 * t)^{q^i} + (l_2 * t^2)^{q^i} + ... + (l_{n-1} * t^{n-1})^{q^i} && \text{from (2.4)} \\ &= l_0 + l_1 * (t)^{q^i} + l_2 * (t^2)^{q^i} + ... + l_{n-1} * (t^{n-1})^{q^i} && \text{from (2.5)} \end{aligned}$$

We have shown that X^{q^i} is a linear function. Another interpretation is that it leaves the x_i's (respectively the l_i's) untouched. Taking X to the q^i solely controls the distribution of $x_1, ..., x_n$. Additionally, we know that $X^{q^i+q^j} = X^{q^i} * X^{q^j}$ is a product of two linear functions, so it is at most quadratic in terms of the x_i's.

2.1.2 Encryption

Encrypting a given plaintext $(x'_1, ..., x'_n)$ is done by evaluating the public key polynomials:

$$\begin{aligned} y_1 &= f_1(x'_1, ..., x'_n) \\ y_2 &= f_2(x'_1, ..., x'_n) \\ &\vdots \\ y_n &= f_n(x'_1, ..., x'_n) \end{aligned}$$

Where the ciphertext is $c = (y_1, ..., y_n)$. Condition (1) is met by HFE, because evaluating a set of n functions over a small finite field k is efficient.

2.1.3 Decryption

First, $Y := (\bar{y_1}, ..., \bar{y_n}) = L_1^{-1}(c_1, ..., c_n)$ for a given ciphertext $(c_1, ..., c_n)$ is computed. Please note, that we need to look at the private key as an element of $(k[t]/g(t))[X]$ (*not!* in $(k_x[t]/g(t))[X]$) now, a polynomial ring over [polynomials over k in the variable t modulo $g(t)$] in X.

Then, the set

$$\mathcal{Z} = \{Z \in (k[t]/g(t))[X] \,|\, \tilde{F}(Z) = Y \iff \tilde{F}(Z) - Y = 0\}$$

is computed. The equation $\tilde{F}(Z) - Y = 0$ can be solved by the factorization of $\tilde{F}(Z) - Y$ with respect to X using the Berlekamp algorithm [Be70]. This can be done efficiently, if r_1 and r_2 (which determine the degree of the private key polynomial) are not too high, because the complexity of the Berlekamp algorithm depends on the degree of the input polynomial. So condition nr. (2) is also met.

As a last step, for all solutions $z \in \mathcal{Z}$, $\bar{z} = L_2^{-1} \circ \phi(z)$ is computed. The plaintext is among the $\bar{z}$'s, but in some cases, we do not know, which one it is. Therefor, some additional information like a hash value or similar is needed in order to find the original plaintext. This problem occurs, because the map $\tilde{F}$ is generally chosen and therefor not bijective.

2.2 Example of an HFE system

We will now present a toy example for an HFE cryptosystem.

Example 2.1. We chose n to be 3, $r_2 = 2, r_1 = 1$ (so D will be $6 = 2^2 + 2^1$), the term ordering to be $\sigma = grlex$ and $g(t) = t^3 + t + 1$. The public key will consist of 3 polynomials in 3 variables. For simplicity, we chose L_1 and L_2 to be the identity function, so

$$L_1 \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = L_2 \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

For the random parameters in $\tilde{F}(X)$, we chose

$$\begin{aligned} \alpha_{10} &= t \\ \alpha_{20} &= t^2 + t + 1 \\ \alpha_{21} &= t^2 + t + 1 \\ \beta_0 &= t^2 + t + 1 \\ \beta_1 &= 1 \\ \gamma &= t \end{aligned}$$

According to the definition we have

$$\tilde{F}(X) = (t^2 + t + 1)X^6 + (t^2 + t + 1)X^5 + (t)X^3 + X^2 + (t^2 + t + 1)X + (t)$$

The public key is created by evaluating $(L_1 \circ \phi \circ \tilde{F} \circ \phi^{-1} \circ L_2)(x_1, x_2, x_3) = (\phi \circ \tilde{F} \circ \phi^{-1})(x_1, x_2, x_3)$, where $\phi^{-1}(x_1, x_2, x_3) = x_1 + tx_2 + t^2x_3$. Eventually, $\phi \circ \tilde{F}(x_1 + tx_2 + t^2x_3)$ produces the public key

$$\begin{aligned} f_1(x_1, x_2, x_3) &= x_1x_2 + x_1x_3 + x_3 \\ f_2(x_1, x_2, x_3) &= x_1x_2 + x_2x_3 + x_2 + x_3 + 1 \\ f_3(x_1, x_2, x_3) &= x_1 + x_3 \end{aligned}$$

From the encryption of the plaintext $p = (0, 1, 1)$ with $\phi^{-1}(p) = (0 * t^0 + 1 * t^1 + 1 * t^2) = t^2 + t$ we obtain the ciphertext $c = (c_1, c_2, c_3)$, where

$$\begin{aligned} c_1 &= f_1(0, 1, 1) = 1 \\ c_2 &= f_2(0, 1, 1) = 0 \\ c_3 &= f_3(0, 1, 1) = 1 \end{aligned}$$

So $\phi^{-1}(c) = t^2 + 1$. Factorizing $\tilde{F}(X) - (t^2 + 1)$ we derivate the following list of factors

$$\begin{aligned} p_1 &= t^2 + t + 1 \\ p_2 &= X^2 + (t^2 + 1)X + (t^2 + t) \\ p_3 &= X^2 + (t^2 + t)X + (t^2 + t) \\ p_4 &= X + (t^2) \\ p_5 &= X + (t^2 + t) \end{aligned}$$

Since p_1 is constant and p_2 and p_3 are irreducible, they can never be zero. $p_4 = 0 \iff X = t^2$ and $p_5 = 0 \iff X = t^2 + t$. Here we need the additional information (like a hash value) in order to see that $t^2 + t$ was the original plaintext. We succesfully recovered p using the factorzation of $\tilde{F}(X) - \phi^{-1}(c) = 0$. A Singular script which demonstrates the usage of the values given above can be found in [Wer07a].

2.3 MQ-GF(2) is $\mathcal{NP}$-hard

We will now present the assumption on which the security of MVPKCS is based on: the $\mathcal{NP}$-hardness of MQ-GF(2). In order to do so, we will reduce "$3\mathcal{SAT}$" to MQ-GF(2) in polynomial time. Therefor we show that everybody who can solve MQ-GF(2) in polynomial time is also able to solve $3\mathcal{SAT}$ in polynomial time. This means that he or she would be able to solve many $\mathcal{NP}$-hard problems in polynomial time in general, because numerous other problems can be reduced to $3\mathcal{SAT}$. The proofs for $\mathcal{NP}$-hardness of $3\mathcal{SAT}$ and $\mathcal{SAT}$ itself can be found in [McC05] and [Coo71].

We define the following:

- A *literal* is either a variable or a negation of a variable (x or $\neg x$ for example).
- A *clause* is a disjunction of literals ($x \vee \neg y \vee z \vee \neg w$ for example).
- A *formula* ϕ is in CNF (Conjunctive Normal Form) iff it is a conjunction of clauses ($(x \vee \neg z) \wedge (\neg y) \wedge (w \vee \neg z)$ for example).

Definition 2.1. $3\mathcal{SAT} := \{\phi \mid \phi$ is a *satisfiable* 3CNF-formula$\}$, where ϕ is a 3CNF formula $\iff \phi$ is in CNF and every clause contains at most 3 literals.

The actual problem is to estimate, whether an arbitrarily chosen formula in n variables $\phi(x_1, ..., x_n)$ is satisfiable or not: $\phi(x_1, ..., x_n) \in 3\mathcal{SAT} \iff \exists\, c_1, ..., c_n : \phi[c_1, ..., c_n]$ evaluates to "true".

Theorem 2.2. *MQ-GF(2) is $\mathcal{NP}$-hard.*

Proof. For an arbitrary satisfiable 3CNF-formula ϕ we transform the clauses in the following way:

Let $c = (x_i \vee x_j \vee x_k)$ be a clause in ϕ, where x_i, x_j, x_k are literals, then we reduce the clause to

$$c' := (z_i + z_j + z_k + z_i z_j + z_j z_k + z_i z_k).$$

Let $c = (x_i \vee x_j)$ be a clause in ϕ, where x_i, x_j are literals, then we reduce the clause to

$$c' := (z_i + z_j + z_i z_j).$$

Let $c = (x_i)$ be a clause in ϕ, where x_i is a literal, then we reduce the clause to

$$c' := (z_i)$$

where we set

$$z_i = \begin{cases} x_i, & \text{if } x_i \text{ is a positive literal} \\ (1 - x_i), & \text{if } x_i \text{ is a negated literal} \end{cases}$$

It is easy to verify, that $c' = 1$ over GF(2) $\iff$ c evaluates to "true". For the whole formula $\phi = c_1 \wedge c_2 \wedge ... \wedge c_n$, we define $\phi' = c'_1 \cdot c'_2 \cdot ... \cdot c'_n$. Because $c'_i = 1$ over GF(2) $\iff$ c_i evaluates to "true" for all clauses c_i in ϕ, the following also holds: $\phi' = 1$ over GF(2) $\iff$ ϕ evaluates to "true". This shows, that we are able to reduce $3\mathcal{SAT}$ to MQ-GF(2), and therefor, that MQ-GF(2) is $\mathcal{NP}$-hard. □

One should note, that the $\mathcal{NP}$-hardness for MQ in general can be shown for a more abstract context too (like commutative rings for example). More information on this can be found in [Wol02]. Although the first intention of HFE and most of the other multivariate public key cryptosystems was to create a general instance of MQ-GF(2), they actually create a very special instance. As we will see later: the underlying structure can be exploited to break the public key of HFE.

3 Gröbner Basis Theory

In this section we will give a quick overview of the current development of the so-called Gröbner Basis-theory. Bruno Buchberger initiated the Gröbner Basis-theory and he was the first one who specified an algorithm for finding Gröbner Bases in 1965. In the first part of this chapter, we will show one of the applications of Gröbner Bases. We will then proceed presenting the proof for the existence of a Gröbner Basis for an arbitrary ideal. In the second part, we will introduce and analyze two algorithms for computing such bases.

3.1 Why are Gröbner Bases interesting?

By recalling the encryption process of HFE from section 2.1.2, we can see that decryption can be done by finding a solution (values for $x_1, ..., x_n$) such that

$$\begin{aligned} c_1 &= f_1(x_1, ..., x_n) \\ c_2 &= f_2(x_1, ..., x_n) \\ &\vdots \\ c_n &= f_n(x_1, ..., x_n) \end{aligned}$$

We assume that an attacker knows $c = (c_1, ..., c_n)$ and $f_1, ..., f_n$, since a realistic attacker model for a public key cryptosystem allows public access to all transmissions and to the public key. Equivalently to the system above, the attacker could also solve the system

$$\begin{aligned} f_1(x_1, ..., x_n) - c_1 &= 0 \\ f_2(x_1, ..., x_n) - c_2 &= 0 \\ &\vdots \\ f_n(x_1, ..., x_n) - c_n &= 0 \end{aligned}$$

This is the point, where Gröbner Bases start to play a crucial role, because they can be used for solving this system, therefor obtain values for $x_1, ..., x_n$, the plaintext.

Consider an arbitrary set of generators $G = \{g_1, ..., g_t\}$ for an ideal $I \subset k[x_1, ..., x_n]$. We define

$$G \text{ is a Gröbner Basis for } I \text{ if } \quad \langle LT(I) \rangle = \langle LT(g_1), ..., LT(g_t) \rangle . \tag{3.1}$$

In other words: every leading term in the ideal I has to be divisible by a leading term of at least one g_i. In current literature, one often finds descriptions in which Gröbner Bases are stated to have certain "nice properties". Indeed, all these nice properties are direct consequences of the simple formalization above. We will now discover one of these properties and how it can be used to solve a system of polynomials. This process consists of two steps: the so-called *elimination of variables* and the *successive completion*.

3.1.1 Elimination of variables

We first need the follwing definition:

Definition 3.1. For an ideal I, we define the ideal $I_l := I \cap k[x_l, ..., x_n]$.
So I_l will contain all $i \in I$, which only consist of the variables $x_l, ..., x_n$.

. The very first and intuitive idea would be to reduce the number of variables in order to end up with a univariate equation. The following theorem will guarantee the correctness of this idea.

Theorem 3.2. *Fix the term ordering to be $\sigma = lex$. Then, if G is a Gröbner Basis of an Ideal $I \subset k[x_1, ..., x_n]$ then $G_l = G \cap k[x_l, ..., x_n]$ is a Gröbner Basis for $I_l = I \cap k[x_l, ..., x_n]$.*

Proof. We will show that $i \in I_l \iff LT(i) = h * LT(g)$ for some $h \in k[x_l, ..., x_n], g \in G_l$.

"$\Rightarrow$"
Consider an arbitrary $i \in I_l \subset I$. We know that $LT(i) \in k[x_l, ..., x_n]$, because $i \in I_l$. Furthermore, we know that $\exists g \in G$ such that $LT(i) = h * LT(g)$ where $h, LT(G) \in k[x_l, ..., x_n]$, because G is a Gröbner Basis of I. It follows, that no other monomial of g may consist of $x_1, ..., x_{l-1}$, because $\sigma = lex$. This means that $g \in G_l$, since $g \in k[x_l, ..., x_n]$. We have shown, that every $LT(i) \in LT(I_l)$ can be generated by a $LT(g)$ for $g \in G_l$.

"$\Leftarrow$"
We assume that $LT(i) = h * LT(g)$ for some $g \in G_l$. Since $h \in k[x_l, ..., x_n]$ and $LT(g) \in k[x_l, ..., x_n]$, we know that $i \in k[x_l, ..., x_n]$. Since not only $g \in G_l$ but also $g \in G$ (because $G_l \subset G$) and G is a Gröbner Basis of I, we know that i also has to be in I. But $i \in I$ and $i \in k[x_l, ..., x_n]$ implies that $i \in I_l$.

This proves that G_l is a Gröbner Basis for I_l, because $\langle LT(I_l)\rangle = \langle LT(G_l)\rangle$. □

Please note, that the assumtion $\sigma = lex$ is not necessary, since there are algorithms which are able to eliminate variables for an arbitrary term ordering (see [FJL06]).

Now consider the ideal $I = \langle f_1(x_1, ..., x_n) - c_1, ..., f_n(x_1, ..., x_n) - c_n\rangle$. By applying the theorem above sucessively, we are able to eliminate variable by variable until we will reach the equation $\bar{f_n}(x_n)$, which will consist of one single variable. There always exists such an equation, because the system has at least one solution and typically not more than eight solutions.

So we do this step forwards for $i = 2, ..., n$ and therefor set up equations $\bar{f_i}(x_i, ..., x_n)$.

3.1.2 Successive completion

Consider the equation in just one variable, constructed as given above. We can now solve $\bar{f}_l(x_l) = 0$, so we will set up the variety of I_l:

$$V(I_l) = \{(a_l, ..., a_n) \mid \bar{f}_l(a_l) = 0,\ a_{l+1}, ..., a_n \in k\}$$

We define the variety in this way, because the overall system $f_1, ..., f_n$ could have more than one solution. In specific, it could be the case that $l \neq n$, so $f_1, ..., f_n$ will be zero for all $a_{l+1}, ..., a_n \in k$. Afterwards, we will consider I_{l-1} and continue the successive completion with the variety of I_{l-1}. Generally, for a variety $V(I_k)$ of I_k, we define the variety of I_{k-1} as

$$V(I_{k-1}) := \{(a_{k-1}, ..., a_n) \mid (a_k, ..., a_n) \in V(I_k), a_{k-1} \in k, \bar{f}_{k-1}(a_{k-1}, ..., a_n) = 0\}$$

This step will be repeated backwards for $i = l, ..., 1$. The variety $V(I_1) = V(I)$ then contains all solutions for

$$\begin{aligned} f_1(x_1, ..., x_n) - c_1 &= 0 \\ f_2(x_1, ..., x_n) - c_2 &= 0 \\ &\vdots \\ f_n(x_1, ..., x_n) - c_n &= 0 \end{aligned}$$

The plaintext is among these solutions. It will be identified using the redundant information source of HFE, since the real private key holder must also be able to estimate the real plaintext among these solutions. So we have shown, how HFE can be broken using a "Gröbner Basis-attack". The essential questions are: How to compute such bases for public keys of HFE? Is the existence of such a basis guaranteed for every ideal? These questions will be answered in the next sections.

3.2 Example for breaking HFE

Before introducing all the theory in behind, we will use what we know so far in order to recover the plaintext from example 2.1 without any knowledge of the private key.

Example 3.1. Continuing example 2.1, we will now estimate a Gröbner Basis for the ideal defined by $\langle f_1 - c_1, f_2 - c_2, f_3 - c_3 \rangle$. Using any of the introduced algorithms below, this results in

$$\begin{aligned} g_1 &= x_3 + 1 \\ g_2 &= x_1 + x_3 + 1 \end{aligned}$$

Because of theorem 3.2 we now know that solving $g_1 = 0, g_2 = 0$ simultaneously will result in values x_1, x_2, x_3 for which $f_i(x_1, x_2, x_3) = c_i$ holds. We do this by starting with

g_1, because it is univariate. We know that $x_3 = 1$. By substituting this in $g_2 = 0$, we see that $x_1 = 0$. For x_2 there is no constraint, so we can set this to an arbitrary value in k. These were exactly the solutions from example 2.1. The additional informations helps us to see that $x_1 = 0, x_2 = 1, x_3 = 1$ was the original plaintext.

Again, a Singular script which demonstrates this calculation can be found in [Wer07a].

We will now introduce the necessary theoretical background behind Gröbner Bases and their calculation.

3.3 Existence of Gröbner Bases

In this section, we will prove the existence of a Gröbner Basis for an arbitrarily chosen ideal $I \subset k[x_1, ..., x_n]$.

3.3.1 Dicksons Lemma

The following lemma is essential for showing the existence of Gröbner Bases.

Theorem 3.3. *(Dicksons Lemma) For every (possibly inifite) sequence of monomials $(t_1, t_2, ...)$, where $t_i \in \mathbb{T}^n$ and the monomial ideal $I = \langle t_1, t_2, ... \rangle$, there exists an $N \in \mathbb{N}$ such that $I = \langle t_1, ..., t_N \rangle$.*

Proof. Following the idea of [Buc96], We will prove Dicksons Lemma by induction on n, so we are doing induction on the number of variables. Please recall the definition $k_x = k[x_1, ..., x_n]$ to be the polynomial ring over a finite field k in n variables $x_1, ..., x_n$.

For $n = 1$, we are operating in $k[x]$. Now select $t = x^i$ such that t has a minimal x-exponent. Now, all x^j where $j > i$ can be generated by $x^j = t * x^{j-i}$.

For $n \to n + 1$, we are operating in $k[x_1, ..., x_n, y]$. For simplifying the notation, we set $x^\alpha := x_1^{\alpha_1} ... x_n^{\alpha_n}$. For every term $t = x^\alpha y^\beta, \beta \in \mathbb{N}$ set $t' := x^\alpha$ (leaving out the y-exponent), the projection on $k[x_1, ..., x_n]$. Consider the ideal $\langle t'_1, t'_2, ... \rangle$. According to the induction hypothesis $\langle t'_1, t'_2, ... \rangle$ is finitely generated by some $J = \{x^{\beta_1}, ..., x^{\beta_N}\}$. Set $m_i = \min\{e \,|x^{\beta_i} y^e,\ x^{\beta_i} \in J\}$ and $m = \max(m_i)$ a fixed number. Please note, that m is *not* the maximum y-exponent, since this may not exists, because the y-exponents can grow infinitely. Now, set $J_l = \{x^\gamma | x^\gamma y^l \in I\}$ for $0 \leq l \leq m-1$. From the induction hypothesis, we know that all $\langle J_l \rangle$'s are finitely generated by some generators $\{x^{\gamma_1(l)}, ..., x^{\gamma_{N'}(l)}\}$. Eventually, we can write the set of generators for I as the union

$$G := \{x^{\beta_i} y^m | x^{\beta_i} \in J\} \cup \left(\bigcup_{0 \leq l \leq m-1} \{x^{\gamma_1(l)} y^l, ..., x^{\gamma_{N'}(l)} y^l\} \right).$$

For showing, that G is really the set of generators for I, consider two cases:

$$x^\alpha y^p \in I = \begin{cases} p \geq m \\ p < m \end{cases}$$

For $p \geq m$, $\exists x^{\beta_i} \in J$ such that $x^\alpha = h \cdot x^{\beta_i}$ for $h \in k_x$. So $x^\alpha y^p$ can be generated by $h \cdot x^{\beta_i} \cdot y^{p-m}$. For $p < m$, $\exists x^{\gamma_i(p)} y^p \in J_p$ such that $x^\alpha = h \cdot x^{\gamma_i(p)}$ for $h \in k_x$. So $I \subseteq \langle G \rangle$, because every $i \in I$ can be generated by some $g \in G$. Now, we show that $\langle G \rangle \subseteq I$. Every $f \in \langle G \rangle \iff f = h_1 g_1 + ... + h_{N''} g_{N''}$ is also in I, because all $g_i \in I$ and all $h_i \in k_x$. □

A direct consequence of Dicksons Lemma is the following:

Theorem 3.4. *(Ascending Chain Condition, ACC)*
Let $I_1 \subset I_2 \subset ...$ be a sequence of monomial *ideals, then $\exists N \in \mathbb{N}$ such that $I_N = I_{N+1} = I_{N+2} = ...$*

Proof. Consider the union $\bigcup_j (I_j)$. According to Dicksons Lemma, this union is generated by a finite set $\{g_1, ..., g_t\}$. Set $N := \min\{m | \langle g_1, ..., g_t \rangle = I_m\}$, then $I_N = I_{N+1} = I_{N+2}$, because $\{g_1, ..., g_t\}$ also generates the union $I_N \cup I_{N+1} \cup ....$ □

We will use the ACC in section 3.4.2, in order to show, that *Buchbergers algorithm* terminates.

3.3.2 Hilberts Basis Theorem

We will now show the existence of a Gröbner Basis for an arbitrary ideal I. The following Theorem was first stated by David Hilbert:

Theorem 3.5. *(Hilberts basis theorem)*
For every Ideal $I \subset k[x_1, ..., x_n]$, there exists a Gröbner Basis $G = \{g_1, ..., g_t\}$ such that $\langle G \rangle = I$.

Proof. Since $\langle LT(I) \rangle$ is a monomial ideal, we can apply Dicksons Lemma (Theorem 3.3) which states, that every monomial ideal is finitely generated. Let $G = \{g_1, ..., g_t\}$ be a set of polynomials such that $\langle LT(I) \rangle = \langle LT(g_1), ..., LT(g_t) \rangle$, then G is a Gröbner Basis of I.

We now have to show that $i \in I \iff i \in \langle G \rangle$ since then $I = \langle G \rangle$ holds.

"$\Rightarrow$"
We show this by assuming, that there exists at least one f for which $f \in I$ but $f \notin \langle G \rangle \iff f \in I \setminus \langle g_1, ..., g_t \rangle$ holds. Select f such that f has minimal leading term. Since $\langle LT(I) \rangle = \langle LT(g_1), ..., LT(g_t) \rangle$, it follows, that $LM(f) = h \cdot LT(g_i)$ for $h \in k_x$ and some $i \in \{1, ..., t\}$. We estimate $(f - h \cdot g_i)$ which is still in $I \setminus \langle g_1, ..., g_t \rangle$. This is a contradiction, since $(f - h \cdot g_i)$ has a smaller leading term than f, which was assumed to belong to I and have a minimal leading term.

"$\Leftarrow$"
We can proof it exactly as in (Theorem 3.3). □

In other words, this tells us that we can repeat estimating $f_i = (f_{i-1} - h \cdot g_i)$ for any $f_0 \in I$ and some $g_i \in G$ succesively, and all $f_i \in I$. Eventually, f_i will be reduced to 0, which is in $\langle LT(g_1), ..., LT(g_t) \rangle$ by definition.

One should realize, that Hilberts Theorem does not provide a constructive method for computing a Gröbner Basis for a given ideal. This will be the task of the Buchberger Algorithm which will be introduced in section 3.4.2.

3.4 Estimating Gröbner Bases

After Hilbert has shown the existence of Gröbner Bases, it should take until 1965, where Bruno Buchberger was able to postulate an algorithm for estimating Gröbner Bases and prove its correctness. In the next section, we will prove Buchbergers criterion, on which the argumentation of correctness is mainly based on. Afterwards, we will introduce the *Buchberger algorithm* and prove its correctness and termination. In the last section, the Faugère F_4 *algorithm*, a powerful enhancement in terms of performance, will be introduced. Using this algorithm, Faugère was able to solve the first HFE challenge in 80 variables in a reasonable amount of time.

3.4.1 The Buchberger criterion

In this section, we wil prove the so-called Buchberger criterion. This will be used in order to show the correctness of the Buchberger Algorithm in the next section. For the following Lemma and the proof of the Buchberger criterion, we will need a definition of a polynomial division in more than one variable. A definition can be found in [DGS06] (pp. 199-203). Using this algorithm and its properties, the following Lemma will provide another charaterization for a Gröbner Basis.

Lemma 3.6. *For all Ideals $I = \langle g_1, ..., g_t \rangle$, where $G = \{g_1, ..., g_t\}$ is a Gröbner Basis of I, all multivariate divisions of all $f \in I$ by G will result in a remainder being* 0.

Proof. Theorem 3.5 provides an approach for making the leading term smaller. This step can be repeated and is bounded due to the term ordering. □

A key idea of the Buchberger theory in general is a simple construction called "S-Polynomial". The S-Polynomial of two polynomials g_i, g_j is defined as:

$$SPoly(g_i, g_j) = \frac{LCM(LM(g_i), LM(g_j))}{LM(g_i)} * g_i - \frac{LCM(LM(g_i), LM(g_j))}{LM(g_j)} * g_j$$

$LCM(x, y)$ is defined as the least common multiple of x and y. Obviously, these S-Polynomials make the leading term of g_i and/or g_j vanish. The following Lemma will show us, that S-Polynomials are even more powerful. It will show, that whenever a leading term in a sum of polynomials vanishes, the sum can be transferred into a representation which makes use of S-Polynomials.

Lemma 3.7. *Let $f = \sum_{i=1}^{t} c_i x^{\alpha_i} g_i$ where $MD(x^{\alpha_i} g_i) = \delta$ and $c_i \in k$ (so all c_i's are constants). Then $\exists c_{ij} \in k$ such that $\sum_{i=1}^{t} c_i x^{\alpha_i} g_i = \sum_{i=1}^{t-1} c_{ij} x^{\alpha_{ij}} S(g_i, g_j)$ where $j := i+1$ and $MD(x^{\alpha_{ij}} S(g_i, g_j)) < \delta$.*

There are two things which are important to notice about this Lemma:

- Whenever there are leading terms nullifying each other, it can be emulated by substituting S-Polynomials into the sum.
- These S-Polynomials (which are used to emulate the neutralization of the leading terms) already have a degree lower than δ.

Both of the properties will be used in the proof of the Buchberger criterion. We will omit the proof for this Lemma, as it can be found in [CLS96].

Theorem 3.8. *(Buchbergers criterion)*
Let $I \subset k_x$ be an ideal defined by $I = \langle G \rangle$ and $G = \{g_1, ..., g_t\}$. Then, G is a Gröbner Basis of $I \iff \forall i, j \ (i \neq j) : S(g_i, g_j)$ leaves remainder 0 when divided by G.

Proof. (see also [BB65], [CLS96])

"$\Rightarrow$"
If G is a Gröbner Basis of I, then $S(g_i, g_j) \in I$. According to Lemma 3.6, the remainder of the division will be 0.

"$\Leftarrow$"
Let $f \in I$, then f can be expressed as $f = \sum_{i=1}^{t} h_i g_i$ for $h_i \in k_x, g_i \in G$. The key idea will be, to show that $MD(f) = \max(MD(h_i g_i)) = \delta$ *if δ is chosen to be minimal*, since then the leading term of f is in $\langle LT(g_1), ..., LT(g_t) \rangle$. This holds, because $LT(f) = LT(h_i g_i) \Rightarrow LT(f) = LT(h_i) LT(g_i)$, so we are able to express $LT(f) = h \cdot LT(g_i)$ for some $h \in k_x$. In general, $MD(f)$ can only be smaller or equal δ, because $MD(f) > \delta$ contradicts $f = \sum_{i=1}^{t} h_i g_i$. It is left to show, that $MD(f)$ cannot be smaller than δ. So from here, we assume for a contradiction, that $MD(f) < \delta$. We set $m_i := MD(h_i g_i)$ and obtain

$$\begin{aligned} f &= \sum_{i=1}^{t} h_i \cdot g_i \\ &= \sum_{m_i=\delta} h_i g_i + \sum_{m_i<\delta} h_i g_i \\ &= \sum_{m_i=\delta} LT(h_i) g_i + \sum_{m_i=\delta} (h_i - LT(h_i)) \cdot g_i + \sum_{m_i<\delta} h_i g_i \end{aligned} \tag{3.2}$$

Consider the first sum $\sum_{m_i=\delta} LT(h_i) g_i$. All the summands have $MD = \delta$, but according to the assumtion, that $MD(f) < \delta$, the whole first sum has to have $MD < \delta$ (because the second and third sum already have $MD < \delta$). This means, that the first sum has exactly the form of Lemma 3.7. From the Lemma, we then know that

$$\sum_{m_i=\delta} LT(h_i) g_i = \sum_{i=1}^{t-1} c_{ij} x^{\alpha_{ij}} S(g_i, g_j), \tag{3.3}$$

where $MD(x^{\alpha_{ij}} S(g_i, g_j)) < \delta$. Now we are using the assumtion, that all the S-polynomials can be divided by G, which means, that

$$S(g_i, g_j) = \sum_{l=1}^{t} b_l g_l \tag{3.4}$$

for all S-polynomials. A direct consequence of the division algorithm [DGS06] is, that $MD(b_l g_l) \leq MD(S(g_i, g_j))$ which implies, that $MD(b_l g_l) < \delta$. So now, we substitute this back into (3.2), which results in:

$$\begin{aligned}
\sum_{m_i=\delta} LT(h_i) g_i &= \sum_{j=1}^{t-1} c_{ij} x^{\alpha_{ij}} S(g_i, g_j) && \text{from (3.3)} \\
&= \sum_{j=1}^{t-1} c_{ij} x^{\alpha_{ij}} \left(\sum_{l=1}^{t} b_l g_l \right) && \text{from (3.4)} \\
&= \sum_{l=1}^{t} \left(\sum_{j=1}^{t-1} c_{jk} x^{\alpha_j k} \right) b_l g_l && \text{(resorting the sum)} \\
&= \sum_{l=1}^{t} \tilde{h_l} g_l && (3.5)
\end{aligned}$$

It is important to notice, that all the summands $\tilde{h_l} g_l$ of the last sum still have $MD< \delta$. So by substituting (3.5) back into (3.2), f can be expressed as

$$\sum_{l=1}^{t} \tilde{h_l} g_l + \sum_{m_i=\delta} (h_i - LT(h_i)) \cdot g_i + \sum_{m_i<\delta} h_i g_i. \tag{3.6}$$

From the beginning, we know that $f = \sum_{i=1}^{t} h_i g_i$, where $\max(MD(h_i \cdot g_i)) = \delta$. Since every summand of (3.5) has $MD< \delta$, this is a contradiction, because as we have chosen δ to be minimal, $\max(MD(\tilde{h_l} g_l)) = \delta < \delta = \max(MD(h_i g_i))$. Therefor, $MD(f) = \delta$, which implies, that $LT(f) \in \langle LT(g_1), ..., LT(g_t) \rangle$. This states, that G is indeed a Groebner basis, because for any arbitrily chosen $f \in I : LT(f) \in \langle LT(g_1), ..., LT(g_t) \rangle$. □

3.4.2 Buchbergers algorithm

In contrast to Hilbert, Buchberger also provided an algorithm which calculates a Gröbner Basis for a given Ideal I. We will present a slightly modified version of the algorithm:

Algorithm: Buchberger

INPUT: A set $F := \{f_1, ..., f_t\}$ which defines the ideal $I = \langle f_1, ..., f_t \rangle$.
OUTPUT: $G = \{g_1, ..., g_{t'}\}$, which is a Gröbner Basis of I.

$G_{next} := F$
Do
$G := G_{next}$
$\forall \{g_i, g_j\} \subseteq G \times G$ with $i \neq j$:
$s := SPoly(g_i, g_j)$
$s := Reduce(s, G)$
if $(s \neq 0)$, then
$G_{next} := G_{next} \cup \{s\}$
Until $(G = G_{next})$
Return G

Where $Reduce(s, G)$ is an implementation of the multivariate division algorithm of a polynomial s by a set of polynomials G.

Now, there are two important questions coming up:

- Does this algorithm terminate?
- Does it really calculate a Gröbner Basis of I?

Proof. The first question is answered by Dickson's Lemma, respectively the Ascending Chain Condition (Theorem 3.4). We associate the current "state" of the algorithm with the set G, where $G_k =$ "G in the k-th loop". Consider the following ascending chain of monomial ideals:

$$\langle LT(G_1)\rangle \subset \langle LT(G_2)\rangle \subset \langle LT(G_3)\rangle \subset ...$$

According to the ACC, this chain will eventually stagnate and the algorithm will hereby terminate, because $G_i = G_{i+1}$ is exactly the termination criterion.

So the only thing left to state is, that $G_k \neq G_{k+1} \Rightarrow \langle LT(G_k)\rangle \subsetneq \langle LT(G_{k+1})\rangle$. $G_k \neq G_{k+1}$ implies, that there was at least one s, for which $s = Reduce(s, G_k) \neq 0$. This implies, that s was the remainder of the multivariate division algorithm. From this, it follows, that $LT(g) \nmid LT(s)\, \forall g \in G$. This means, that $LT(s) \in \langle LT(G_k \cup \{s\})\rangle$, but $LT(s) \notin \langle LT(G_k)\rangle$, which then shows us that $\langle LT(G_k)\rangle \subsetneq \langle LT(G_{k+1})\rangle$.

The second question is answered by the Buchberger criterion (Theorem 3.8). Since $G_i = G_{i+1}$ implies, that all S-Polynomials of the current set of generators G_i are being reduced to zero (because G_{i+1} would be the union of G_i and some s otherwise), we can apply the Buchberger criterion, which states, that G_i is really a Gröbner Basis of I. □

Obviously, one does a lot of unnecessary work in the algorithm provided above. If, for example, the S-Polynomial of f_1 and f_2 can not be reduced to 0, then we are trying this over and over again without adding something to G_{next}. In real implementations, which are equivalent in terms of the calculated result, these unnecessary computations are not done. One memorizes the pairs which have been calculated with a set. In every loop, one selects pairs out of this set, computes the S-Polynomial, reduces it and adds all new possible combinations of this freshly computed S-Polynomial with all $f \in G$. This assures, that no pair is being calculated twice.

3.5 The Faugère F_4 algorithm

The most time consuming step in the computation of Gröbner Bases with the classical Buchberger algorithm is the reduction modulo G. In 1999, Faugère proposed yet another algorithm [Fa02] for computing Gröbner Bases. We need the following definitions:

Definition 3.9. For a pair of polynomials $p := (g_i, g_j)$, we define $Left(p) = (t, g_i)$, where $LCM(LM(g_i), LM(g_j)) = t \cdot LM(g_i)$. Accordingly, we define $Right(p) = (t', g_j)$, where $LCM(LM(g_i), LM(g_j)) = t' \cdot LM(g_j)$.

Definition 3.10. For a set of pairs $S := \{p_1, ..., p_t\}$, we set $Left(S) := \{Left(p_i) \mid p_i \in S\}$ and $Right(S) := \{Right(p_i) \mid p_i \in S\}$.

Left and Right can be seen as the preparation for the S-Polynomial calculation.

Definition 3.11. Set $G := \{g_1, ..., g_s\}$, a set of polynomials. We define $minDeg := \min\{deg(LCM(LM(g_i), LM(g_j))) \mid g_i, g_j \in G, i \neq j)\}$. Furthermore, for a set of pairs $P := \{p_1, ..., p_t\}$ we specify the "Select"-function as follows:

$$\text{Select}(P) = \{(g_i, g_j) \mid (g_i, g_j) \in P, deg(LCM(LM(g_i), LM(g_j))) = minDeg\}.$$

This set contains the pairs, which have a "minimal" extension in terms of S-Polynomials in order to neutralize the leading term. Other selection strategies are possible, Faugère proposed this one.

The overall algorithm is not as interesting as the definition of Reduce(), the heart of F_4. It looks like the following:

ALGORITHM FAUGÈRE F_4

INPUT: A set $F := \{f_1, ..., f_t\}$ which defines the ideal $I = \langle f_1, ..., f_t \rangle$.
OUTPUT: $G = \{g_1, ..., g_{t'}\}$, which is a Gröbner Basis of I.

```
G := F
Pairs := {(g_i, g_j) | g_i ∈ G, g_j ∈ G, i ≠ j}
while Pairs ≠ ∅
        /* Select() was defined in definition 3.11:
         * Select and remove them
         */
        selectedPairs := Select(Pairs)
        Pairs := Pairs \ selectedPairs
        /* Left() and Right() as defined in 3.9 */
        SPolys := Left(selectedPairs) ∪ Right(selectedPairs)
        F+ := Reduce(SPolys, G)
        for all s ∈ F+ do
                G := G ∪ {s}
```

Pairs := Pairs $\cup\{(s, g) \mid g \in G\}$
end for
end while
Return G

Where Reduce(), the more important function is defined as follows:

FUNCTION REDUCE()

INPUT: A set of pairs, each containing of a monomial and a polynomial, called $SPolys$ and a set of polynomials G, representing the current state of the basis computation.
OUTPUT: A set F^+, which contains all the reduced forms for all $s \in SPolys$ modulo G.

$F := \{t \cdot g \mid (t, g) \in SPolys\}$.
/* insert all the reducers from G into the set */
Done := LT(F)
while Done $\neq T(F)$
pick $t \in T(F) \setminus$ Done
Done := Done $\cup\{t\}$
if $LM(g_i) \mid t \iff t = mLM(g_i)$ for some $g_i \in G, m \in k_x$
F := F $\cup \{m \cdot g_i\}$
end while

$\tilde{F}$:= gaussian elimination on F

/* only those f's which are really reduced may pass */
$F^+ := \{f \in \tilde{F} \mid LM(f) \notin LM(F)\}$
Return F^+

Doing a gaussian elimination on a set means to transform this set of polynomials into a matrix representation. The columns represent $T(F)$, and the entries represent the coefficients of these terms (see example below).

Again, the same questions arise:

- Does F_4 terminate?
- Does it really compute a Gröbner Basis?

Proof. The proofs for termination and correctness of the Faugère F_4 algorithm can be found in [Fa02]. □

Apart from the question of correctness and termination, one wonders, whether and why the F_4 algorithm is faster than the classical Buchberger algorithm. The improvement in terms of efficiency is important, in specific for attacking HFE with the algorithm provided above. This improvement exists and is mainly caused by the special reduction technique. The difference should be cleared up with an example:

Example 3.2. Consider an ideal $I = \langle G \rangle$, defined by the following generators (using lexicographic order): $G := \{g_1, g_2, g_3, g_4, r\}$, where $g_1 = xy^2 + 1,\ g_2 = x^2y + 1,\ g_3 = yz + 1,\ g_4 = xz + 1,\ r = x + z$. The classical Buchberger algorithm would now do the following:

(i) Take a pair, say (g_1, g_2), for example, and calculate the S-Polynomial $s_{12} = xg_1 - yg_2 = x - y$.

(ii) Reduce it by G, which means dividing it by r, because $LT(r) \mid LT(s_{12})$ (reduction (1)).

(iii) Take a pair, say (g_3, g_4), for example, and calculate the S-Polynomial $s_{34} = -xg_3 + yg_4 = -x + y$.

(iv) Reduce it by G, which means dividing it by r, because $LT(r) \mid LT(s_{34})$ (reduction (2), which basically does nothing else but reduction (1)).

It is obvious, that reduction (2) is redundant, because $s_{34} = -1 \cdot s_{12}$. Although the F_4 algorithm given above would not select (g_1, g_2) and (g_3, g_4), because of the specified selection tactic, let us assume, it did. Then, the matrix F (on which a gaussian elimination is going to applied) would look like the following:

nr.	x^2y^2	xyz	x	y	z	corrosponds to
①	1	0	1	0	0	$x^2y^2 + x$
②	-1	0	0	-1	0	$-x^2y^2 - y$
③	0	-1	-1	0	0	$-xyz - x$
④	0	1	0	1	0	$xyz + y$
⑤	0	0	1	0	1	$x + z$

The first steps in the gaussian elimination would probably be adding ① to ② and ③ to ④, which means, that $②_{new} :=$ ②+ ① and $④_{new} :=$ ④+ ③, which then leads to the following matrix (resorted):

nr.	x^2y^2	xyz	x	y	z	corrosponds to
①	1	0	1	0	0	$x^2y^2 + x$
③	0	-1	-1	0	0	$-xyz - x$
$②_{new}$	0	0	1	-1	0	$x + y = s_{12}$
$④_{new}$	0	0	-1	1	0	$-x + y = s_{34}$
⑤	0	0	1	0	1	$x + z$

In the next step, ④$_{new}$ would be neutralized, which then also causes reduction (2) - which was redundant - never to happen. So F_4 is able to detect and avoid redundant reductions by reducing all parts of the S-Polynomials together in one big matrix. Apart from this, special sparse linear algebra solver implementations make the calculation of the gaussian reduction even faster.

Of course, ②$_{new}$ still would have to be reduced modulo ⑤, but as the complexity of the system increases, the speed up caused by the avoidance of redundant reductions and caused by sparse linear algebra implementations becomes significant.

In current literature, one often finds terms like "simultaneously" or "reduction in parallel" when it comes to the reduction technique of F_4. Obviously, this is not fully true, since the reduction as it is done sequentially. The reductions simply coordinate each other in order to avoid redundant steps as shown above.

4 How to save HFE

We will now discuss how to save HFE. Therefor, we will present two modifications. The first one will be a rather simple adaption of the characteristic of the base field k. Furthermore, we will argue why all Gröbner Bases-algorithms will fail when we change the base field. The second modification is called "Internal Perturbed HFE", in short: IPHFE [DSc02]. This adaption is somehow more sophisticated. Its basic tactic is to perturb the HFE system by the insertion of a map from K to K. The perturbation concept will be introduced in the respective section in detail.

In order to support these claims, we will present the results of extensive computer experiments.

5 Saving HFE by adapting the characteristic of k

As mentioned above: one very easy way to make HFE secure again is to adapt the characteristic of k. Changing the cardinality of k is not enought in every case, because for every $h \in \mathbb{N}$, a ground field like $k = GF(2^h)$ can easily been transformed into a case where $k = GF(2)$ and K is chosen with respect to h. In the next subsection, we will explain why the adaption takes place and why HFE can consequently defeat a Gröbner Bases-attack.

5.1 The role of the Field Equations

Please reconsider that the variety of a Gröbner Basis normally contains solutions in K. We will now analyze how to tell the algorithm to search for solutions over K (and not over $AC(K)$) only. Therfor, we will use the so-called field equations. Field equations are of the form

$$x_i^q \equiv x_i.$$

They contain the essential information that the x_i's can not take any arbitrary value but rather only values in k. From the theorems of Lagrange, we know that for every Group G and any $g \in G$, $g^{|G|} = g$. Consequently, $x_i^{|G|} = x_i^q$, so $x_i^q - x_i = 0$. The Gröbner Basis-algorithm makes use of them in the way that it reduces S-Polynomials (or polynomials in general) by these field equations. We now have two possibilities: append them to the ideal $\langle f_1 - c_1, ..., f_n - c_n \rangle$ (where the f_i's are the public key polynomials of HFE and c_i is one component of the ciphertext), or leave them out.

5.1.1 Inserting the field equations

From basic algebra, we know that given $n, d \in \mathbb{N}$, we can form $\binom{n+d}{d}$ different monomials in n variables of a degree smaller or equal to d. When chosing $k = GF(11)$ and $n = 32$, the number of monomials already exceeds 2^{32}. This implies that if the algorithm proceeds until he can actually make use of the field equations, he will run out of memory, because it will take 4 GB of memory just to store one single polynomial in a classical array-coefficient-based version. Additionally, the algorithm will use an extreme amount of

time, because as the polynomials are getting bigger, it takes much more time to do a reduction modulo the current basis which is an essential step of all Gröbner Basis-Algorithms.

5.1.2 Leaving out the field equations

When the ideal does not contain the field equations, the following happens: As we have seen in theorem 3.2, the Gröbner Basis-algorithm considers every solution for the polynomial system, but so far, we ave only considered solutions in K, k^n respectively. In terms of breaking HFE, we are interested in the solutions which are in K, but whithout the field equations, the Gröbner Basis-algorithm does not know about a field, so the algorithm has to "preserve" all solutions from the *algebraic closure of* K in the Gröbner Basis.

Example. Let us consider the polynomial $\tilde{F}(X) = X^2 + X + 1$ over $K = k^n, n = 5, k = GF(11)$. Note that $\tilde{F}$ is not an HFE private key, it was chosen to keep things simple. In K, this polynomial clearly has no solution, because it is irreducible. But in the algebraic closure, it has a solution (called "solution at infinity"). When doing a level 2 algebraic extension of K this results in the field $\bar{K} = GF((11^n)^2)$. As extension variable, we use w and as the irreducible polynomial we use $g(w) = w^2 + w + 1$. Now we see that w is a solution to $\tilde{F}(X) = 0$, because $w^2 + w + 1 = 0$ by definition of the irreducible polynomial. Please note that the choice of $g(w)$ is not important since fields of the same size are isomorphic. The solutions just look slightly different when choosing another irreducible extension polynomial.

When calculating the set of solutions over k^n, the Gröbner Basis, the algorithm produces, must include this solution w. Indeed, when we calculate the Gröbner Basis using singular, we obtain 25 polynomials. After setting $x_2 = ... = x_5 = 0$ — because the solution at infinity looks like $w * (1 + 0t + 0t^2 + ... + 0t^4)$ — polynomial number 5 looks like the following: $g_5(x_1, ..., x_5) = x_1^2 + x_1 + 1$ which then "embodies" the solution at infinity. A singular script which demonstrates this can be found in [Wer07c].

Observation. Statistically speaking, we have a high propability that the HFE system has exactly as many solutions as the degree of the polynomial $\tilde{F}(X) - C$ (which has to be factored in order to find the plaintext).

This comes, because it is very likely that a random polynomial in K is irreducible (see also section 5.2.1). Consequently, we have a high chance that the factorization of $\tilde{F}(X) - C$ is square free which then means, that it has indeed the same amount of solutions (in the algebraic closure of K) as its degree. This is guaranteed by the so-called "fundamental theorem of algebra" which can be found in every ordinary math book. When setting $k = GF(11)$, $r_1 = r_2 = 2$, the private key polynomial will have a degree of $11^2 + 11^2 = 242$.

Let $\#s$ denote the number of solutions to $\tilde{F}(X) - C = 0$ which is the same as the

degree of the private key polynomial $\tilde{F}(X)$. So we know that the number of solutions to the Gröbner Basis has to be $\#s$. This means that either the univariate polynomial has to have degree $\#s$ or the product of the leading degrees of the polynomials in the Gröbner Basis has to be $\#s$, because the leading degree determines the number of solutions. If such a product has to be 242, then at least one polynomial has to have degree 11, because of the prime factors of 242 being $242 = 11 * 11 * 2$. We see that the polynomials are not storable again when leaving out the field equations, because a polynomial in a realistic amount of variables and of degree 11 exceeds 4GB (see section 5.1.1).

Of course, it is easy to construct counter-examples for the observation, but the chances for having a private key polynomial and a ciphertext such that $\#s \neq deg(\tilde{F}(X) - C)$ for example are negligible.

5.2 Computer experiments

The experiments are divided in two parts. The first part will consider experiments on the number of solutions of an equation

$$\tilde{F}(X) - Y = 0,$$

where $Y = \phi^{-1}(c)$ and c is some ciphertext. The second part will consist of measurements of time and memory usage, it took F_4 to break certain HFE systems.

5.2.1 Experiment on the number of solutions

In order to verify the claim that (statistically speaking) the number of solutions of $\tilde{F}(X) - Y = 0$ in terms of X is equal to the degree of $\tilde{F}(X) - Y$, we ran an experiment: First, we set up an HFE system and its hidden field polynomial $\tilde{F}(X)$. We then encrypted a random plaintext X' by finding $Y = \tilde{F}(X')$. Afterwards the program calculated $\tilde{F}(X) - Y$ and factored this polynomial. We did 800 test cases for $k = GF(11)$, 400 using n = 17 and 400 using n = 19. Not a single factorization contained a factor with a multiplicity higher than one, which means that the number of solutions in all 800 tests was exactly the degree of $\tilde{F}(X) - Y$.

5.2.2 Experiment on time and memory usage

We first set up an HFE system and generated the public key equations. Afterwards we encrypted some random plaintext and then used MAGMA [MaSy] to find the Gröbner Basis of the ideal $I = \langle f_1(x_1, ..., x_n) - c_1, ..., f_n(x_1, ..., x_n) - c_n \rangle$, where c_i denotes the i-th component of the ciphertext vector and $f_i(x_1, ..., x_n)$ is the i-th public key polynomial over k. Our program then found all solutions and verified the variety indeed included the original plaintext.

The MAGMA command used to calculate the Gröbner Basis was `Groebner(I : HFE)`, where I denotes the ideal as given above. The parameter `HFE` causes MAGMA to disregard all polynomials with the degree bigger than 4. According to the MAGMA documentation [MaG], this only has an influence if the ground field is $GF(2)$. This was

confirmed by experiments, so the parameter could have been left out. It was inserted, because the actual influence of the parameter is not known, since the implementation of F_4 is closed-source. All experiments were done without putting the field equations into the ideal I, as this slows things down (see Fig. 6). Tables below show the running time and the required memory of each n.

In both figures we take n as the X-coordinate and show the running time (on the left, in seconds) and the required memory (on the right, in MB) as the logarithmic Y-coordinate. It clearly shows the exponential growing tendency with increasing n. Please note that all values below 1 second/MB have been rounded up to 1 for improving the visibility of the graphs. A more detailed overview over timings and memory usage can be found in the appendix. Much more theoretical and experimental work is still needed to fully understand the whole behavior.

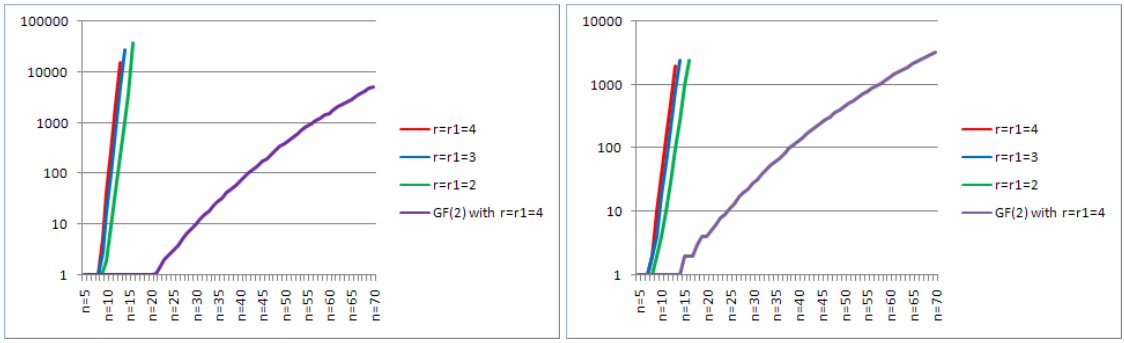


Figure 1: Timings and memory usage for HFE systems over *GF(3)*

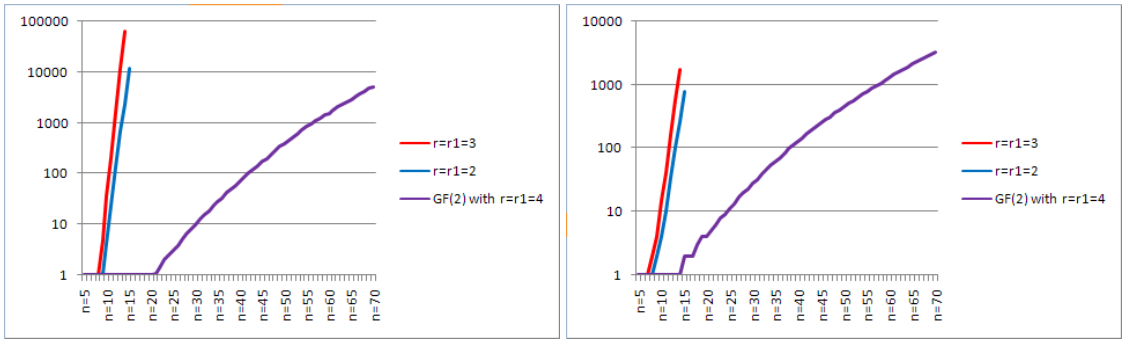


Figure 2: Timings and memory usage for HFE systems over *GF(5)*

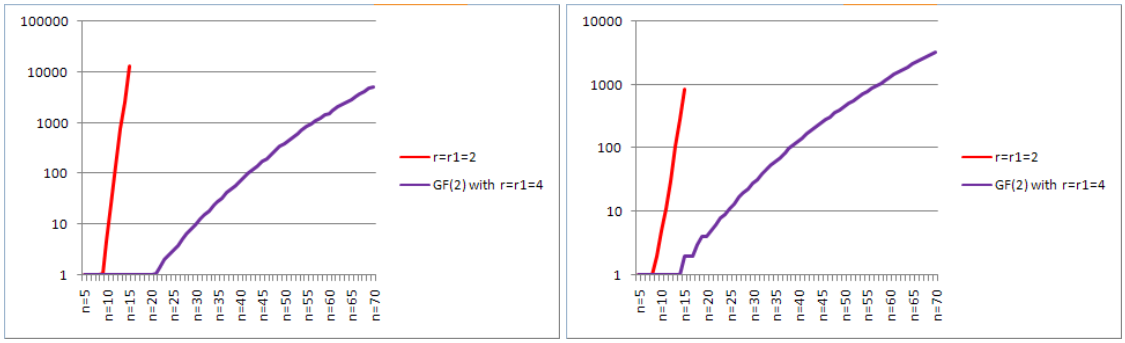


Figure 3: Timings and memory usage for HFE systems over *GF(7)*

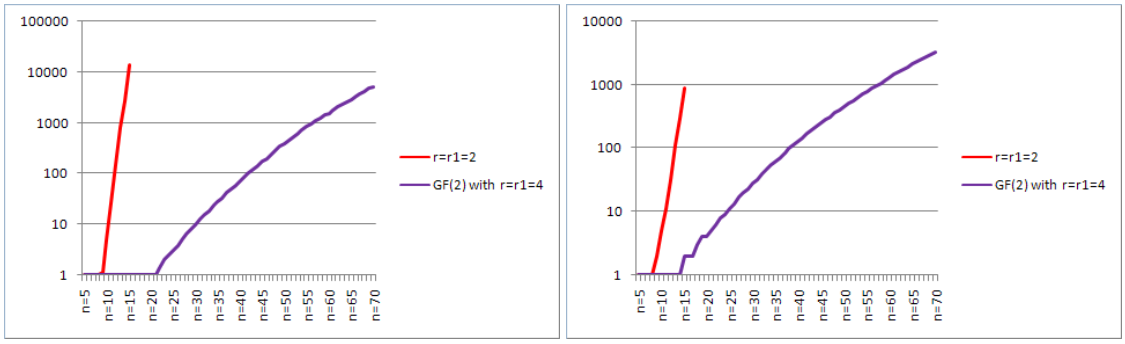


Figure 4: Timings and memory usage for HFE systems over *GF(11)*

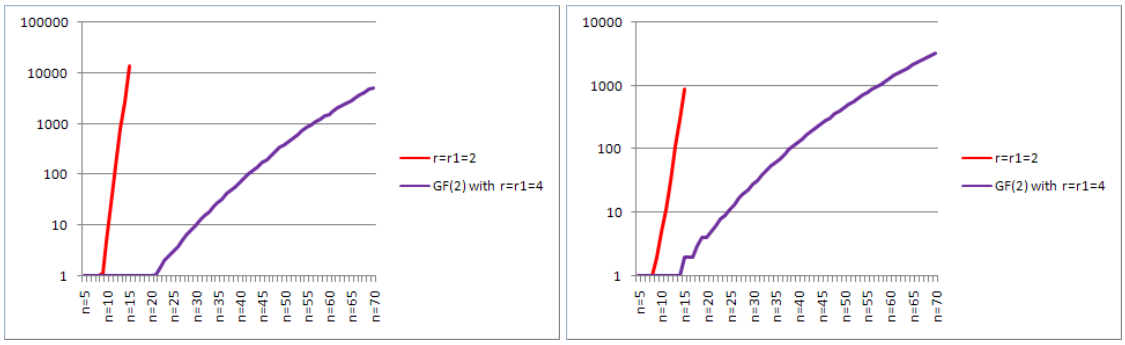


Figure 5: Timings and memory usage for HFE systems over *GF(13)*

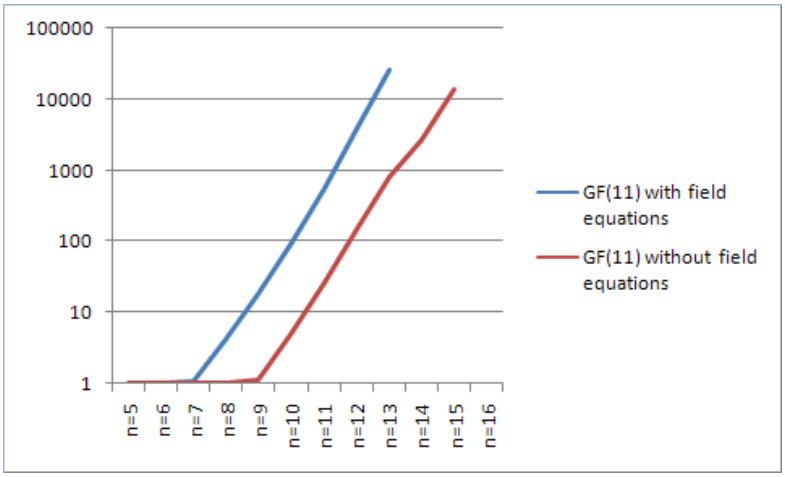


Figure 6: Timings for HFE over *GF(11)* with and without field equations "$x_i^{11} \equiv x_i$"

This experiment fleshes out the claim that changing the characteristic of the ground field k indeed has an enormous influence on the efficiency of the Gröbner Basis-attack. We conclude that we are able to build an HFE system which makes use of $k = GF(11)$ and is able to defeat the F_4-attack and all other known attacks.

Remark. After all, the F_4 implementation of the MAGMA algebra system gave us even bigger degrees for the univariate polynomial. It is not completely clear yet what is going on exactly, because the number of solutions would exceed the *maximum* amount of possible solutions. It follows that either there are redundancies or the implementation – which is closed source – was not done 100% accordingly to the description of the F_4 algorithm as provided by Faugère.

6 Internal perturbation of HFE (IPHFE)

In this section we will discuss another variation of HFE. The internal perturbed version of HFE was suggested by Ding and Schmidt [DSc02]. A new k-linear map $Z : K \to K$ and a parameter IP_r, namely the dimension of the image space of Z in K, is introduced. The private key polynomial $\tilde{F}(X)$ is extended in the following way:

$$\begin{aligned} \tilde{F}(X) \quad = \quad & \sum_{i=0}^{r_2} \sum_{j=0}^{i-1} \alpha_{ij} X^{q^i+q^j} + \sum_{i=0}^{r_1} \beta_i X^{q^i} + \gamma + \\ & \sum_{i=0}^{r_1} \sum_{j=0}^{n-1} \lambda_{ij} X^{q^i} Z^{q^j} + \\ & \sum_{i=0}^{n-1} \sum_{j=0}^{i} \mu_{ij} Z^{q^i+q^j} + \sum_{i=0}^{n-1} \theta_i Z^{q^i} \end{aligned} \tag{6.1}$$

where the coefficients, respectively the constant $\alpha_{ij}, \beta_i, \gamma, \lambda_{ij}, \mu_{ij}, \theta_i$ are randomly chosen elements in $(k[t]/g(t))$ and $Z = Z(X)$ is a k-linear map with dimension IP_r. The additional sums are called the "internal perturbation". Two questions arise: How to find such a map Z and how to decrypt?

6.1 Finding a k-linear map of dimension IP_r

One way to do this is the following: first, a few new variables called $z_1, ..., z_{\mathrm{IP}_r}$ are introduced. Now $Z(X)$ is defined as

$$Z : X \to z_1 + z_2 t + ... + z_{\mathrm{IP}_r} t^{\mathrm{IP}_r} + 0t^{\mathrm{IP}_r+1} + ... + 0t^{n-1} \tag{6.2}$$

From another point of view, one could also create the coefficients of the internal perturbation directly by chosing linear respectivly quadratic combinations of the z_i's but this does not matter, because they get mixed up due to the construction of the internal perturbation anyhow. After the z_i's have been mixed up by Z^{q^i}, some random linear combinations $c_i(x_1, ..., x_n) = \sum_{j=1}^{n} b_{ij}x_j + b_{i(j+1)}$ where all $b_{ij} \in k$ are chosen.

For creating the public key, the z_i's are substituted by these c_i's. The c_i's can be seen as basis vectors for an IP_r-dimensional space. Since we chose exactly IP_r of them, we really obtain an IP_r-dimensional space. For being absolutely sure, one should chose linearly independent c_i's but since the chance for the c_i's being linearly independent is overwhelmingly high when n is big enought, we simply chose them randomly.

In the private key, the z_i's do not get substituted, because otherwise, the private key holder would loose the information where the classic key ends and where the perturbation starts (this is basically one part of the trapdoor in this case!). Instead, the z_i's remain in the private key as abstract symbols. Additionally, the assignment of the z_i's to the corresponding c_i's is preserved for validating the correct candidate in the image space of Z (see below).

6.2 Encryption and decryption

(6.1) again results in n at most quadratic polynomials $f_1(x_1, ..., x_n), ..., f_n(x_1, ..., x_n)$ over k in the variables $x_1, ..., x_n$ and not in the z_i, because the z_i's are substituted by linear combinations of the x_i's. So encryption is done as usual by evaluating these public key polynomials for a given plaintext.

Decryption is somewhat more difficult, because we cannot simply factorize $\tilde{F}(X)$ anymore, since the occurence of Z causes it not to be univariate anymore. The most intuitive way is to imagine the z_i to be external variables (see example below). This is the reason why IP_r must be chosen small (usually ≤ 4), because in order to factorize we search through the whole image space of $Z(X)$ in K. Obviously the size of the image space is q^{IP_r}, because for every z_i we have q choices. For every Z' in the image space of $Z(X)$, which is nothing else than a one single guess for the z_i's, we now substitute the chosen one into $\tilde{F}(X)$ and call it $\tilde{F}_{Z'}(X)$. So for a given ciphertext c, we compute $C = \phi^{-1} \circ L_1^{-1}(c)$ and then factorize $\tilde{F}_{Z'}(X) - C$. If this does not produce any solution we are moving on to the next Z', else we check whether the solution we found for the x_i's really match

the z_i's guessed. If this is the case, then we add this solution to the list of solutions to check against the additional information and proceed with the next solution or the next Z'. Therefor after a finite number of steps, we will hit the correct guess for the vaules of the z_i's and therefor recover the original plaintext.

6.3 Toy example for IPHFE

We continue example 2.1, where the classical private key polynomial was chosen to be $F_{\text{classic}}(X) = (t^2+t+1)X^6 + (t^2+t+1)X^5 + (t)X^3 + X^2 + (t^2+t+1)X + (t)$ for $q = 2, r_1 = 1, r_2 = 2$. We choose IP_r to be 2 for this example, so we will deal with two additional z-variables z_1 and z_2. For the internal perturbation, we select

$$IP(X) = t * X^2Z^2 + (t+1) * Z^5$$

so we set $\lambda_{11} = t$, $\mu_{20} = (t+1)$ and all others to 0. According to (6.2), we now define Z to be $(z_1 + t * z_2)$, so we substitute Z by $(z_1 + t * z_2)$ in $IP(X)$ which results in

$$IP_z(X) = tX^2z_1 + (t+1)X^2z_2 + (t^2+t+1)z_1z_2 + (t+1)z_1 + tz_2$$

which then forms the final private key $\tilde{F}(X) = F_{\text{classic}}(X) + IP_z(X)$ (consisting of X and abstract symbols z_1, z_2). We chose

$$\begin{aligned} c_1(x_1, x_2, x_3) &= x_1 + x_3 \\ c_2(x_1, x_2, x_3) &= x_2 + 1 \end{aligned}$$

to be the concrete liner combinations for z_1 and z_2.

By evaluating $\phi \circ \tilde{F}(X) \circ \phi^{-1}(x_1, x_2, x_3)$ and substituting z_i by c_i in the result, we obtain the public key which reads

$$\begin{aligned} f_1(x_1, x_2, x_3) &= x_2x_3 + x_1 + x_3 \\ f_2(x_1, x_2, x_3) &= x_2x_3 \\ f_3(x_1, x_2, x_3) &= x_1x_2 + x_1x_3 + x_2x_3 + x_3 \end{aligned}$$

and encrypt the plaintext $(p_1, p_2, p_3) = (1, 1, 0)$ which produces the ciphertext $c = (1, 0, 1)$, so $C = (1 * t^0 + 0 * t + 1 * t^2) = t^2 + 1$.

Now we decrypt by guessing our first Z'_1 to be $z_1 = z_2 = 0$. Consequently, $\tilde{F}_{Z'_1} = \tilde{F}_{\text{classic}}$. The factorization of $\tilde{F}_{Z'_1} - C$ tells us that the solutions are t^2 with $phi(t^2) = (0, 0, 1)$ and $t^2 + t$ with $\phi(t^2 + t) = (0, 1, 1)$. We check whether $c_1(0, 0, 1) = z_1 = 0$ which is not the case. We check whether $c_1(0, 1, 1) = z_1 = 0$ which is also not the case, so these solutions can be discarded, because the values for z_1 and z_2 were obviously not 0.

We proceed in this way and indeed, for Z'_3 we guess $z_1 = 1$ and $z_2 = 0$ which really was the case when encrypting the plaintext. Consequently, $\tilde{F}_{Z'_3} = \tilde{F}_{\text{classic}} + (t)X^2 + (t+1)$. As the only solution for $\tilde{F}_{Z'_3} - C = 0$, we obtain $t + 1$ with $phi(t + 1) = (1, 1, 0)$, the original plaintext.

A Singular script which demonstrates the usage of these constructions can be found in [Wer07b].

6.4 Computer experiments on IPHFE

6.5 The algebraic attack revisited

A lot of structural analysis has been done for IPHFE. Much effort has been put on separating the internal perturbation from the original private key in order to make HFE vulnerable against the usual algebraic attacks again. We have tried a different approach. Instead of ascribing IPHFE back to a usual HFE system, we applied the algebraic attacks directly on the internally perturbed version of HFE. In this section, we will present the results.

For doing the experiments on the IPHFE cryptosystems, we used the same infrastructure as decsribed in section 5.2.2. The experiments here differ by the usage of the `HFE` parameter which causes MAGMA to disregard all polynomials of a degree higher than four. In the first test, we used the `HFE` parameter. We found out that even for small values like $r1 = r2 = 4$, $\mathrm{IP}_r = 4$, the plaintext was not in the variety of the computed basis anymore for $n > 17$. As the ideal has been reduced to $\langle 1 \rangle$, this is clearly a bug in the MAGMA-implementation of F_4, because the system has at least one solution, namely the plaintext. We conjecture that the internal perturbation forces us not to throw away polynomials of a degree higher than four for realistic n-values, but this cannot be proven because of the bug.

For the second run, we did not use the `HFE` parameter anymore. We used an IPHFE system with $k = GF(2)$, $\mathrm{IP}_r = 2, 3, 4$ and $r_1 = r_2 = 3, 4$. The timings do not seem to be exponential and the memory usage looks very odd (below 3 MB for n in the range of $1, ..., 17$ and a sudden jump to >4GB for bigger n-values). The X-axis shows time and memory and the Y-axis shows the parameter n (number of variables). In the timing diagram (on the left), the time in seconds is logarithmically scaled. Additionally, the green line shows the timings for a classical HFE system over $GF(2)$.

6.5.1 Results for computing Gröbner Bases with the HFE parameter

Please note that the timings and memory for $\mathrm{IP}_r = 4$ could not be verified due to a bug in the MAGMA implementation of F_4. Although MAGMA did not calculate the correct ideal we observed time and memory usages which were consistent with the timing when using $\mathrm{IP}_r = 2, 3$, so we show them (as a dotted line) anyhow.

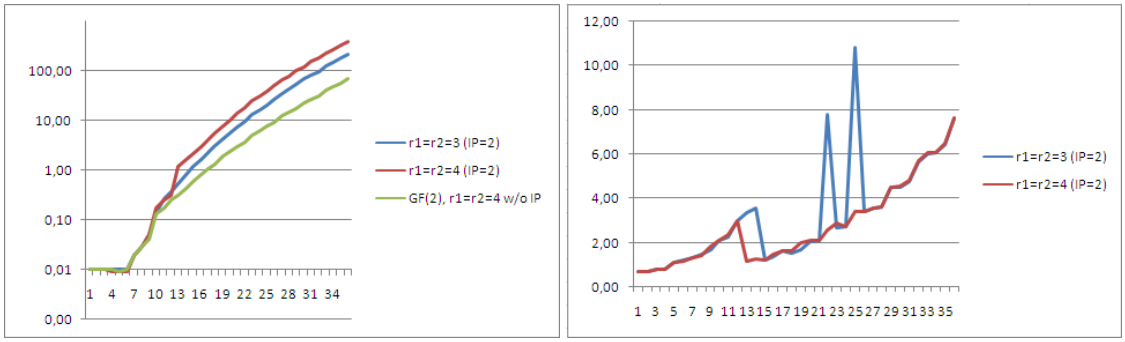


Figure 7: Timings and memory usage for IPHFE systems with $\mathrm{IP}_r = 2$

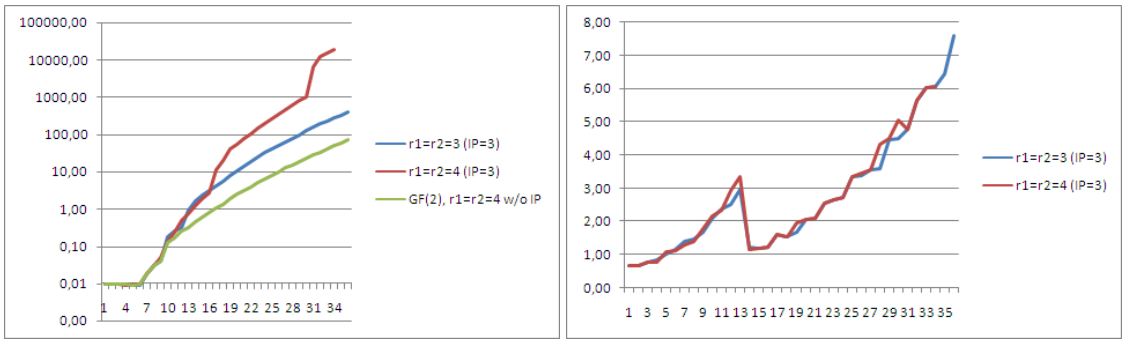


Figure 8: Timings and memory usage for IPHFE systems with $\mathrm{IP}_r = 3$

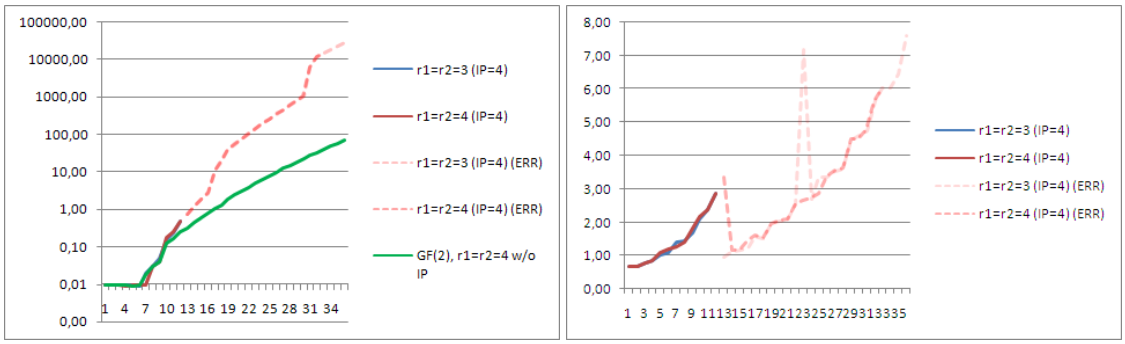


Figure 9: Timings and memory usage for IPHFE systems with $\mathrm{IP}_r = 4$

6.5.2 Results for computing Gröbner Bases without using the HFE parameter

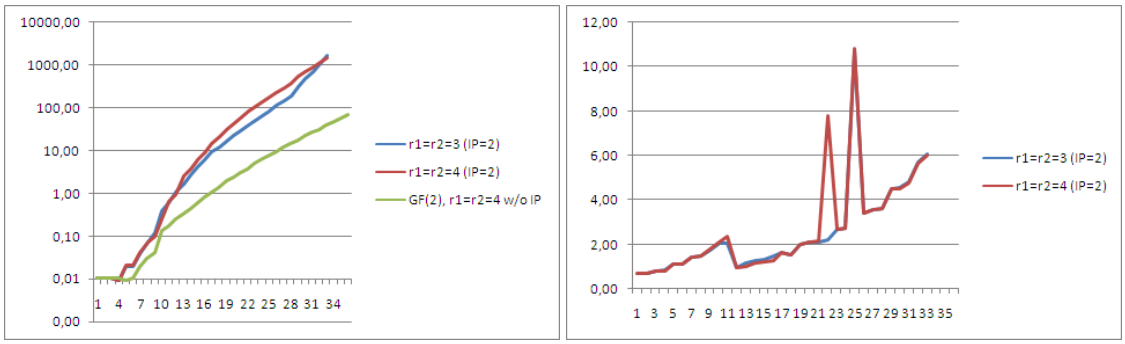


Figure 10: Timings and memory usage for IPHFE systems with $\mathrm{IP}_r = 2$

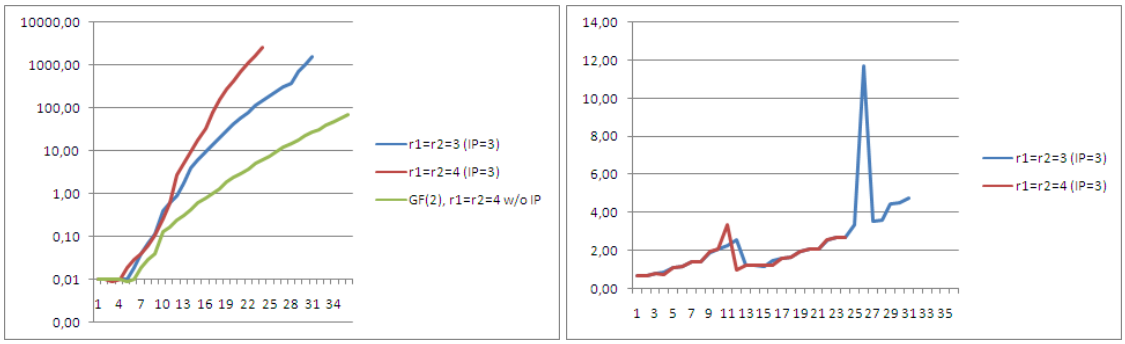


Figure 11: Timings and memory usage for IPHFE systems with $\mathrm{IP}_r = 3$

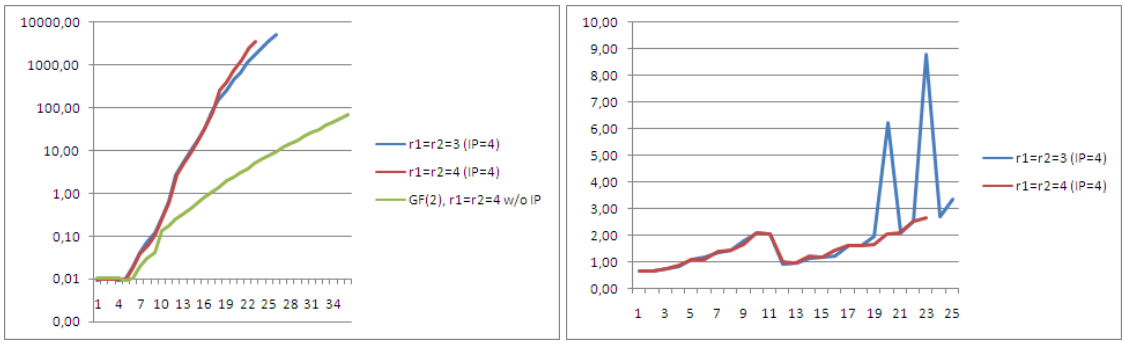


Figure 12: Timings and memory usage for IPHFE systems with $\mathrm{IP}_r = 4$

All in all, we can not say that the internal perturbation – without any further modification of the original scheme – is a good method for making HFE secure again.

7 MAGMA bugs

During the testing phase of the thesis, we discovered two bugs in the MAGMA engine for computing Gröbner Bases. The first bug occurs when leaving out the field equation for a small number of variables (like $n = 3$ or $n = 5$). The amount of solutions which can be extracted from the calculated Gröbner Basis is too low. This means, that the Gröbner Basis-algorithm drops some of the solutions at infinity which contradicts theorem 3.2. This means that the algorithm was not implmented according to the description of Faugère.

An implementation of the environment causing this bug can be found at

`http://happy-werner.de/BSc/magmaBug_degree_of_gb_too_low.zip`.

The second bug occured when using IPHFE. Please note that to the Gröbner Basis-algorithm, this is just a usual system of polynomials like any other. For a fixed plaintext, the F_4-implementation computed a Gröbner Basis being $\langle 1 \rangle$. Consequently the system has no solution but we know that there exists at least one solution, namely the plaintext.

An implementation of the environment causing this bug can be found at

`http://happy-werner.de/BSc/magmaBug_IPHFE_no_solution.zip`.

A Time- and memory-tables

A.1 Tables for $\mathbb{F}_3$

TIMINGS	$r_i = 2$	$r_i = 3$	$r_i = 4$
$n = 5$	0,01	0,01	0,01
$n = 6$	0,01	0,03	0,03
$n = 7$	0,05	0,13	0,16
$n = 8$	0,15	0,56	1,01
$n = 9$	0,53	2,72	5,44
$n = 10$	1,90	17,44	41,77
$n = 11$	9,30	96,78	283,88
$n = 12$	47,12	714,06	2264,28
$n = 13$	223,88	5059,40	15477,45
$n = 14$	717,17	26654,78	>84000
$n = 15$	3623,57	>84000	
$n = 16$	37145,93		
$n = 17$	>84000		

MEMORY USAGE	$r_i = 2$	$r_i = 3$	$r_i = 4$
$n = 5$	1	1	1
$n = 6$	1	1	1
$n = 7$	1	1	1
$n = 8$	1	2	2
$n = 9$	2	4	10
$n = 10$	4	18	37
$n = 11$	10	57	142
$n = 12$	29	214	508
$n = 13$	101	807	1948
$n = 14$	267	2441	
$n = 15$	950		
$n = 16$	2435		

A.2 Tables for $\mathbb{F}_5$

TIMINGS	$r_i = 2$	$r_i = 3$
$n = 5$	0,01	0,01
$n = 6$	0,02	0,04
$n = 7$	0,04	0,16
$n = 8$	0,23	0,95
$n = 9$	1,03	4,79
$n = 10$	4,26	35,90
$n = 11$	23,89	210,02
$n = 12$	136,00	1589,49
$n = 13$	706,81	11697,51
$n = 14$	2359,84	64142,64
$n = 15$	12085,06	>84000
$n = 16$	>84000	

MEMORY USAGE	$r_i = 2$	$r_i = 3$
$n = 5$	1	1
$n = 6$	1	1
$n = 7$	1	1
$n = 8$	1	2
$n = 9$	2	4
$n = 10$	4	15
$n = 11$	10	40
$n = 12$	34	152
$n = 13$	102	575
$n = 14$	267	1734
$n = 15$	787	

A.3 Tables for $\mathbb{F}_7$

TIMINGS	$r_i = 2$
$n = 5$	0,01
$n = 6$	0,02
$n = 7$	0,05
$n = 8$	0,24
$n = 9$	1,07
$n = 10$	4,60
$n = 11$	25,23
$n = 12$	139,00
$n = 13$	763,45
$n = 14$	2505,47
$n = 15$	13006,56
$n = 16$	>84000

MEMORY USAGE	$r_i = 2$
$n = 5$	1
$n = 6$	1
$n = 7$	1
$n = 8$	1
$n = 9$	2
$n = 10$	5
$n = 11$	11
$n = 12$	30
$n = 13$	108
$n = 14$	283
$n = 15$	836

A.4 Tables for $\mathbb{F}_{11}$ and $\mathbb{F}_2$ for comparison

TIMINGS	**GF(11)**	**GF(2)**	TIMINGS (cont'd)	**GF(2)**
$n = 5$	0,01	0,01	$n = 41$	82,51
$n = 6$	0,02	0,01	$n = 42$	103,27
$n = 7$	0,07	0,01	$n = 43$	118,96
$n = 8$	0,25	0,01	$n = 44$	137,69
$n = 9$	1,13	0,01	$n = 45$	171,33
$n = 10$	4,74	0,01	$n = 46$	193,73
$n = 11$	25,87	0,02	$n = 47$	237,72
$n = 12$	147,03	0,03	$n = 48$	274,68
$n = 13$	799,96	0,04	$n = 49$	333,26
$n = 14$	2722,40	0,13	$n = 50$	376,39
$n = 15$	13744,27	0,17	$n = 51$	451,14
$n = 16$	>84600	0,25	$n = 52$	522,93
$n = 17$		0,32	$n = 53$	603,70
$n = 18$		0,44	$n = 54$	699,91
$n = 19$		0,61	$n = 55$	815,79
$n = 20$		0,81	$n = 56$	907,32
$n = 21$		1,05	$n = 57$	1084,66
$n = 22$		1,35	$n = 58$	1172,62
$n = 23$		1,94	$n = 59$	1385,72
$n = 24$		2,41	$n = 60$	1519,22
$n = 25$		3,03	$n = 61$	1764,68
$n = 26$		3,79	$n = 62$	2041,06
$n = 27$		5,12	$n = 63$	2242,23
$n = 28$		6,35	$n = 64$	2537,06
$n = 29$		7,70	$n = 65$	2811,87
$n = 30$		9,41	$n = 66$	3202,55
$n = 31$		12,48	$n = 67$	3531,15
$n = 32$		15,04	$n = 68$	4015,17
$n = 33$		17,93	$n = 69$	4636,33
$n = 34$		23,26	$n = 70$	4962,35
$n = 35$		27,33		
$n = 36$		31,98		
$n = 37$		41,27		
$n = 38$		49,08		
$n = 39$		57,39		
$n = 40$		70,98		

MEMORY USAGE	**GF(11)**	**GF(2)**	MEMORY (cont'd)	**GF(2)**
$n = 5$	0,76	0,67	$n = 41$	15,54
$n = 6$	0,76	0,67	$n = 42$	17,73
$n = 7$	0,95	0,67	$n = 43$	18,34
$n = 8$	1,03	0,65	$n = 44$	21,43
$n = 9$	1,06	0,72	$n = 45$	22,42
$n = 10$	1,47	0,71	$n = 46$	21,55
$n = 11$	2,88	0,74	$n = 47$	24,37
$n = 12$	5,89	0,86	$n = 48$	30,59
$n = 13$	14,23	0,93	$n = 49$	31,47
$n = 14$	34,15	1,18	$n = 50$	30,74
$n = 15$	105,76	1,35	$n = 51$	34,64
$n = 16$		1,24	$n = 52$	36,06
$n = 17$		2,55	$n = 53$	41,31
$n = 18$		2,98	$n = 54$	43,19
$n = 19$		1,72	$n = 55$	47,86
$n = 20$		1,98	$n = 56$	50,73
$n = 21$		2,21	$n = 57$	53,40
$n = 22$		2,38	$n = 58$	54,74
$n = 23$		2,52	$n = 59$	57,73
$n = 24$		3,07	$n = 60$	59,57
$n = 25$		3,45	$n = 61$	66,97
$n = 26$		3,62	$n = 62$	69,02
$n = 27$		4,21	$n = 63$	66,99
$n = 28$		4,91	$n = 64$	76,00
$n = 29$		4,78	$n = 65$	108,57
$n = 30$		5,98	$n = 66$	101,17
$n = 31$		5,74	$n = 67$	115,65
$n = 32$		7,21	$n = 68$	131,64
$n = 33$		7,78	$n = 69$	134,84
$n = 34$		8,92	$n = 70$	140,20
$n = 35$		9,40		
$n = 36$		10,83		
$n = 37$		11,42		
$n = 38$		13,00		
$n = 39$		13,76		
$n = 40$		16,20		

References

[FJo03] Jean-Charles Faugère , Antoine Joux: *Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems Using Gröbner Bases* (2003). In Crypto 2003, LNCS 2729, pp. 44-60, Springer.

[MI88] Tsutomu Matsumoto, Hideki Imai: *Public quadratic polynomial-tuples for efficient signature verification and message-encryption.* (1988) In EUROCRYPT 1988, volume 330 of LNCS, pages 419-545.

[Pat95] Jacques Patarin: *Cryptoanalysis of the Matsumoto and Imai public key scheme of Eurocrypt'88.* (1995) In D. Coppersmith, editor, *Advances in Cyptology - Crypto '95*, volume 963 of LNCD, pages 248-261.

[Be70] Elwyn Berlekamp *Factoring Polynomials over Large Finite Fields* (1972). Mathematics of Computation 24:713-735; Math. Rev. 48:1943. `http://math.berkeley.edu/ berlek/poly.html`

[MaSy] *MAGMA Algebra System* `http://magma.maths.usyd.edu.au/magma/`

[MaG] *The MAGMA system documentation* Section on Gröbner Bases: `http://magma.maths.usyd.edu.au/magma/htmlhelp/text1158.htm#11134`

[BB65] Bruno Buchberger: *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal* (1965), Mathematical Institute, University of Innsbruck, Austria. Dissertation.

[Buc96] Johannes Buchmann: *Algebra für Informatiker* (1996). Lecture notes.

[Buc04] Johannes Buchmann: *Einführung in die Kryptographie* (2004). Springer Verlag.

[CLS96] David A. Cox, John B. Little, Don O'Shea: *Ideals, Varieties and Algorithms* (1996). Springer Verlag.

[Fa99] Jean-Charles Faugère : *A New Efficient Algorithm for Computing Gröbner Bases (F_4)* (1999). Journal of Pure Applied Algebra 139, pp. 61-88

[Fa02] Jean-Charles Faugère : *A new efficient algorithm for computing Gröbner bases (F_4)* (2002). Université Paris. `http://fgbrs.lip6.fr/@papers/F99a.pdf`

[DGS06] Jintai Ding, Jason E. Gower, Dieter S. Schmidt: *Multivariate Public Key Cryptosystems* (2006), pages 199-202. Springer Verlag.

[McC05] Paul McCabe: *Lecture notes for Computational Complexity and Computability* (2005). Lecture notes.

[Coo71] Stephen Cook: *The Complexity of Theorem Proving Procedures* In: Proceedings of the third annual ACM symposium on Theory of computing, pages 151–158.
http://portal.acm.org/citation.cfm?id=805047

[FJL06] K. Fukuda, A. N. Jensen, N. Lauritzen, R. Thomas: *The generic Gröbner walk* (2006). In: arXiv. http://arxiv.org/pdf/math/0501345

[Wer07a] Fabian Werner: Software and further material on this thesis on the web site of Fabian Werner (2007) http://www.happy-werner.de/BSc

[Wer07b] Fabian Werner: IPHFE example script in SINGULAR (2007) http://www.happy-werner.de/BSc/IPHFE.sing

[Wer07c] Fabian Werner: Example on solutions at infinity in SINGULAR (2007) http://www.happy-werner.de/BSc/solAtInf.zip

[Wol02] Christopher Wolf: *"Hidden Field Equations" (HFE) - Variations and Attacks* (2002). Universität Ulm, Diplomarbeit. http://www.christopher-wolf.de/dpl

[DSc02] Jintai Ding, Dieter Schmidt *Cryptoanalysis of HFEv and Internal Perturbation of HFE* (2005). In: PKC 2005, LNCS 3386, pp. 288-301

 THANK YOU FOR READING