NTNU
Norwegian University of
Science and Technology

Post-Quantum Multivariate Cryptography

A Study of Gui and GeMSS Signature
Schemes using Gröbner Bases

Øyvind Follan

Master of Science in Physics and Mathematics
Submission date: July 2018
Supervisor: Øyvind Solberg, IMF

Norwegian University of Science and Technology
Department of Mathematical Sciences

Sammendrag

Målet for denne oppgaven er å beskrive signatursystemene Gui og GeMSS, som er to bidrag sendt inn til National Institute of Standards and Technology sin standardiseringsprosess for kvantesikre kryptografiske primitiver. Begge disse systemene baserer seg på vanskeligheten av å løse simultane polynomligninger i flere variable. Felles for systemene er at de tilhører familien Hidden Field Equations (HFE). Sammen med sine modifikasjoner har HFE vist seg å være et sterkt fundament for kryptografiske systemer. De mest effektive angrepene mot slike systemer er per dags dato angrep ved hjelp av Gröbner-baser, både formulert for klassiske- og kvantedatamaskiner. Et par andre interessante angrep vil bli presentert, der et generisk møtes-i-midten-angrep viser seg å knekke en instans av Gui. Forfatterne bak Gui foreslår å skru på et parameter for å motstå dette angrepet, i bytte mot redusert effektivitet. Forfatterne hevder at kjøretiden for å generere en signatur øker med omtrent 50%, men både eksperimentelle og analytiske resultater gitt i denne oppgaven argumenterer for at kjøretiden øker med en faktor e, Eulers tall.

Abstract

The main goal of this thesis is to analyze and compare the two signature schemes Gui and GeMSS, both submitted to the Post-Quantum Cryptography Standardization Process initiated by the National Institute of Standards and Technology. Both schemes are based on the hardness of solving a system of multivariate polynomial equations, using the construction known as Hidden Field Equations (HFE). HFE together with its modifications have been extensively studied for over 20 years and has withstood the test of time. Today's most promising attacks on HFE-based schemes are Gröbner basis algorithms, both in the quantum and classical setting. Gui and GeMSS will be analyzed with regards to a Gröbner basis approach, as well as some other notable attacks. One instance of Gui and its proposed security level is broken using a generic Meet-in-the-middle attack. The authors of Gui suggest a tweak of parameters to counter this attack, resulting in reduced efficiency of the scheme. The authors claim the run time of signature generation will increase by approximately 50%, while both analytical and experimental results presented in this thesis suggest an increase by a factor of e, Euler's number.

Preface

This thesis concludes a five-year integrated master's degree in Applied Physics and Mathematics at the Norwegian University of Science and Technology (NTNU), with the main profile Industrial Mathematics. The field of study for this thesis is cryptography, with the focus on secure multivariate cryptosystems in a post-quantum world. This thesis was conducted under the supervision of Professor Øyvind Solberg at the Algebra research group at the Department of Mathematical Sciences at NTNU. The thesis is a direct extension of my specialization project from last semester and accounts for 30 ECTS credits. I wish to express my sincerest gratitude to Professor Kristian Gjøsteen and Dr. Martin Strand for valuable feedback and for pointing me in the right direction, and to Ward Beullens, PhD candidate at COSIC, KU Leuven, Belgium, for the helpful e-mail correspondence. I would also like to thank Andreas Sandø Krogen for proofreading my thesis.

Øyvind Follan
Trondheim, Norway
July 2018

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Cryptographers are no longer confident that quantum computers of large scale are impossible. Both conventional Diffie-Hellman and RSA are broken by Shor's quantum computer factoring algorithm proposed in 1994 [1]. As of today, no quantum attacks have been effectively formulated for solving systems of general multivariate polynomial equations. This motivates the use of such systems as a mathematical foundation for cryptographic systems. The National Institute of Standards and Technology, denoted NIST, has initiated a process to standardize quantum-resistant cryptographic systems. Several viable candidates are built on multivariate polynomial equations, and in this thesis schemes based on the so-called Hidden Field Equations (HFE) are studied. HFE has effectively been broken, by the use of Gröbner bases algorithms. Some modifications have been made to the original HFE system, most notably the vinegar (v) and minus (-) modifications. HFE together with these modifications, denoted HFEv-, is considered resistant to Gröbner basis attacks. In this paper, two proposals to the NIST standardization contest built on HFEv- will be presented, namely Gui [2] and GeMSS [3]. The security of these schemes will be evaluated from a Gröbner basis perspective, as well as some other historically successful attacks.

The first part of this thesis, Chapter 2 is a summary of the semester project performed and graded last semester.

Chapter 2

Preliminaries and Gröbner basis Theory

The underlying mathematical foundation for the entire thesis is the theory of multivariate polynomial rings and algebraic extensions. To formally present cryptographic schemes based on multivariate polynomial equations, some notions regarding these building blocks need to be introduced and precisely defined. Also, the main tool for analyzing the security of these schemes, Gröbner bases, will be presented.

2.1 Notions and general definitions

First of all, a polynomial and its components need to be defined.

Definition 1. A *monomial* in the variables $(x_1, x_2, ..., x_n)$ is a product

$$x^{\lambda} = x_1^{\lambda_1} x_2^{\lambda_2} ... x_n^{\lambda_n}, \tag{2.1}$$

where $\lambda_i \in \mathbb{Z}_{\geq 0}$. A *polynomial* p in $(x_1, x_2, ..., x_n)$ is a finite linear combination of monomials, with coefficients p_λ from some commutative field k,

$$p(x_1, x_2, ..., x_n) = \sum_{\lambda} p_\lambda x^{\lambda}. \tag{2.2}$$

The set of all such polynomials is denoted $R = k[x_1, x_2, ..., x_n]$, and is under addition and multiplication a commutative ring. From here on, R denotes such a polynomial ring.

Definition 2. A *system of polynomial equations* is a set of equations

$$\begin{aligned} p_1(x_1, x_2, ..., x_n) &= 0, \\ p_2(x_1, x_2, ..., x_n) &= 0, \\ &\vdots \\ p_m(x_1, x_2, ..., x_n) &= 0, \end{aligned} \tag{2.3}$$

where $p_i \in R$. The solutions (if any) of the given system of polynomials is a set of values of the form $a = (a_1, a_2, ..., a_n) \in k^n$ satisfying $p_i(a) = 0,$ for $1 \leq i \leq m$.

Ideals are certain subsets of rings and are of great importance. An ideal is defined as follows.

Definition 3. An *ideal* I in a commutative ring R is a subset of R satisfying the following properties:

1. I is a subgroup of $(R, +)$.
2. For every pair $(a, b) \in I \times R,\ a \cdot b \in I$.

Example 4. Evidently, both $\{0\}$ and R are ideals of R.

Example 5. The set of even numbers is an ideal in the ring of integers, since any integer multiplied by an even number is an even number.

Explicitly writing out the terms of an ideal is unfeasible, since there are (often) an infinite number of members. Therefore, ideals are commonly presented by some generating set.

Definition 6. Let $F = \{f_1, f_2, ..., f_m\} \subset R$. The *ideal generated by* F in R is defined as

$$\langle F \rangle = \left\{ \sum_{i=1}^{m} g_i f_i \mid g_i \in R \text{ for } 1 \leq i \leq m \right\}. \tag{2.4}$$

It is important to note that an ideal in general has several different generating sets, as can be seen from the following example.

Example 7. The polynomial sets $F_1 = \{x\}$ and $F_2 = \{x, x^2\}$ are obviously different, but generate the same ideal in $R = k[x]$ by definition.

This is a *very* important property, and will be further investigated later. Another important result is that every ideal I in $R = k[x_1, ..., x_n]$ can be expressed by a finite set of generators. This is a famous result, and is a formulation of Hilbert's Basis Theorem.

Theorem 8 (Hilbert's Basis Theorem)**.** *For every ideal $I \subset k[x_1, ..., x_n]$, there exist a finite generating set. In other words, $I = \langle f_1, ..., f_m \rangle$, for some $f_i \in I$.*

Proof. See [4]. □

An interesting observation of ideals in polynomial rings is the following. Let I be an ideal generated by a set of polynomials $\{f_1, ..., f_m\} \subset R$. A solution a to the associated system of polynomial equations $f_i = 0,\ 1 \leq i \leq m$ also satisfies $p(a) = 0$ for every $p \in I$. This is easily seen, directly from the definition of the ideal I. This further motivates the study of ideals generated by a set of polynomials, since the ideals in a sense 'share' zeroes with a generating set of the ideal. A practical notation for the solutions of such a polynomial system is introduced.

Definition 9. Let $F = \{f_1, f_2, ..., f_m\} \subset R$ be a set of polynomials. One defines the *affine variety of F* as

$$V(F) = \{(a_1, a_2, ..., a_n) \in k^n \mid f_i(a_1, a_2, ..., a_n) = 0, \text{ for } 1 \leq i \leq m\}\,. \quad (2.5)$$

One can immediately notice that the solutions of (2.3) are infact the same as $V(p_1, p_2, ..., p_m)$. This motivates further analysis of affine varieties as an algebraic structure, to develop more techniques of finding solutions of such polynomial systems. Similarly, one can define the affine variety of an ideal.

Definition 10. Let I be an ideal in R. Then *the affine variety of I*, denoted $V(I)$ is the set of all points in k^n such that

$$V(I) = \{v \in k^n \mid p(v) = 0,\ \forall p \in I\}\,. \quad (2.6)$$

Since an ideal generally is infinite, finding its affine variety can be difficult. Luckily, there is a strong correspondence between the affine variety of an ideal and the affine variety of a generating set for the ideal.

Theorem 11. *Let I be an ideal in R generated by $F = \{f_1, ..., f_m\}$. Then, the following equalities hold.*

$$V(I) = V(F) \quad (2.7)$$

Proof. The inclusion $V(I) \subset V(F)$ is evident. Since F is in I, any $v \in V(I)$ satisfies $f_i(v) = 0$ for all $1 \leq i \leq m$ by definition.
The inclusion $V(F) \subset V(I)$ also follows. Let $v \in V(F)$. Any element $p \in I$ can be expressed as $p = p_1 f_1 + ... + p_m f_m$. Since $f_i(v) = 0$ for all $1 \leq i \leq m$, indeed $p(v) = (p_1 f_1)(v) + ... + (p_m f_m)(v) = p_1(v) f_1(v) + ... + p_m(v) f_m(v) = 0$, which implies $v \in V(I)$. □

As systems of polynomial equations are of interest, some consideration has to be put on the algebraic multiplicity of the solutions. Let $f_1 = (x+1), f_2 = (y+1) \in GF(2)[x,y]$, and define the two polynomial sets $F = \{f_1, f_2\}, \hat{F} = \{f_1^2, f_2^3\}$. One can immediately notice that $a = (1,1)$ is in both $V(F)$ and $V(\hat{F})$. However, the solutions for the polynomial equations $F = 0$ and $\hat{F} = 0$ have indeed different degrees of algebraic multiplicities. This motivates the following definition.

Definition 12. Let I be an ideal in R. The *radical* of I, denoted $\sqrt{I}$ is an ideal in R, defined as

$$\sqrt{I} = \{f \mid f^\alpha \in I, \text{ for some integer } \alpha \geq 1\}. \tag{2.8}$$

Radical ideals are important for the correspondence between varieties and ideals. We are now ready for the most important result in this section. As previously stated, an ideal need not have a unique generating set, but the affine variety of the ideal is independent of the representation of the ideal. A different generating set merely changes the way to represent a given polynomial in the ideal, but the polynomial itself has the same properties. This means that *any* set of polynomials generating the same ideal I yield the same affine variety. This result follows immediately from Theorem 11 and is summarized in the following corollary.

Corollary 13. *For any two generating sets of polynomials (F, G) for the same ideal I, the following holds*

$$V(F) = V(G). \tag{2.9}$$

Proof. By Theorem 11, $V(I) = V(F)$ and $V(I) = V(G)$. Then it immediately follows $V(F) = V(G)$. □

In the hunt for solutions of multivariate polynomial equations, this result is of utmost importance. Intuitively, this theorem states that a solution of such a system can be found by first constructing the ideal generated by the polynomials, and then try to find some "well organized" generating set for this ideal. Since these polynomial sets share the same affine variety, the hope is that the well-organized set is easier to solve than the original one. This is actually the fundamental idea behind Gröbner basis techniques for solving a set of multivariate polynomial equations. Before a definition of Gröbner bases is given, we need to define multivariate polynomial division. Polynomial division in one variable is trivial, but for polynomials in many variables, several problems arise. For polynomials in one variable, there is a natural way of comparing the orders of monomials, e.g. x^2 is of lower degree than x^3. However, for polynomials with multiple variables, a problem occurs. How should one determine the monomial of smallest degree given x_1^2 and x_1x_2? The notion of monomial ordering is introduced.

Definition 14. A *monomial ordering*, $\succ$, for a set of monomials, $\mathcal{M} \subset R$, is an ordering on the set, for which all pairs $(u, v) \in \mathcal{M} \times \mathcal{M}$ satisfy:

1. $\succ$ is a total ordering on the set,
2. if $u \succ v, \forall w \in \mathcal{M},\ u \cdot w \succ v \cdot w$,
3. $w \succ 1,\ \forall w \in \mathcal{M}$.

Especially three monomial orderings need to be clarified, lexicographic, total degree and graded reverse lexicographic order.

Let $m_1 = x^{\lambda^{(1)}},\ m_2 = x^{\lambda^{(2)}}$ be two monomials in R, where $\lambda^{(1)}, \lambda^{(2)} \in \mathbb{Z}^n_{\geq 0}$.

Definition 15. Lexicographic ordering, lex, first compares the exponents of x_1, then x_2, and so on. More precisely, if $m_1 \succ_{\mathrm{lex}} m_2$, then the leftmost non-zero value of $\lambda^{(1)} - \lambda^{(2)}$ is positive.

An example illustrating $\succ_{\mathrm{lex}}$:

$$m_1 = x_1^5 x_2^9 x_3^2 x_4^6,\ m_2 = x_1^3 x_2 x_3^4 x_4 \implies$$
$$\lambda^{(1)} = (5, 9, 2, 6),\ \lambda^{(2)} = (3, 1, 4, 1) \implies m_1 \succ_{\mathrm{lex}} m_2 \text{ since } \lambda^{(1)} - \lambda^{(2)} = (2, 8, -2, 5)$$

Definition 16. Total degree ordering, tot, is simply defined as

$$m_1 \succ_{\mathrm{tot}} m_2 \iff \sum \lambda^{(1)} > \sum \lambda^{(2)}.$$

An example illustrating $\succ_{\mathrm{tot}}$:

$$m_1 = x_1^5 x_2^3 x_3^5 x_4^1,\ m_2 = x_1^1 x_2^3 x_3^2 x_4^3 \implies$$
$$\lambda^{(1)} = (5, 3, 5, 1),\ \lambda^{(2)} = (1, 3, 2, 3) \implies m_1 \succ_{\mathrm{tot}} m_2 \text{ since } \sum \lambda^{(1)} = 14 > \sum \lambda^{(2)} = 9.$$

Definition 17. Graded reverse lexicographic ordering, $\mathrm{grevlex}$, is defined as

$$m_1 \succ_{\mathrm{grevlex}} m_2 \iff \sum \lambda^{(1)} > \sum \lambda^{(2)},$$

or

$$\sum \lambda^{(1)} = \sum \lambda^{(2)}$$

and the rightmost nonzero entry of $\lambda^{(1)} - \lambda^{(2)}$ is negative.

In other words, $\mathrm{grevlex}$ first compares the total degree of each monomial, but in the event of a tie, it picks the reverse of a lexicographic ordering would yield. Two examples illustrating $\succ_{\mathrm{grevlex}}$:

$$\lambda^{(1)} = (2, 3, 8, 4),\ \lambda^{(2)} = (3, 2, 3, 4) \implies m_1 \succ_{\mathrm{grevlex}} m_2 \text{ since}$$
$$\sum \lambda^{(1)} = 17 > \sum \lambda^{(2)} = 12$$

$$\lambda^{(1)} = (3,8,3),\ \lambda^{(2)} = (6,3,5) \implies m_1 \succ_{\text{grevlex}} m_2 \text{ since}$$
$$\sum \lambda^{(1)} = 14 = \sum \lambda^{(2)} \text{ and } \lambda^{(1)} - \lambda^{(2)} = (-3,5,-2).$$

There is also need for some common definitions for important terms in a polynomial.

Definition 18. Let p be a polynomial $p = \sum_i c_i m_i$, for some monomials m_i, and let $\succ$ be a monomial order. One defines the

- *Leading monomial*:
$$\text{LM}(p) = m_i, \text{ such that } m_i \succ m_j \text{ for all } j \neq i.$$
- *Leading coefficient*:
$$\text{LC}(p) = c_i, \text{ where } c_i \text{ is the coefficient of LM}(p) \text{ in } p.$$
- *Leading term*:
$$\text{LT}(p) = \text{LC}(p)\,\text{LM}(p).$$
- *Least common multiple*: Let $m_1 = x^{\lambda^{(1)}}$, $m_2 = x^{\lambda^{(2)}}$ be two monomials in R. Let $\gamma = (\gamma_1, ..., \gamma_n) \in \mathbb{Z}^n_{\geq 0}$ where $\gamma_j = \max(\lambda^{(1)}_j, \lambda^{(2)}_j)$ for $1 \leq j \leq n$. One defines the *least common multiple*, denoted LCM, of m_1 and m_2 as
$$\text{LCM}(m_1, m_2) = x^\gamma. \tag{2.10}$$

Example 19 (Illustrating LCM)**.** Let $m_1 = x_1^2 x_2^3$ and $m_2 = x_1^2 x_2$ be two monomials in $k[x_1, x_2]$. This leads to $\gamma = (2,3)$. The LCM of m_1 and m_2 is then
$$\text{LCM}(m_1, m_2) = x_1^2 x_2^3 \tag{2.11}$$

The least common multiple of two monomials m_1 and m_2 is the monomial of minimal total degree that is divisible by both m_1 and m_2.

We expand Definition 18 to not only apply to single polynomials. For a set of polynomials F, denote
$$\text{LM}(F) = \{\text{LM}(f) \mid f \in F\}, \tag{2.12}$$
and
$$\text{LT}(F) = \{\text{LT}(f) \mid f \in F\}. \tag{2.13}$$

For an ideal $I \neq 0$, let $\mathrm{LT}(I)$ be the set of leading terms of I,

$$\mathrm{LT}(I) = \{\mathrm{LT}(f) \mid \forall f \in I\}. \tag{2.14}$$

With these definitions under control, we can finally define multivariate polynomial division.

Theorem 20 (Division algorithm in R)**.** *Let $F = \{f_1, f_2, ..., f_m\} \subset R$. For a fixed monomial ordering, any polynomial $p \in R$ can be written as*

$$p = a_1 f_1 + a_2 f_2 + ... + a_m f_m + r \tag{2.15}$$

where $r, a_i \in R$, for all $1 \leq i \leq m$. The remainder of p divided by F is called r, and is either zero or a linear combination of monomials not divisible by any of the $\mathrm{LT}(F)$.

Proof. The proof is given by Algorithm 1.

Algorithm 1: Multivariate Division Algorithm

Input: A set of polynomials $F = \{f_1, ..., f_m\}$, a polynomial p, a monomial ordering $\succ$

Output: $\{a_1, ..., a_m, r\}$ such that $p = a_1 f_1 + ... + a_m f_m + r$

```
a_1 ← 0, ..., a_m ← 0
r ← 0
while p ≠ 0 do
    i ← 1
    Div ← False
    while i ≤ m AND Div == False do
        if LT(f_i) divides LT(p) then
            a_i ← a_i + LT(p)/LT(f_i)
            p ← p − LT(p)/LT(f_i) · f_i
            Div ← True
        else
            i ← i + 1
    if Div == False then
        r ← r + LT(p)
        p ← p − LT(p)
return (a_1, ..., a_m, r)
```

□

For correctness and termination of this algorithm, see [4]. It is important to note that the remainder r is not in general uniquely determined by the set F, and will depend on the order of the division of the polynomials in F. However, using this algorithm, one can actually investigate whether a polynomial is a member of a given ideal or not, stated in the following corollary.

Corollary 21. *Let $I = \langle f_1, f_2, ..., f_m \rangle$ be an ideal in R. For any polynomial $p \in R$, let r be the remainder of p when divided by $\{f_1, ..., f_m\}$. Then,*

$$r = 0 \implies p \in I. \tag{2.16}$$

Note that there is *not* an equivalence between ideal membership and a zero remainder after division, as illustrated in the following example.

Example 22. Let $k = \mathrm{GF}(2)$, $R = k[x, y]$, and $I = \langle y^2 - 1,\ x^2y - 1 \rangle$ under the lexicographic monomial ordering, $x \succ_{\text{lex}} y$. Divide $p = x^2y^2 + xy^2 + x + y$ first by $f_1 = y^2 - 1$, then by $f_2 = x^2y - 1$. We note that $\mathrm{LT}(f_1)$ divides $\mathrm{LT}(p)$ by a factor x^2. Hence one can write $p = x^2 f_1 + x^2 + xy^2 + x + y$. Here, neither $\mathrm{LT}(f_1)$ nor $\mathrm{LT}(f_2)$ divides $\mathrm{LT}(x^2 + xy^2 + x + y) = x^2$, so it is extracted into our remainder, $r_1 = x^2$. The remaining leading term, xy^2 is divisible by $\mathrm{LT}(f_1)$ by a factor x. We write $p = (x^2 + x)f_1 + 2x + y + r_1$. Note that we are currently in $\mathrm{GF}(2)[x, y]$, yielding $2x = 0$. Evidently, y is not divisible by neither $\mathrm{LT}(f_1)$ nor $\mathrm{LT}(f_2)$, arriving at the final remainder $r = x^2 + y$. In this case, it seems p is not in the ideal generated by f_1 and f_2.

This time, first divide p by f_2, then by f_1. We observe that $\mathrm{LT}(f_2)$ divides $\mathrm{LT}(p)$ by a factor y, yielding $p = yf_2 + xy^2 + x$. As $\mathrm{LT}(f_2)$ no longer divides $\mathrm{LT}(p)$, we move on to $\mathrm{LT}(f_1)$. Here, the leading term of f_1 divides $xy^2 + x$ by a factor x, giving the final result $p = yf_2 + xf_1$. The remainder is here 0, hence p is indeed in the ideal generated by f_1 and f_2. This illustrates the point that the order of dividing p by the set F greatly affects the result.

Determining whether or not a polynomial is in an ideal is an important and well-studied problem. This motivates the search for some representation of a given set of polynomials, such that the remainder is independent of the order of the division. It turns out that Gröbner bases satisfy this exact criterion. We are now ready to present the following definition.

Definition 23 (Gröbner basis)**.** For a fixed monomial ordering, a finite set of polynomials $G = \{g_1, g_2, ..., g_l\}$ in an ideal $I \subset R$ is a *Gröbner basis* of I if

$$\langle \mathrm{LT}(g_1), \mathrm{LT}(g_2), ..., \mathrm{LT}(g_l) \rangle = \langle \mathrm{LT}(I) \rangle. \tag{2.17}$$

In general, the ideal generated by a set of polynomials is different from the ideal generated by the leading terms in the set. Also, as a direct consequence of Theorem 8, the following corollary is evident.

Corollary 24. *Every non-zero ideal $I \subset k[x_1, ..., x_n]$ has a Gröbner basis.*

As shown in Example 22, chosing the correct order of division was crucial for establishing whether or not p was in the ideal I. This leads to the following important and useful trait of Gröbner bases.

Proposition 25. *Let $G = \{g_1, g_2, ..., g_l\}$ be a Gröbner basis of an ideal $I \subset R$ for some monomial ordering, and let $p \in R$. Then, the remainder of p when divided by G is independent of the order of division.*

Proof. Suppose the remainder is not unique. Then, the division algorithm gives $p = a_1 g_1 + a_2 g_2 + ... + a_l g_l + r$ and $p = a_1^{'} g_1 + a_2^{'} g_2 + ... + a_l^{'} g_l + r^{'}$, where $a_i, a_i^{'}, r, r^{'} \in R$ for $1 \leq i \leq l$. Since we know that $\langle G \rangle$ is an ideal, indeed

$$r - r^{'} = (a_1 - a_1^{'}) g_1 + (a_2 - a_2^{'}) g_2 + ... + (a_l - a_l^{'}) g_m \ \in \langle G \rangle. \tag{2.18}$$

By the definition of a Gröbner basis we know that $\mathrm{LT}(r - r^{'})$ is not divisible by any leading term in G, since no term of r nor $r^{'}$ is divisible by any $\mathrm{LT}(G)$. The only option for $r - r^{'} \in \langle G \rangle$ is $r - r^{'} = 0$. The uniqueness of the remainder follows. □

It is important to note that even though the remainder r is unique, the elements (a_i, a_i') in the above representation need not be, meaning

$$p = a_1 g_1 + ... + a_l g_l + r = a_1^{'} g_1 + ... + a_l^{'} g_l + r, \tag{2.19}$$

where $a_i \neq a_i'$ is perfectly possible. As a direct consequence of Propopsition 25, the following can be stated regarding ideal membership.

Corollary 26. *For an ideal $I \subset R$, let G be an associated Gröbner basis of I. Any polynomial $p \in R$ is a member of I if and only if the remainder of p divided by G is zero.*

Proof. If the remainder after division is zero, p is indeed in I by Corollary 21. Conversely, if p is in I, and $G = \{g_1, ..., g_l\}$ is a Gröbner basis for I, then there exists a representation of p in the following form

$$p = a_1 g_1 + ... + a_l g_l. \tag{2.20}$$

Since the remainder is unique after dividing p by G, and this representation clearly shows a remainder of zero, the division of p by G always yields a zero remainder. □

This is an important result, both from a theoretical standpoint to determine ideal memberships, but also from an algorithmic viewpoint, as this enables several Gröbner basis algorithms to deterministically terminate. One of the most famous algorithms for finding a Gröbner basis is called Buchberger's Algorithm, named after the German mathematician who is credited with the discovery of Gröbner bases. Before an explanation of Buchberger's Algorithm is presented, the notion of S-polynomials needs to be introduced.

Definition 27. Let f, g be two elements in R. For a given monomial ordering, one defines the *S-polynomial* of f and g as

$$S(f,g) = \frac{\mathrm{LCM}(\mathrm{LM}(f), \mathrm{LM}(g))}{\mathrm{LT}(f)} f - \frac{\mathrm{LCM}(\mathrm{LM}(f), \mathrm{LM}(g))}{\mathrm{LT}(g)} g. \tag{2.21}$$

By construction, the leading terms of f and g cancel out, resulting in a polynomial satisfying $f \succ S$ and $g \succ S$. These S-polynomials are crucial when calculating a Gröbner basis, as they serve an important role in determining whether or not a Gröbner basis has been found, stated in the following theorem.

Theorem 28 (Buchberger's Criterion)**.** *For a given ideal $I \subset R$, a generating set $G = \{g_1, ..., g_l\}$ is a Gröbner basis for I if and only if the remainder of $S(g_i, g_j)$ divided by G is zero, for all pairs $(g_i, g_j) \in G$.*

Proof. $\Longrightarrow$: Since G is a Gröbner basis for the ideal I, and $S(g_i, g_j) \in I$, it follows directly from Corollary 26.

$\Longleftarrow$: See [4] □

Now, Buchberger's Algorithm is ready to be presented, as one of the fundamental tools for finding Gröbner bases.

Algorithm 2: Buchberger's Algorithm

Input: A set of polynomials $F = \{f_1, ..., f_m\}$ which generate an ideal I,
A monomial ordering $\succ$

Output: A Gröbner basis G for I

```
G ← F
repeat
    G' ← G
    for each pair (p, q) in G' do
        S ← S(p, q)
        S' ← reduce S by the elements of G' using polynomial division
        if S ≠ 0 then
            G ← G ∪ {S}
until G = G'
return G
```

For correctness and termination, the reader is referred to [4].

The Gröbner basis for an ideal is not necessarily uniquely determined. For this reason, the *reduced Gröbner basis* is introduced, as it is indeed a unique representation of the ideal satisfying the criterion of a Gröbner basis.

Definition 29. A Gröbner basis G is a *reduced Gröbner basis* if

1. $LC(p) = 1 \; \forall p \in G$,

2. $\forall p \in G$, no monomial of p lies in $\langle \mathrm{LT}(G \setminus p) \rangle$.

In other words, it is in some sense the smallest Gröbner basis for an ideal, as removing any term yields a set not satisfying the criteria of a Gröbner basis. From here on, the reader can assume that once a Gröbner basis is mentioned, it is indeed reduced. As we are now comfortable discussing Gröbner bases, their applications can be further discussed. As this next section will show, solving a system of multivariate polynomial equations and finding a Gröbner basis under a certain monomial ordering are closely linked.

2.2 Elimination theory

To illustrate the power of Gröbner basis algorithms for solving systems of equation, we begin this section with an example.

Example 30. Let $k = \mathbb{Z}/13\mathbb{Z}$ and $R = k[x, y, z]$. Examine the polynomial system $f_i = 0$ for $i = 1, 2, 3$,

$$\begin{aligned} f_1 &= xz^2 - y - 1 = 0, \\ f_2 &= x^2y - 2 = 0, \\ f_3 &= yz^2 - 2 = 0. \end{aligned} \tag{2.22}$$

Let I be the ideal generated by $F = \{f_1, f_2, f_3\}$. By computing the reduced Gröbner basis for I under the lexicographic monomial order, we arrive at $G = \{g_1, g_2, g_3\}$ where

$$\begin{aligned} g_1 &= x - 3z^8 + 6z^6 + 3z^2 + 6, \\ g_2 &= y + 6z^8 - 6z^2 + 2, \\ g_3 &= z^{10} - z^4 - 4z^2 - 4. \end{aligned} \tag{2.23}$$

As previously stated, solving (2.22) is equivalent to finding the affine variety of G. Clearly, the set G is easier to analyze, since g_3 is a polynomial in one variable, while both g_1 and g_2 are on the simple forms $x - h_1(z)$ and $y - h_2(z)$ respectively. There are numerous efficient algorithms for finding roots of a polynomial in one variable, e.g. Berlekamp's Algorithm [5]. One can easily see that $z = \pm 2$ satisfies $g_3 = 0$. Next, one can observe that $g_1 = 0 \implies x = h_1(z)$, meaning that $x = h_1(\pm 2) = 2$ and similarly $g_2 = 0 \implies y = h_2(\pm 2) = 7$ give all the solutions in k. To summarize, there are two solutions to (2.22), namely $(2, 7, \pm 2)$. Note that the degree of g_3 is much higher than any of the polynomials in the original set F. This is a recurring problem when finding Gröbner bases, the degrees of the polynomials might grow exceptionally high.

The above example illustrates a particularly beautiful result, and needs to be further investigated. Notice how g_3 in the above example is a polynomial in $I \cap k[z]$, where $I \cap k[z]$ contains all the elements in the ideal I where x, y have been eliminated. This itself is an ideal, and is in fact generated by the element g_3. Formally, this is called an elimination ideal, and is defined in the following way.

Definition 31. For an ideal $I = \langle f_1, ..., f_m \rangle \subset k[x_1, ..., x_n]$, the *i-th elimination ideal*, denoted I_i, is defined as

$$I_i = I \cap k[x_{i+1}, ..., x_n] \tag{2.24}$$

Note that different monomial orderings lead to different elimination ideals. Elimination ideals conserve some of the important underlying structure of a Gröbner basis, formalized in the following Theorem.

Theorem 32 (The Elimination Theorem)**.** *Let I be an ideal in $k[x_1, ..., x_n]$, and let G be a Gröbner basis of I with respect to the lexicographic ordering, $x_1 \succ_{\text{lex}} x_2 \succ_{\text{lex}} ... \succ_{\text{lex}} x_n$. Then, a Gröbner basis for the i-th elimination ideal is given by*

$$G_i = G \cap k[x_{i+1}, ..., x_n] \tag{2.25}$$

Proof. For any i, it is clear that $G_i \subset I_i$, which implies

$$\text{LT}(G_i) \subset \text{LT}(I_i) \implies \langle \text{LT}(G_i) \rangle \subset \langle \text{LT}(I_i) \rangle. \tag{2.26}$$

Hence, it remains to show

$$\langle \text{LT}(I_i) \rangle \subset \langle \text{LT}(G_i) \rangle. \tag{2.27}$$

Any $p \in I_i$ is also in I. Hence, the leading term of p is divisible by a leading term of some $g \in G$, since G is a Gröbner basis of I. But since p only involves the variables $(x_{k+1}, ..., x_n)$, then also g must be in the variables $(x_{k+1}, ..., x_n)$, by the lexicographic ordering. Hence, $g \in \langle G_i \rangle$. Together, this implies that G_i is a Gröbner basis for I_i. □

For finite fields, ideals with an affine variety consisting of a finite set of points produce a very interesting reduced Gröbner basis under the lexicographic ordering. This is an extremely important result and is the foundation of a Gröbner basis approach to solving a system of multivariate polynomial equations.

Lemma 33 (Shape Lemma)**.** *For a finite field k, let I be a radical ideal where the $n-$th coordinates of the points in $V(I)$ are distinct. Then, denote the generator of the $n-1$th elimination ideal, $I \cap k[x_n]$, as g_n. Then,*

1. *The reduced Gröbner basis of I with respect to $\succ_{\text{lex}}$ is*

$$G = \{x_1 - g_1, x_2 - g_2, ..., x_{n-1} - g_{n-1}, g_n\},$$

   where $g_i \in k[x_n]$. for all $1 \leq i \leq n$

2. *The set of zeros of I is*

$$V(I) = \{(g_1(a_i), ..., g_{n-1}(a_i), a_i) \text{ for } i = 1, ..., \deg(g_n)\},$$

   where the a_i are the roots of g_n.

Proof. See [6]. □

This Lemma formalizes the result found in Example 30. For our cryptographic purposes, the base fields are always finite, and the most commonly used are $\mathrm{GF}(2)$ and $\mathrm{GF}(2^8)$. By appending the so-called *field equations* ($x_i^q - x_i$, where q is the number of elements in the base field) the resulting ideal is radical, see [7]. The solutions of the polynomial systems appearing in our cryptographic context are always finite, and with high probability unique [8]. In this case, the generator for the n-th elimination ideal has a single root. Hence, the criteria in the Shape Lemma are indeed satisfied. The Shape Lemma is a powerful result since any polynomial system represented in this form is practically solved. This is a problem for cryptographic systems based on polynomial equations since they are broken if the system is rewritten to such a form. However, even though the Shape Lemma does indeed guarantee the existence of such a representation, it does not say much about the complexity and hardness of finding it. Computing a Gröbner basis turns out to be computationally challenging. The Buchberger Algorithm, Algorithm 2, was proposed more than 40 years ago and was not initially constructed to be a fast tool. More recently, several new algorithms have been proposed.

2.3 Fast Gröbner basis algorithms

The most computationally expensive part of Buchberger's algorithm is the reduction of the S-polynomials. However, in 1999 Jean-Charles Faugère proposed a new algorithm, named F4 [9]. The general idea behind F4 is very similar to Buchberger's algorithm, but one major difference is that it uses ideas from linear algebra to reduce several polynomials simultaneously. Later, Faugère also presented the improved F5 Algorithm [10]. Common for these algorithms, is that they are much faster when calculating a Gröbner basis under the degree reverse lexicographical ordering or total degree ordering. As the Shape Lemma applies to a Gröbner basis under lexicographical ordering, these algorithms are not quite enough to arrive at the solution of a polynomial system. Luckily, there are algorithms for converting a given Gröbner basis under a monomial ordering to a Gröbner basis for another monomial ordering, most notably FGLM [11, 12].

The run time of the F5 algorithm is heavily dominated by row reduction of huge matrices. The size of these matrices are determined by the number of equations and variables in the system of equations, as well as the so-called *degree of regularity*, d_{reg}, of the system of equations [13]. The degree of regularity of a system of equations will be further discussed in 4.3. Particularly, for a system of n equations in n variables, the complexity of F5 is dominated by row reducing a matrix consisting of $\binom{n}{d_{\mathrm{reg}}}$ columns and rows. Row reducing a matrix of size

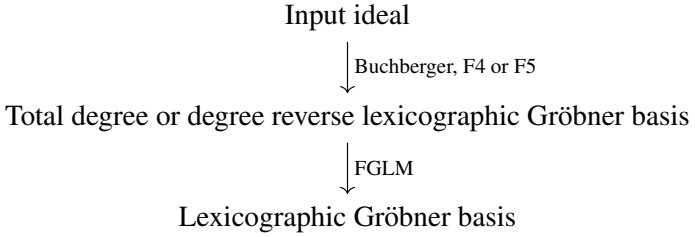

Figure 2.1: Illustration of the algorithmic workflow for finding a Gröbner basis under lexicographic ordering.

$\binom{n}{d_{\text{reg}}} \times \binom{n}{d_{\text{reg}}}$ can be performed in

$$C_{\text{RowReduce}} \sim \mathcal{O}\binom{n}{d_{\text{reg}}}^{\omega} \tag{2.28}$$

operations. The constant ω is called the linear algebra constant, and is determined by the complexity of matrix multiplication. In [14] the authors present $\omega = 2.38$, the lowest value to our knowledge. The interested reader is referred to [15] to learn about F5.

There are other techniques for solving general systems of polynomial equations, where one approach is linearization. If the polynomial system only contains monomials of degree 2 or lower, the general idea is to replace each product $x_i x_j$ by a new variable y_{ij}. To illustrate how effective this approach can be, let $F = \{f_1, ..., f_m\}$ be an overdetermined set of quadratic equations in $(x_1, ..., x_n)$, where $m = n(n+1)/2$. These m quadratic equations give rise to $n(n+1)/2$ linear equations in $n(n+1)/2$ variables after linearization, which can easily be solved by Gaussian Elimination. As we wish to solve the original system for $(x_1, ..., x_n)$, we proceed as follows. At most, two possible values x_i satisfy $x_i^2 = y_{ii}$. To chose the correct value for each x_i, the relations $x_i x_j = y_{ij}$ are used, and the correct solutions are relatively easily found.

Linearization techniques are also useful for less overdetermined systems, with the introduction of Relinearization [16]. Here, the same linearization is made, but additional equations are introduced as well, to account for the commutative properties of $x_i x_j$, namely

$$\begin{aligned}(x_i x_j)(x_k x_l) = (x_i x_k)(x_j x_l) = (x_i x_l)(x_j x_k) \implies \\ y_{ij} y_{kl} = y_{ik} y_{jl} = y_{il} y_{jk}\end{aligned} \tag{2.29}$$

These techniques can easily be combined with Gröbner basis algorithms, and in fact direct links between Relinearization algorithms and Gröbner basis algorithms

have been found [17]. Now that some theoretical foundation has been made, we can commence discussing the applications of multivariate polynomial equations in cryptography.

Chapter 3

Cryptography and Hidden Field Equations

Secure communication has become incredibly important over the last few decades with the development of the global communication infrastructure. There are endless areas where being able to communicate securely is of utmost importance, e.g. economy, health, social networks, cloud computing, identification, and a lot more. Public Key Cryptography has been shown to be an essential part of our communication protocols, both for encryption and digital signatures. It turns out multivariate polynomial systems can be used as a tool for both. Signature schemes are important for the authenticity of a message. A signature scheme generally consists of three algorithms.

Definition 34 (Signature scheme)**.** A signature scheme is a collection of algorithms $(\mathrm{KeyGen}, \mathrm{Sig}, \mathrm{Ver})$, that ensure the authenticity of a communicated message, where

- KeyGen generates a public key and a secret key $(\mathrm{pk}, \mathrm{sk})$.
- Sig takes as input a message M and the secret key sk and produces a signature σ.
- Ver takes as input a message M, the public key pk and a signature σ, and produces a boolean.

For the correctness of the signature scheme, the following must be satisfied for a given message M and a key pair $(\mathrm{pk}, \mathrm{sk})$,

$$\mathrm{Ver}(M, \mathrm{pk}, \mathrm{Sig}(M, \mathrm{sk})) = \mathrm{True}\,.$$

Also, for any message $M' \neq M$,

$$\mathrm{Ver}(M', \mathrm{pk}, \mathrm{Sig}(M, \mathrm{sk})) = \mathrm{False},$$

with high probability.

When sending a message, the sender would simply attach the signature produced by Sig. Then, the receiver would input the message, the public key, and the signature into Ver, and only trust the correspondence if it outputs True. This way, any manipulation of the message over the insecure communication channel would lead to a discarding of the message received.

Encryption schemes are also very important and are used to hide the information in a message by converting it into seemingly random text. This is done with the help of mathematical mappings, where only the designated party holding some secret information is able to find the preimage of the map.

Definition 35 (Encryption scheme)**.** An encryption scheme is a collection of algorithms that provide confidentiality of messages. An encryption scheme usually consist of the following three algorithms $(\mathrm{KeyGen}, \mathrm{Enc}, \mathrm{Dec})$

- KeyGen generates a public key and a secret key $(\mathrm{pk}, \mathrm{sk})$.
- Enc takes as input a plaintext, M, and the public key pk, and produces a ciphertext, C.
- Dec takes as input a ciphertext C, the secret key sk and reproduces the plaintext M.

For correctness of the encryption scheme, the following must be satisfied for a given plaintext M and a pair $(\mathrm{pk}, \mathrm{sk})$,

$$\mathrm{Dec}(\mathrm{Enc}(M, \mathrm{pk}), \mathrm{sk}) = M.$$

These two families of schemes combined give a strong security setting around communication. If a message is guaranteed to be unreadable for anyone other than the intended receiver, and one can ensure that the sender is indeed the correct sender, one can be fairly confident that the communication is secure.

Another important tool used extensively in cryptography is the use of hash functions. Hash functions are one-way functions that generally accept an input of any length, and returns an output of fixed length.

As mentioned in the introduction, many of the cryptographic systems currently in use will be broken in the event of a large-scale quantum computer being built, motivating the search for cryptosystems based on other primitives. Additionally,

some of the already existing and implemented cryptosystems, e.g. AES [18], can, in fact, be modeled by a set of multivariate polynomial equations. Hence, further research within this field is of great interest, both as a means to find new quantum secure systems, but also to further develop our understanding of currently implemented cryptosystems.

Several cryptosystems based on multivariate polynomials have been proposed. Common for all these systems is the need for fast evaluation and difficult inversion. This means that a system of completely random multivariate polynomials will not work, as such a system is hard to invert even for the intended entity. One is dependent on having some secret information regarding the system, to ensure fast inversion. This means that the polynomial equations need to be constructed in some clever sense, such that inversion is easy for the intended party, while still being nearly impossible for anyone else. Many of the so far suggested systems based on multivariate polynomial equations are further developments on the C* system proposed by Matsumoto and Imai in 1988 [19]. However, C* was fairly quickly broken, but it inspired the community to further investigate cryptosystems based on multivariate polynomials. One of the successors of C* is Hidden Field Equations, denoted HFE [20].

3.1 HFE for Encryption Schemes

The general idea behind HFE is to arrive at a seemingly random system of polynomial equations after composing several maps that are easy to invert. The seemingly random system of equations is our public key, while the underlying composition is the secret key. To explain the system more precisely, let $\mathbb{F} = \mathrm{GF}(q)$, where q is some power of a prime p. Fix some irreducible polynomial of degree n, denoted $g(x) \in \mathbb{F}[x]$, and denote $\mathbb{E}$ an algebraic extension of $\mathbb{F}$, constructed in the obvious way $\mathbb{E} = \mathbb{F}[x]/\langle g(x) \rangle$. Then $|\mathbb{E}| = q^n$, and there is a natural way to identify the extension field $\mathbb{E}$ as a vector space over $\mathbb{F}$. The basis vectors are the powers of a primitive root of $g(x)$ in $\mathbb{E}$, denoted by ξ. This isomorphism between $\mathbb{E}$ and $\mathbb{F}^n$ is fundamental for the HFE construction. Let ϕ denote such an isomorphism, defined in the following way,

$$\begin{aligned} \phi &: \mathbb{F}^n \to \mathbb{E} \\ \phi(x_1, ..., x_n) &= (x_1 + x_2\xi + ... + x_n\xi^{n-1}). \end{aligned} \tag{3.1}$$

The fundamental idea of HFE is to find a polynomial $F(X)$ in $\mathbb{E}[X]$, denoted the central polynomial, with certain properties, and then find the representation of this

polynomial as a map from $\mathbb{F}^n$ to $\mathbb{F}^n$. The central map has the following form,

$$F\colon \mathbb{E} \to \mathbb{E}$$
$$F(X) = \sum_{i,j}^{q^i+q^j \leq D} \alpha_{i,j} X^{q^i+q^j} + \sum_{k}^{q^k \leq D} \beta_i X^{q^k} + \gamma, \tag{3.2}$$

where $\alpha, \beta, \gamma \in \mathbb{E}$, and D is some fixed degree. We fix some $X \in \mathbb{E}$, which can be represented as $X = x_1 + x_2\xi + x_3\xi^2 + ... + x_n\xi^{n-1}$. After applying F on X and calculating the result modulo the irreducible polynomial $g(x)$, we obtain the following element in $\mathbb{E}$,

$$F(X) = f_1 + f_2\xi + f_3\xi^2 + ... + f_n\xi^{n-1}, \tag{3.3}$$

where the f_i depend on the given x_i in the representation of X. By applying the isomorphism (3.1), we define $F' = \phi^{-1} \circ F \circ \phi$,

$$F'\colon \mathbb{F}^n \to \mathbb{F}^n$$
$$F'(x_1, ..., x_n) = (f_1(x_1, ..., x_n), f_2(x_1, ..., x_n), ..., f_n(x_1, ..., x_n)). \tag{3.4}$$

We note that F' represents the central map in its *multivariate representation* over $\mathbb{F}^n$, while F denotes the central map in its *univariate representation* over $\mathbb{E}$. The f_i are quadratic by the choice of the central map F, since the Frobenius map $X \to X^q$ is linear over $\mathbb{E}$. A term in the first sum in (3.2) is simply the product of two linear maps, resulting in a term of degree at most 2. Next, two affine bijections S and T in $\mathbb{F}^n$ are introduced. They are crucial for the cryptosystem, as they are used to hide the structure of the central polynomial. Both S and T can be written as polynomials of total degree 1, and the composition of S, F' and T still yield quadratic polynomials. The final system of polynomials produced by the HFE construction is

$$P\colon \mathbb{F}^n \to \mathbb{F}^n,$$
$$P = S \circ \phi^{-1} \circ F \circ \phi \circ T, \tag{3.5}$$

which is the public key in our scheme. The system of polynomials is explicitly written in its multivariate form as

$$p_1(x_1, ..., x_n) = \sum_{1 \leq i \leq j \leq n} \alpha_{1,i,j} x_i x_j + \sum_{i=1}^{n} \beta_{1,i} x_i + \gamma_1$$
$$\vdots$$
$$p_n(x_1, ..., x_n) = \sum_{1 \leq i \leq j \leq n} \alpha_{n,i,j} x_i x_j + \sum_{i=1}^{n} \beta_{n,i} x_i + \gamma_n. \tag{3.6}$$

Together, the three maps (S, F, T) comprise the secret key. An important observation of the system of equations, is that it is not necessarily bijective. The central map F of degree D can have several solutions y to $F(y) = a$, for some $a \in \mathbb{E}$. As this map is essential to the encryption and decryption scheme, it is important that one is able to retrieve the original message when decrypting an encrypted message. To be able to retrieve the original message, some redundancy is introduced in the plaintext, to be able to distinguish the correct plaintext from potential other candidates found from another solution to $F(y) = a$. Instead of encrypting a message M' directly, one common approach is to concatenate the message with its hash value for some suitable hash function. This way, the intended party can distinguish between the potential several solutions by chosing the one corresponding to the given hash value. Let $M = (M_1, ..., M_n) \in \mathbb{F}^n$ be the intended message with the redundancy attached. Encryption of M consists of simply evaluating the n public polynomials in the points $(M_1, ..., M_n)$. Evaluation is extremely fast, and can be done by anyone. Note that encrypting a message of arbitrary length is easily achieved, by partitioning the message in blocks of length n and potentially adding some padding at the last block. Hence, we assume message length n to simplify notation. The encrypted ciphertext of M is

$$C = (p_1(M_1, ..., M_n), p_2(M_1, ..., M_n), ..., p_n(M_1, ..., M_n)). \tag{3.7}$$

To restore the message M from a received ciphertext C, simply use the inverse map of T, and map the result into the extension field by ϕ. Then, search for Y in $\mathbb{E}$ such that $F(Y) = T^{-1}(C)$. Since there can be D solutions, the redundancy is important here. If there are several possible roots of $F(Y) - T^{-1}(C)$, choose the one compliant with the agreed-upon redundancy. Next, the element in $\mathbb{E}$ is mapped into the vector space $\mathbb{F}^n$ by ϕ^{-1}, and the inverse of S is applied. The original message M is then retrieved.

Figure 3.1 illustrates how an entity with knowledge of the secret transformations can move between the different stages of the scheme. However, by only using the public polynomials, it is much harder to go from c to M. To clear up any confusion, the following extensive example shows how HFE can be used in encryption.

Example 36 (HFE encryption example)**.** Let $\mathbb{F} = \mathrm{GF}(2)$ and $g(x) = x^3 + x^2 + 1$. Hence, our extension field is $\mathbb{E} \cong \mathbb{F}[x]/\langle g(x) \rangle$. Denote ξ some primitive root of $g(x)$ in $\mathbb{E}$. Let the central map F be defined as

$$F(X) = X^{(2^2+2^1)} + X^{(2^1+2^1)} + X^{2^1} + 1 \tag{3.8}$$

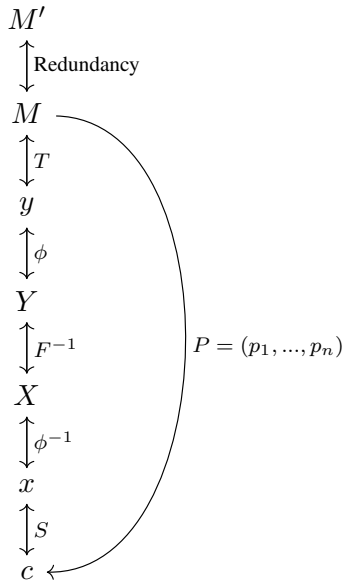

Figure 3.1: Diagram showing the basic HFE construction. The maps (S, F, T) are two-way maps and easy to invert. The public key P is a one-way map.

and the affine transformations T, S are given as

$$T\colon u \to A_1 u + b_1 \text{ where } A_1 = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix} \text{ and } b_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \tag{3.9}$$

$$S\colon u \to A_2 u + b_2 \text{ where } A_2 = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } b_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}. \tag{3.10}$$

We wish to encrypt the message $M = (M_1, M_2, M_3) \in \mathbb{F}^3$. First, we apply T on M, and get

$$y = T(M) = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ M_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} M_2 + M_3 \\ M_1 + 1 \\ M_1 + M_3 \end{bmatrix}. \tag{3.11}$$

By the isomorphism (3.1), this can be represented in $\mathbb{E}$ as $Y = M_2 + M_3 +$

$(M_1+1)\xi+(M_1+M_3)\xi^2$. By applying the central map, one is left with,

$$\begin{aligned} Z=F(Y) = 1&+(M_2+M_3+(M_1+1)\xi+(M_1+M_3)\xi^2)^2 \\ &+(M_2+M_3+(M_1+1)\xi+(M_1+M_3)\xi^2)^4 \\ &+(M_2+M_3+(M_1+1)\xi+(M_1+M_3)\xi^2)^6 \\ &= (M_1M_3+M_2M_3+M_1)+(M_1M_2+M_1+M_2+M_3)\xi \\ &+(M_1M_2+M_2M_3+M_3+1)\xi^2, \end{aligned} \tag{3.12}$$

after reducing modulo the polynomial $g(x)$. Using the bijection between $\mathbb{E}$ and $\mathbb{F}^3$, we denote $z=\phi^{-1}(Z)$,

$$z=(M_1M_3+M_2M_3+M_1,\ M_1M_2+M_1+M_2+M_3,\ M_1M_2+M_2M_3+M_3+1). \tag{3.13}$$

Denote by z_i the i-th component of z. The affine transformation S is applied to z, resulting in

$$S(z)=\begin{bmatrix}0&1&1\\1&1&0\\0&0&1\end{bmatrix}\begin{bmatrix}z_1\\z_2\\z_3\end{bmatrix}+\begin{bmatrix}1\\0\\0\end{bmatrix}=\begin{bmatrix}z_2+z_3+1\\z_1+z_2\\z_3\end{bmatrix}. \tag{3.14}$$

Hence, the resulting system of polynomials is

$$\begin{aligned} p_1(M_1,M_2,M_3) &= M_2M_3+M_1+M_2, \\ p_2(M_1,M_2,M_3) &= M_1M_2+M_1M_3+M_2M_3+M_2+M_3, \\ p_3(M_1,M_2,M_3) &= M_1M_2+M_2M_3+M_3+1, \end{aligned} \tag{3.15}$$

and comprises the public key. We wish to encrypt and send the message $M=(1,1,1)$. By evaluating the public key in the coordinates of M, we achieve the ciphertext $C=(C_1,C_2,C_3)$, where

$$\begin{aligned} C_1 &= p_1(1,1,1)=1, \\ C_2 &= p_2(1,1,1)=1, \\ C_3 &= p_3(1,1,1)=0. \end{aligned} \tag{3.16}$$

To ensure correctness of the encryption scheme, we need $\mathrm{Dec}(C,\mathrm{sk})=M$. Simply, the transformations are performed in reverse. First, the inverse of S is applied to C, resulting in

$$z=S^{-1}(C)=\begin{bmatrix}1&1&1\\1&0&1\\0&0&1\end{bmatrix}\left(\begin{bmatrix}1\\1\\0\end{bmatrix}-\begin{bmatrix}1\\0\\0\end{bmatrix}\right)=\begin{bmatrix}1\\0\\0\end{bmatrix}. \tag{3.17}$$

Next, we represent z in the extension field $\mathbb{E}$ as $Z = 1$, and search for a value $Y \in \mathbb{E}$ such that $F(Y) = Z$. It is easily found that $Y = 0$ is a solution, and in fact the single solution. In general, there might be several solutions at this step. Next, we denote $y = \phi^{-1}(Y)$, and the inverse of T is applied to y, yielding

$$M = T^{-1}(y) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}. \tag{3.18}$$

Hence, the encryption scheme is consistent, as $\mathrm{Dec}(\mathrm{Enc}(M, \mathrm{pk}), \mathrm{sk}) = M$.

One can also note that a Gröbner basis attack is extremely simple in this example. Denote I the ideal generated by $(p_1 - C_1, p_2 - C_2, p_3 - C_3)$. Under the natural lexicographic ordering $M_1 \succ_{\text{lex}} M_2 \succ_{\text{lex}} M_3$, the Gröbner basis is found to be

$$G = \{M_1 + 1, M_2 + 1, M_3 + 1\}, \tag{3.19}$$

and has indeed the affine variety

$$V(G) = \{(1, 1, 1)\}, \tag{3.20}$$

which is our original message M.

Naturally, for such a small example, the Gröbner basis attack was extremely easy and took less than 10^{-4} seconds to perform using Macaulay2. However, for the HFE construction for increasing n, increasing degree of the algebraic extension, and increasing D, it gets significantly harder. In Figure 3.2, some run times are shown for computing a Gröbner basis under the lexicographic ordering for HFE(D, n), ranging from $n = 5$ to $n = 35$, for $D \in \{6, 60\}$ using Macaulay2. Interestingly, for systems with a fairly low degree central map, the algebraic attack is exceptionally much faster. For $D = 60$, the runtime is exponential in n.

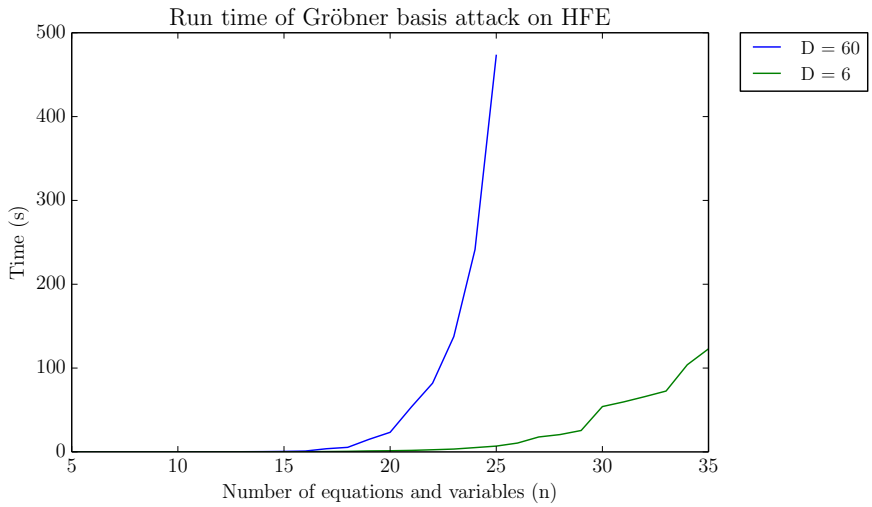

Figure 3.2: Run times for algebraic attacks on HFE constructed in Macaulay2, for increasing number of equations and variables.

In the extended version of [20], Patarin proposed two HFE challenges with a symbolic prize of 500 USD to the first person able to break one. Faugère and Joux showed in 2003 that one of the challenges was indeed breakable [15]. The challenge they broke was an HFE construction over $\mathrm{GF}(2)$ where the degree of the central map was 96, and the polynomial system involved a total of 80 equations in 80 variables. This was a major breakthrough for Gröbner basis computations, as this problem was thought to be intractable. The authors used the F5 algorithm, with some custom modifications for binary arithmetics. As the original HFE system was practically broken, something needed to be done to strengthen the security of the scheme. Some of the measurements made to improve the scheme actually make HFE less suited for encryption schemes, while being strong candidates for signature schemes. Before the most notable modifications for HFE is given, the fundamental idea behind HFE signature schemes is presented.

3.2 HFE for Signature Schemes

One of the most basic signature schemes based on HFE is simply a small tweak of the encryption scheme proposed above. Let $P = S \circ \phi^{-1} \circ F \circ \phi \circ T = (p_1, ..., p_n)$ be a set of public polynomials constructed as previously described. Then, let $d = \mathcal{H}(M)$ be the hash value of a message M to be signed, for some suitable hash function $\mathcal{H}$. A signature for M can then simply be a decryption C of d using the secret key (S, F, T). Then, the signature is verified by evaluating the public

polynomials in the point C, and comparing

$$P(C) = \mathcal{H}(M). \tag{3.21}$$

Even though this is a very simple scheme it still shares the same fundamental idea as more advanced schemes. However, as the central map F need not be a permutation of the elements over the extension field, there is not necessarily a valid signature for any digest, as the inversion of F might not be possible. For this reason, some randomness is introduced to the digest. As will be evident in the next chapter, the modifications made to HFE to strengthen its security make encryption and decryption less efficient and more complicated. However, in signature schemes, these tweaks serve as great tools.

Chapter 4

Modern HFE and its attacks

The original HFE construction was practically broken by Faugère and Joux [15]. After several years of research, analysis of HFE lead to several tweaks of the basic design. Together with these tweaks, HFE has been found to be a very secure construction. In this chapter, these design rationales are presented, together with the most effective attacks to the modified HFE as of 2018.

4.1 Hidden Field Equations Modifications

The most notable tweaks to HFE are the so-called vinegar and minus modifiers. The basic idea of these modifiers is as follows.

- Vinegar (v): Some variables are added to the system of polynomials.
- Minus $(-)$: Some equations are removed from the public key.

We denote HFE using the vinegar, minus or both modifications by HFEv, HFE-, and HFEv-, respectively. The minus modifier was first introduced in the schemes FLASH and SFLASH [21] in order to withstand linearization attacks, as mentioned in Chapter 3, and is together with the vinegar modifier a foundation for promising secure cryptographic schemes. The general construction of the public and secret keys are almost identical to the scheme presented in Chapter 3, however, there are some small changes.

Denote by v the number of vinegar variables added. The central map F in a

HFEv scheme is defined in the following way,

$$F\colon\ \mathbb{E}\times\mathbb{F}^v\to\mathbb{E}$$
$$F(X,v_1,...,v_v)=\sum_{0\le i\le j}^{q^i+q^j\le D}\alpha_{i,j}X^{q^i+q^j}+\sum_{0\le k}^{q^k\le D}\beta_i(v_1,...,v_v)X^{q^k}+\gamma(v_1,...,v_v). \tag{4.1}$$

The coefficients $\alpha_{i,j}$ behave in the same way as in (3.2), while the β_i and γ are changed slightly. In the HFEv setting, β_i and γ map these additional v vinegar variables in $\mathbb{F}^v$ into elements of $\mathbb{E}$. An important feature is that the β_i are all affine maps, while γ is at most quadratic, in $(v_1,...,v_v)$. This ensures that the resulting multivariate map still only contain quadratic terms. This modification improves the security of the scheme, which will be further discussed later in this chapter. Also, this central map generates an overdetermined set of equations in its multivariate representation. The idea is that the signer of a message can chose random values for these vinegar variables, and still arrive at a consistent system of equations with high probability. As a result, the modification is at no extra computational cost for the signer, while providing improved security.

The affine transformation T from equation (3.5) also needs to be revamped to account for the v extra vinegar variables. One arrives at the new affine transformation

$$T\colon\ \mathbb{F}^{n+v}\to\mathbb{F}^{n+v}\,. \tag{4.2}$$

The objective of this transformation is to mix the vinegar variables with the other variables, effectively hiding the structure of this modified central map F.

The minus modifier is a very simple modification, but with many benefits. It actually makes both the Gröbner basis attacks and linearization attacks mentioned in Section 2.3 much more difficult. To remove a equations from the public key, the previous affine transformation S of rank n is replaced with a map

$$S\colon\ \mathbb{F}^n\to\mathbb{F}^{n-a}\,. \tag{4.3}$$

Normally, one chooses S of maximal rank, i.e. S being surjective. The public key in a HFEv- construction is composed in a similar way as in standard HFE

$$P\colon\ \mathbb{F}^{n+v}\to\mathbb{F}^{n-a}$$
$$P=S\circ\phi^{-1}\circ F\circ(\phi\times\mathrm{id}_v)\circ T, \tag{4.4}$$

where id_v denotes the identity map from $\mathbb{F}^v\to\mathbb{F}^v$. Effectively, the HFEv- scheme

produces a set of $n - a$ equations in $n + v$ variables of the following form,

$$\begin{aligned} p_1(\underline{x}, \underline{v}) &= \sum_{1 \leq i \leq j \leq n} \alpha_{1,i,j} x_i x_j + \sum_{i=1}^{n} \beta_{1,i}(\underline{v}) x_i + \gamma_1(\underline{v}), \\ &\vdots \\ p_{n-a}(\underline{x}, \underline{v}) &= \sum_{1 \leq i \leq j \leq n} \alpha_{n-a,i,j} x_i x_j + \sum_{i=1}^{n} \beta_{n-a,i}(\underline{v}) x_i + \gamma_{n-a}(\underline{v}), \end{aligned} \tag{4.5}$$

where $\underline{x} = (x_1, ..., x_n)$ and $\underline{v} = (v_1 ..., v_v)$. Notice that this new set of polynomials is almost identical to the set produced in (3.6), where the only differences are the coefficients β and γ no longer being constants, and some of the equations have been removed. For use in encryption, this scheme is very impractical, as the mapping is even further from being a bijective map. However, it does serve as a great tool for signature schemes.

4.2 HFEv- for signatures

Before a presentation of the two signature schemes Gui and GeMSS is given, the common building block for these schemes is introduced. Essentially, every signature scheme based on HFEv- incorporate the following algorithm. Let $(\mathrm{pk}, \mathrm{sk}) = \big(P, (S, F, T)\big)$ be a key pair as described above. Generating a signature for a message M consists of finding an element $\sigma \in \mathbb{F}^{n-a}$ such that $P(\sigma) = \mathcal{H}(M||r)$ for some hash function $\mathcal{H}$. Since P is not necessarily surjective, a random *salt*, denoted r, of length s is introduced. If a given hash value does not have a signature, simply choose a new value for the random salt, and try

again. The standard HFEv- signature generation is presented in Algorithm 3.

Algorithm 3: HFEv- signature generation

Input: sk = (S, F, T), message M
Output: (σ, r) such that $P(\sigma) = \mathcal{H}(M||r)$

$v \in_R \mathbb{F}^v$, where $\in_R$ denotes random sampling
repeat
  $r \in_R \mathbb{F}^s$
  $h \leftarrow \mathcal{H}(M||r)$
  Find $x \in F^n$ such that $S(x) = h$
  $X \leftarrow \phi(x)$
  $F'_V(\cdot) \leftarrow F'(\cdot, v)$, insert vinegar variables in the central map, resulting in a map strictly over $\mathbb{E}$
  $\mathcal{Y} \leftarrow \{ Y \in \mathbb{E} \mid F'_V(Y) = X\}$ (by e.g. Berlekamp's algorithm)
until $|\mathcal{Y}| > 0$
$Y \in_R \mathcal{Y}$
$y = \phi^{-1}(Y)$
$\sigma = T^{-1}(y||v)$
return (σ, r)

To verify a given signature (σ, r) for a message M, simply compute $P(\sigma)$ and $\mathcal{H}(M||r)$ and compare the two. If

$$P(\sigma) = \mathcal{H}(M||r), \tag{4.6}$$

the signature is accepted. For correctness of the scheme, let $(\mathrm{pk}, \mathrm{sk}) = (P, (S, F, T))$ be a valid key pair, and assume (σ, r) is a legitimate output from the signature algorithm for the message M. We note

$$P(\sigma) = S \circ F \circ T(\sigma) = S \circ F(y||v) = S \circ {F_V}^1(y) = S(x) = h = \mathcal{H}(M||r). \tag{4.7}$$

The reader is referred to Section 3.1 for an extensive example on how similar calculations are performed. This basic idea is the foundation of signature schemes based on HFEv-. For the remainder of the paper, $P^{-1}(y)$ refers to the action of finding elements x satisfying $P(x) = y$, and is only computationally feasible for the entity holding the secret key. Now that the HFEv- construction has been presented, some analysis regarding the most effective attacks against the scheme will be given.

[1] F_V denotes the multivariate map F where the v vinegar variables have been inserted.

4.3 Attacks on HFEv-

There are two main types of attacks against the HFEv- signature scheme, which are defined as follows.

Definition 37. Let $(\mathrm{pk}, \mathrm{sk}) = (P, (S, F, T))$ be a valid key pair. One has the following two families of attacks.

- Key recovery attack: Reproduce the secret key sk from the public key pk.
- Signature forgery: For some message M, find elements $(\sigma, r) \in (\mathbb{F}^{n+v} \times \mathbb{F}^s)$ such that $P(\sigma) = \mathcal{H}(M || \, r)$.

Clearly, a successful key recovery attack is stronger than a signature forgery attack, since the recovered key permits the attacker to create signatures using the signature generation algorithm.

As mentioned in Chapter 3, Gröbner basis techniques have been used to break HFE systems [15]. However, the systems broken by Faugère et al. did not make use of the minus and vinegar modifications. As it turns out, these tweaks greatly increase the difficulty in algebraically solving the system of equations generated. Other notable attacks include the so-called MinRank attack, proposed in [16]. The authors were also able to defeat the classic HFE construction without the use of the minus and vinegar modifiers. However, with the modifications, also the MinRank approach seems to struggle. In the following, a brief introduction to several attacks will be presented.

4.3.1 Key recovery attacks

MinRank, Kipnis-Shamir attack To explain the basic idea of the MinRank attack it is easier to restrict the problem to a classic HFE instance, i.e. without the minus and vinegar modification. In [16], the authors translated an HFE system into an instance of the so-called MinRank problem, which is defined as follows.

Definition 38 (The MinRank problem)**.** Let n, r, l be positive integers, and let $M_0, M_1, ..., M_l$ be square matrices of size n with coefficients in $\mathbb{E}$. The MinRank problem is then to find a vector $(u_1, ..., u_l)$ where $u_i \in \mathbb{E}$ such that

$$\mathrm{Rank}\left(\sum_{i=1}^{l} u_i M_i - M_0\right) \leq r. \tag{4.8}$$

In other words, the MinRank problem is to find a linear combination of some matrices yielding a matrix bounded above by some rank. The key strategy in this attack is to represent the maps S^{-1}, T and the public key P as maps over the

extension field $\mathbb{E}$, denoted S^{*-1}, T^* and P^* respectively. As we are in the original HFE case, both a and v are zero, meaning S is indeed invertible. For simplicity we restrict S^{-1} and T to linear maps over $\mathbb{F}^n$, which mean they can be expressed over $\mathbb{E}$ as

$$S^{*-1}(X) = \sum_{i=0}^{n-1} s_i X^{q^i} \in \mathbb{E}[X], \tag{4.9}$$

$$T^*(X) = \sum_{i=0}^{n-1} t_i X^{q^i} \in \mathbb{E}[X]. \tag{4.10}$$

The important observation is that the central univariate HFE polynomial, F can be written as a quadratic form over $\mathbb{E}$. For simplicity, assume that F has the following form

$$F(X) = \sum_{i,j}^{q^i+q^j \leq D} \alpha_{i,j} X^{q^i+q^j} \in \mathbb{E}[X], \tag{4.11}$$

i.e. we neglect the linear and constant terms. With F in this shape, the equations in the public key only consist of quadratic terms. Similarly, the public equations P can be expressed over $\mathbb{E}$ as

$$P^*(X) = S^*(F(T^*(X))) = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} p_{i,j} X^{q^i+q^j} \in \mathbb{E}[X]. \tag{4.12}$$

By introducing the vector $\underline{X} = (X^{q^0}, X^{q^1}, ..., X^{q^{n-1}})$ one can rewrite F and P^* as

$$F(X) = \underline{X} A \underline{X}^\top, \tag{4.13}$$

$$P^*(X) = \underline{X} P \underline{X}^\top, \tag{4.14}$$

for some square coefficient matrices $A = [a_{i,j}]$ and $P = [p_{i,j}]$ with elements from $\mathbb{E}$. One can immediately note that the rank of the matrix A is heavily dependent on D. As the degree of F is bounded by D, any pair (i, j) satisfying $q^i + q^j > D$ implies $a_{i,j} = 0$. This means that the matrix A is of the form

$$A = \left[\begin{array}{cc:c} & & 0 \\ & A' & 0 \\ \hdashline 0 & 0 & 0 \end{array}\right], \tag{4.15}$$

where A' is of size $(r \times r)$ with $r = \left\lfloor \log_q(D-1) \right\rfloor + 1$. This is called the rank of the map F. It follows that the rank of A is bounded from above by r. By rewriting (4.12) to

$$S^{*-1}(P^*(X)) = F(T^*(X)), \tag{4.16}$$

it follows from the rank condition of F that the matrix representation of $S^{*-1}(P^*(\cdot))$ does not exceed rank r. Denote the left hand side of (4.16) as $\bar{P}$. It follows that

$$\begin{aligned}\bar{P}(X) = S^{*-1}(P^*(X)) =& \sum_{i=0}^{n-1} s_i \left(\sum_{j=0}^{n-1} \sum_{k=0}^{n-1} \left(p_{i,j} X^{q^i+q^j} \right) \right)^{q^k} \\ =& \sum_{i=0}^{n-1} s_i \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} p_{i,j}^{q^k} \left(X^{q^i+q^j} \right)^{q^k} \\ =& \underline{X} \left(\sum_{i=0}^{n-1} s_i \mathcal{P}_i \right) \underline{X}^\top. \end{aligned} \tag{4.17}$$

We have hence rewritten $\bar{P}$ to be a quadratic form, where the matrices $\mathcal{P}_i$ are calculated by cyclically rotating the columns and rows of P by i steps, and raising the entries to the q^i-th power. We know that the rank of the linear combination of the matrices $\mathcal{P}_i$ cannot exceed r, since the right hand side of (4.12) has rank bounded above by r. This formulation is exactly the MinRank problem stated above. By solving an instance of MinRank, the values s_i are found, which indeed reveals the transformation S in the secret key. The authors of [16] proposed to solve the given MinRank instance by modeling the rank condition as a set of multivariate equations. Let $s = (s_0, ..., s_{n-1})$ be an unknown solution of the MinRank instance considered and denote

$$\mathcal{P} = \sum_{i=0}^{n-1} s_i \mathcal{P}_i. \tag{4.18}$$

Since $\mathrm{Rank}(\mathcal{P}) \leq r$, we have

$$\mathrm{Dim}\left(\mathrm{Ker}\left(\mathcal{P}\right)\right) \geq n - r. \tag{4.19}$$

Hence, there exist (at least) $n - r$ linearly independent vectors in the kernel of $\mathcal{P}$ in $\mathbb{E}^n$. Denote these vectors $x^{(1)}, ..., x^{(n-r)}$. As the vectors have n coordinates, one can fix the first $n - r$ coordinates to random values, and still expect to find $n - r$ linearly independent vectors in $\mathrm{Ker}(\mathcal{P})$, see [7]. This setup gives rise to the following set of equations

$$\mathcal{P} x^{(i)} = 0, \tag{4.20}$$

for $1 \leq i \leq n-r$. For each i, each kernel equation (4.20) gives rise to n equations in r variables, in addition to the n unknowns $(s_0, ..., s_{n-1})$. All in all, this system consists of $n(n-r)$ equations in $n+r(n-r)$ variables. Since $r << n$, the resulting system is very overdetermined. The relinearization technique explained in Section 2.3 can then be applied and quickly find a solution to the MinRank problem. It is

important to note that the $(s_0, ..., s_{n-1})$ is one possible solution to the MinRank instance, but not necessarily the only solution. The authors of [22] address this issue and propose several methods to overcome this. Hence, the transformation S^{-1} can be found. A similar argument can be made for retrieving T. Hence, the secret key is practically exposed, by the obvious relation

$$\begin{aligned} & P = S \circ \phi^{-1} \circ F \circ (\phi \times \mathrm{id}_v) \circ T \\ \implies & F = (\phi \times \mathrm{id}_v) \circ S^{-1} \circ P \circ T^{-1} \circ \phi^{-1}. \end{aligned} \tag{4.21}$$

We arrive at an attack where the complexity is dominated by the solution of the MinRank problem.

Finding a solution to the MinRank problem can also be formulated slightly differently, in what is known as the minors modeling [23]. Instead of searching for a set of vectors in the kernel of $\mathcal{P}$, they utilized the determinant in order to model the rank condition of $\mathcal{P}$, or more precisely, they utilized the minors of $\mathcal{P}$ which are defined as follows.

Definition 39. A *minor* of a matrix A is the determinant of some square submatrix of A. The *degree* of the minor is the size of said submatrix.

Clearly, if the rank of the matrix $\mathcal{P}$ is bounded above by r, any minor of $\mathcal{P}$ of degree higher than r is zero. This means solving an instance of the MinRank problem is equivalent to finding a vector $(s_0, ..., s_{n-1})$ such that all minors of degree $(r + 1)$ of $\mathcal{P}$ are zero. The number of distinct minors of degree $(r + 1)$ in a square matrix of size n is simply

$$m = \binom{n}{r+1}^2. \tag{4.22}$$

This can be found by observing that any submatrix of size $(r + 1)$ is simply a selection of $(r + 1)$ rows and $(r + 1)$ columns, among a total of n possibilities each. Under the constraint that all of these determinants should be zero, one arrives at a system of m multivariate polynomial equations in n variables. The highly overdetermined set of equations is in practice easier to solve than the set produced using the kernel approach.

This MinRank attack is focused on the HFE construction without any modifiers, but it can be extended to HFEv-. In introducing the vinegar variables and removing some equations, the rank of the matrix representation of the central map increases. In [13] the authors showed that adding v vinegar variables increases the rank of the map by v. Similarly, in [24] the authors showed that removing a equations leads to a rank increase of a. Summarized, the rank of an HFEv- map is

bounded from above by $r + a + v$. This leads to a complexity [25]

$$C_{\text{MinRank,Classical}} \sim \binom{n + r + a + v}{r + a + v}^{\omega}, \tag{4.23}$$

where ω still is the linear algebra constant introduced in Section 2.3. There are also other known key-recovery attacks for variants of HFE which are built on the minors modeling idea, see [26].

4.3.2 Signature forgery attacks

Brute force In order to find elements $(\sigma, r) \in \mathbb{F}^{n+v} \times \mathbb{F}^s$ such that $P(\sigma) = \mathcal{H}(M||r) \in \mathbb{F}^{n-a}$ for some message M, one first rewrites the equations as $P(\sigma) - \mathcal{H}(M||r) = 0$. Then, by fixing $v + a$ coordinates of σ, one arrives at a system of $(n - a)$ equations in $(n - a)$ variables. By evaluating the $(n - a)$ equations for every possible combination, one arrives at a valid signature with high probability [8]. If no combination is valid, simply chose a new random salt r, and try again. Each equation in $(n-a)$ variables can at most have $(n-a)(n-a-1)/2$ quadratic terms[2]. Since $\mathbb{F}$ is of characteristic q, and each polynomial is seemingly random, the probability of each coefficient being zero is $1/q$. The average polynomial has then

$$\tau(n - a, q) = \frac{(n - a)(n - a - 1)}{2} \frac{q - 1}{q} \tag{4.24}$$

quadratic terms. Clearly, if an attempted σ fails to satisfy the first equation in the system, it is discarded. Meaning, there are many choices of σ that will not be evaluated in all the polynomials. Denote h_i the i-th component of the vector $\mathcal{H}(M||r)$. The probability of $p_i(\sigma) - h_i = 0$ is $1/q$, assuming that p_i randomly distributes elements into $\mathbb{F}$. The expected number of equations needed to be evaluated for each choice of σ is then

$$\sum_{i=1}^{n-a} q^{1-i} < q, \tag{4.25}$$

hence the complexity of a brute force attack is approximately

$$\begin{aligned} C_{\text{BruteForce,Classical}} &\sim \cdot\, q \cdot q^{n-a} \tau(n - a, q) \cdot (n - a) \\ &\sim \cdot\, q^{n-a+1} (n - a)^3. \end{aligned} \tag{4.26}$$

In [27], the authors introduce a fast exhaustive search algorithm for polynomial systems over $\mathrm{GF}(2)$. Again, the idea to first fix $(v + a)$ variables is used. Using

[2]Over fields of characteristic 2, we need to subtract $(n - a)$ from this formula, as $x^2 = x$. The magnitude still remains of the same order.

the fast exhaustive search on the resulting system has a complexity of

$$C_{\text{FastExhaustiveSearch,Classical}} \sim \log_2(n-a) \cdot 2^{n-a+2}. \tag{4.27}$$

In the quantum world, Grover's algorithm [28] speeds up a brute force approach. In [29] the authors provide a way to solve a system of $(n-a)$ polynomial equations in $(n-a)$ variables over $\mathrm{GF}(2)$ using

$$C_{\text{BruteForce,Quantum}} \sim 2 \cdot 2^{(n-a)/2} \cdot (n-a)^3, \tag{4.28}$$

quantum bit operations. By fixing $(v+a)$ components of σ, and then applying Grover's algorithm we arrive at almost quadratic speedup compared to classical search algorithms.

Algebraic attack by Gröbner bases Gröbner basis attacks also attempt to solve the system of equations $P(\sigma) - \mathcal{H}(M \,||r) = 0$ for σ. The idea to first fix $(v+a)$ coordinates of σ is used. Then, by using Gröbner basis techniques, e.g. F4 or F5, the system of equations can be reduced to the shape presented in the Shape Lemma (33), and the solutions are easily found. It has been experimentally found that systems of equations originating from an HFEv- construction are more easily solved than random systems of equations [30, 13]. As mentioned in Section 2.3, the complexity of finding a Gröbner basis heavily depends on row-reducing a large matrix. The size of this matrix heavily depends on the so-called degree of regularity of a system of equations, denoted d_{reg}. As shown in [13] the degree of regularity for HFEv- systems is upper bounded by

$$d_{\text{reg}} \leq \begin{cases} \frac{(q-1)(r+a+v-1)}{2} + 2 & \text{if } q \text{ is even and } (r+a) \text{ is odd,} \\ \frac{(q-1)(r+a+v)}{2} + 2 & \text{otherwise.} \end{cases} \tag{4.29}$$

The parameter r is still the rank parameter defined above, $r = \left\lfloor \log_q(D-1) \right\rfloor + 1$. More recently, a lower bound for the degree of regularity was experimentally showed to be

$$d_{\text{reg}} = \left\lfloor \frac{a+r+v+7}{3} \right\rfloor, \tag{4.30}$$

when restricted to the case $\mathbb{F} = \mathrm{GF}(2)$, see [31]. As this was experimentally found for fairly low values (a, r, v), one should be cautious to extrapolate these results to arbitrary values. However, as a safety precaution it is reasonable to rather utilize this lower bound than (4.29). As presented in Section 2.3, the most computationally challenging part of finding a Gröbner basis is to row reduce matrices of enormous sizes. When restricted to the case $q = 2$, the complexity of this attack is

$$C_{\text{Gröbner basis ,Classical}} \sim 3 \cdot \left(\binom{n-a}{d_{\text{reg}}}^{\omega} \right) \binom{n-a}{2}, \tag{4.31}$$

see [32].

Very recently, in December 2017, a quantum algorithm restricted to solving polynomial equations over $\mathrm{GF}(2)$ named Quantum Boolean Solve was presented [33]. At the time of publishing, this algorithm was the fastest algorithm for solving polynomial equations over $\mathrm{GF}(2)$. The algorithm is a Gröbner basis algorithm combined with Grover's algorithm. For a system of $(n-a)$ equations in $(n-a)$ variables, this probabilistic algorithm uses

$$C_{\mathrm{QuantumBooleanSolve}} \sim \mathcal{O}(2^{0.462\cdot(n-a)}) \tag{4.32}$$

quantum bit operations. The complexity analysis of this algorithm is crucial to determining secure parameters for an HFEv- scheme to be able to withstand quantum computer attacks.

As previously mentioned, both Gui and GeMSS are built on the HFEv- construction. However, due to a peculiar result called the birthday paradox, some extra layers of protection are needed in order to avoid huge public keys. Both Gui and GeMSS incorporate what is called Feistel-Patarin rounds. In the following section, the Feistel-Patarin construction will be presented, together with a surprisingly effective attack.

4.4 Feistel-Patarin Construction

In a group of 23 individuals, there is a probability of around $1/2$ that two persons in the group share a birthday. This fascinating result is called the birthday paradox [34] and has implications for the security of many cryptographic systems, including HFEv- schemes.

Since the signature generation in HFEv- produces a vector $\sigma \in \mathbb{F}^{n-a}$ there is a maximum total of q^{n-a} different signatures. By the birthday paradox, one can expect to find two messages with the same signature after approximately $q^{(n-a)/2}$ trials. Hence, by precomputing $P(x)$ for $q^{(n-a)/2}$ different values x, and $\mathcal{H}(M||r)$ for $q^{(n-a)/2}$ different choices of r, one should find a valid message-signature pair with high probability. One way to prevent such attacks is to simply use a very high value for $(n-a)$. However, this would result in a very large public key and slow performance in general. Instead, by using the idea of iterated Feistel ciphers, commonly used in block ciphers, this birthday attack can be countered. In [35], the author examines the so-called Feistel-Patarin construction, which can be defined as follows.

Definition 40 (Feistel-Patarin construction)**.** Let P be some invertible map[3] of the form $\mathbb{F}^n \to \mathbb{F}^n$, and denote by k some repetition factor. For simplicity, let $\mathbb{F} =$

[3] At least invertible for some entity.

$\mathrm{GF}(2)$, and denote by $\oplus$ bitwise XOR. Let $h = \mathcal{H}(M||r) = (h_1, ..., h_k) \in \mathbb{F}^{n \cdot k}$ be the hash of some message M to be signed, partitioned into k vectors of equal length n. Denote $S_0 \in \mathbb{F}^n$ some predetermined vector, typically the zero vector. Then, $(\sigma, r) \in \mathbb{F}^n \times \mathbb{F}^s$ is a signature for the message M, where σ is produced in the following way,

$$\sigma = P^{-1}(h_k \oplus P^{-1}(h_{k-1} \oplus P^{-1}(...P^{-1}(h_1 \oplus S_0)))). \tag{4.33}$$

Verifying a signature is done by computing

$$\begin{aligned} (h_1, ..., h_k) &= \mathcal{H}(M||r), \\ S_0' &= h_1 \oplus P(...P(h_{k-1} \oplus P(h_k \oplus P(\sigma)))), \end{aligned} \tag{4.34}$$

and comparing $S_0' = S_0$. If they are equal, the signature is accepted.

One important observation is that verifying a signature is not of the form $P(\sigma) = \mathcal{H}(M||r)$ presented in Equation (4.6), but rather $f(\sigma, \mathcal{H}(M||r)) = 0$, where f is referring to the process in Equation (4.34). This means that the classical birthday attack no longer works, one is not able to simply pre-compute and store a large table of values for $P(x)$ and $\mathcal{H}(M|r)$ for arbitrary values x and r and search for a collision. Under the assumption that this $f\colon \mathbb{F}^n \to \mathbb{F}^n$ is evenly distributed, the probability of a random value σ satisfying $f(\sigma, \mathcal{H}(M||r)) = 0$ is about $1/2^n$, meaning the expected number of guesses before a success approaches 2^n. However, the classic Meet-in-the-middle attack [36] can be applied to the Feistel-Patarin construction, resulting in what is called the Feistel-Patarin attack.

Feistel-Patarin attack To illustrate the attack, fix $k = 2$ to arrive at the following signature generation,

$$\begin{aligned} (h_1, h_2) &= \mathcal{H}(M||r), \\ \sigma &= P^{-1}(h_2 \oplus P^{-1}(h_1 \ \oplus S_0)). \end{aligned} \tag{4.35}$$

The signature of M is then (σ, r). Signature verification is done by computing

$$\begin{aligned} (h_1, h_2) &= \mathcal{H}(M||r), \\ S_0' &= h_1 \oplus P(h_2 \oplus P(\sigma)), \end{aligned} \tag{4.36}$$

and comparing S_0' with the predetermined vector S_0. The signature is accepted if the two are equal. In this illustration, we use $S_0 = 0^n$, the zero vector. Finding a valid message-signature pair is done by the following. Assume that P uniformly maps $\mathbb{F}^n$ into $\mathbb{F}^n$, and is a reasonably fast-evaluated map. In order to forge a signature on the message M, calculate $h^{(j)} = \mathcal{H}(M||r^{(j)})$ for a random salt $r^{(j)}$,

where j is in some index set. Then, split each hash value into two equal parts $h_1^{(j)}$ and $h_2^{(j)}$ such that $h^{(j)} = (h_1^{(j)}||h_2^{(j)})$. Pre-compute a total of $2^{n/3}$ tuples of the form

$$(M^{(j)}, S_1^{(j)}, S_1^{(j)} \oplus h_2^{(j)}), \tag{4.37}$$

where $S_1^{(j)}$ is any value satisfying $P(S_1^{(j)}) = h_1^{(j)}$. Note that finding these tuples is more costly than simply doing $2^{n/3}$ evaluations of the map P. Under the assumption that P uniformly distributes $\mathbb{F}^n$ onto itself, obtaining $2^{n/3}$ distinct values for the first $2^{n/3}$ random inputs is extremely unlikely, which again is due to the birthday paradox. We need approximately $2^{2n/3}$ evaluations of P until $2^{n/3}$ distinct values are found, i.e. we have to pick on average $2^{2n/3}$ different random salts $r^{(j)}$.

Next, we search a value $\sigma \in \mathbb{F}^n$ such that $P(\sigma) = S_1^{(j)} \oplus h_2^{(j)}$ for *any* j. Since we have a total of $2^{n/3}$ potential values on the right hand side of the equation, the probability of a randomly sampled σ satisfying this is $2^{-2n/3}$. Hence, the expected number of trials before finding such a σ is $2^{2n/3}$. After finding σ satisfying $P(\sigma) = S_1^{(\alpha)} \oplus h_2^{(\alpha)}$ for some index α, we have indeed created a message-signature pair $\left(M, (\sigma, r^{(\alpha)})\right)$ without the knowledge of the private key. To verify that the signature is indeed accepted, we first calculate

$$(h_1^{(\alpha)}||h_2^{(\alpha)}) = \mathcal{H}(M||r^{(\alpha)}), \tag{4.38}$$

and insert it into the verification algorithm,

$$\begin{aligned} S_0' =& h_1^{(\alpha)} \oplus P(h_2^{(\alpha)} \oplus P(\sigma)) \\ =& h_1^{(\alpha)} \oplus P(h_2^{(\alpha)} \oplus S_1^{(\alpha)} \oplus h_2^{(\alpha)}) \\ =& h_1^{(\alpha)} \oplus P(S_1^{(\alpha)}) \\ =& h_1^{(\alpha)} \oplus h_1^{(\alpha)} \\ =& 0^n. \end{aligned} \tag{4.39}$$

The signature is therefore accepted. By using approximately $2^{2n/3}$ evaluations of the public map P, one is able to forge a message-signature pair. This is a much better attack than the one mentioned above, needing approximately 2^n evaluations of P. Keep in mind that this attack is *completely generic*, as it does not take into account the form of the underlying function P. The attack can be extended to an arbitrarily large k, where one can expect to find valid message-signature pairs after $\sim 2^{\frac{kn}{k+1}}$ evaluations of P. This analysis shows that the parameters n and k must be chosen with utmost care.

In the case where the said map is an HFEv- map, one can approximate the computations needed to evaluate the map. As stated in (4.24), each polynomial in

the public key has on average $\tau(n)$ quadratic terms. This means there are a total of $\tau(n)$ multiplications for each polynomial. As the number of quadratic terms is vastly superior to the number of linear terms, one can approximate the total number of additions to be of the same order, $\tau(n)$. Hence, a total of

$$2 \cdot \tau(n) \cdot (n - a) \tag{4.40}$$

field operations are needed to evaluate the public map. This leads to a complexity of the Feistel-Patarin Attack of approximately

$$C_{\text{Feistel-Patarin Attack}} \sim (n - a) \cdot \tau(n) \cdot 2^{(n-a) \cdot \frac{k}{k+1}}. \tag{4.41}$$

It would be interesting to formulate a similar attack in the quantum setting.

Now that the most common attacks to the HFEv- construction have been introduced, we are ready to discuss the two NIST proposals Gui and GeMSS and assess their security levels.

Chapter 5

Gui and GeMSS

In this chapter, a presentation of the schemes Gui and GeMSS will finally be given. An analysis of the chosen parameter sets for the two contributions is presented. In order to assess the security levels of the schemes, the NIST security definition for digital signatures as well as the NIST security strength categories are introduced.

In the case of Gui and GeMSS, both the private keys and public keys are practically the same. They have some different naming conventions, so to remove any confusion, for the rest of this paper the following naming convention will be introduced. In the public key

$$P = S \circ \phi^{-1} \circ F \circ (\phi \times \mathrm{id}_v) \circ T, \tag{5.1}$$

defined for a standard HFEv- scheme, we rename the affine transformations, and use the following convention,

$$P = L \circ \phi^{-1} \circ F \circ (\phi \times \mathrm{id}_v) \circ R. \tag{5.2}$$

Here, L indicates the left affine transformation, while R stands for right. In contrast to the description of HFEv- given above, Gui and GeMSS are implemented with invertible L transformations. The reason for this is a speed up in the signature generation, at the cost of a slightly larger private key. This means that the public key is not strictly

$$P = L \circ \phi^{-1} \circ F \circ (\phi \times \mathrm{id}_v) \circ R, \tag{5.3}$$

but rather the $(n - a)$ first components of this P. This calls for a tweak in the construction of the signature and will be detailed later in this chapter. To further ease notation, we remove any confusion by using these naming conventions for the parameters of both Gui and GeMSS.

Parameters and notation

- $\mathbb{F} = \mathrm{GF}(2)$, finite field of 2 elements.
- $\mathbb{E} = \mathrm{GF}(2^n)$, extension of degree n of $\mathbb{F}$.
- ϕ, canonical isomorphism between $\mathbb{F}^n$ and $\mathbb{E}$.
- D, degree of the central polynomial.
- a, number of minus equations.
- v, number of vinegar variables.
- k, Feistel-Patarin repetition factor.
- r, a salt vector in $\mathbb{F}^s$.
- $\mathcal{H}$, hash function of the form $\mathcal{H}\colon \{0,1\}^* \to \{0,1\}^\kappa$, where κ is the output size of $\mathcal{H}$.

With the common terminology given, we are ready to present Gui.

5.1 Gui

Key Generation Given the parameters mentioned above, the key generation algorithm operates as follows. The private key is simply generated by sampling random affine transformations in $\mathbb{F}^n$ and $\mathbb{F}^{n+v}$ until finding two that are invertible, respectively L and R. The central map F is also sampled randomly, respecting the shape as presented in (4.1). Next, the intermediate public key P' is calculated,

$$P' = L \circ \phi^{-1} \circ F \circ (\phi \times \mathrm{id}_v) \circ R. \tag{5.4}$$

The public key P is then extracted by removing the a last entries in P'. As the signature generation in standard HFEv- relies on finding preimages, it is more convenient to store the inverses of L and R in the private key. Hence, L^{-1} and R^{-1} are calculated and stored in the private key together with the central map F. Denote by $\mathcal{M}_\eta(\mathbb{F})$ the set of all square matrices of size η with elements from the field $\mathbb{F}$. The function $\mathrm{Aff}(M, c)$ returns the affine transformation $M \cdot x + c$, and $\mathrm{HFEv}(\mathbb{F}, n, D, v)$ returns a random central map in accordance with the HFEv central map defined in (4.1). The key generation algorithm is presented in Algorithm 4.

Algorithm 4: Gui key generation

Input: $(\mathbb{F}, n, D, a, v)$
Output: Public key pk, Secret key sk

do
  $M_L \in_R \mathcal{M}_n(\mathbb{F})$
until M_L *is invertible*
$c_L \in_R \mathbb{F}^n$
$L \leftarrow \mathrm{Aff}(M_L, c_L)$
do
  $M_R \in_R \mathcal{M}_{n+v}(\mathbb{F})$
until M_R *is invertible*
$c_R \in_R \mathbb{F}^{n+v}$
$R \leftarrow \mathrm{Aff}(M_R, c_R)$
$F \leftarrow \mathrm{HFEv}(\mathbb{F}, n, D, v)$
$P' \leftarrow L \circ \phi^{-1} \circ F \circ (\phi \times \mathrm{id}_v) \circ R$
$\mathrm{sk} \leftarrow (L^{-1}, F, R^{-1})$
$\mathrm{pk} \leftarrow$ the a first components of P'
return (pk, sk)

Signature Generation For a given message M we wish to produce a signature σ. In the Gui scheme, the underlying hash function $\mathcal{H}$ is part of the SHA2 hash family. A random salt r of length s is introduced in the digest to be signed, in the following way,

$$h' = (\mathcal{H}(\mathcal{H}(M)||r) \,||\, \mathcal{H}(\mathcal{H}(\mathcal{H}(M)||r))||...||\, \mathcal{H}^l(\mathcal{H}(M)||r)). \tag{5.5}$$

There are several reasons for using such a random salt. As previously mentioned, it serves as a safeguard for the signature generation, in the sense that there are hash values that do not have valid signatures for a given key pair. There are also other reasons for using a random modification, and some motivation can be found in [37]. The length of h' is limited by the constraint

$$k(n-a) \leq \left|h'\right| = l \cdot \kappa. \tag{5.6}$$

We choose the smallest such l satisfying this. For given parameters (k, n, a, κ) one can explicitly calculate l as

$$l = \left\lceil \frac{k(n-a)}{\kappa} \right\rceil. \tag{5.7}$$

The vector h' is then partitioned into k vectors, $(h_1, ..., h_k)$, each of $(n-a)$ elements from $\mathbb{F}$. The last $l \cdot \kappa - k(n-a)$ elements of h are discarded. This

preprocessing of the message is summarized in Algorithm 5.

Algorithm 5: Message preprocessing and partitioning

Function `preprocessMessage(`M, r`)`
  $l \leftarrow \lceil k(n-a)/\kappa \rceil$
  $h' \leftarrow (\mathcal{H}(\mathcal{H}(M)||r) \,||...||\, \mathcal{H}^l(\mathcal{H}(M)||r))$
  for *i = 1 to k* **do**
    $h_i = \left(h'_{(i-1)\cdot(n-a)+1}, ..., h'_{i\cdot(n-a)}\right) \in \mathbb{F}^{n-a}$
return $(h_1, ..., h_k)$

Both the signature generation and verification use this preprocessing algorithm. In the signature generation process, some random elements $\gamma^{(i)} \in \mathbb{F}^a$ for $1 \leq i \leq k$ are introduced. These elements serve as dummy elements to find preimages of the transformation L. In the description of HFEv- given in Chapter 4 the map L was of the form $\mathbb{F}^n \to \mathbb{F}^{n-a}$. As previously mentioned, Gui makes use of an invertible L from $\mathbb{F}^n$ to $\mathbb{F}^n$. Instead of looking for preimages of a certain element $h \in F^{n-a}$, we append a random vector of length a to h and simply apply L^{-1}. The authors of Gui argue that this approach speeds up signature generation, at the cost of a larger private key. The signature generation process is presented in Algorithm

Algorithm 6: Gui signature generation

Input: $\text{sk} = (L^{-1}, F, R^{-1})$, $M \in \mathbb{F}^*$, repetition factor k
Output: $\sigma \in \mathbb{F}^{(n-a)+k(a+v)+s}$

1 $S_0 \leftarrow 0^{n-a}$, initialization vector comprising of $n-a$ zeros.
2 $\gamma^{(1)}, ..., \gamma^{(k)} \in_R (\mathbb{F}^a)^k$
3 $v^{(1)}, ..., v^{(k)} \in_R (\mathbb{F}^v)^k$
4 $r \in_R \mathbb{F}^s$
5 $(h_1, ..., h_k) \leftarrow \texttt{preprocessMessage}(M, r)$.
6 **for** $i \leftarrow 1$ **to** k **do**
7 $\quad x \leftarrow L^{-1}(h_i \oplus S_{i-1}, \gamma_1^{(i)}, ..., \gamma_a^{(i)})$
8 $\quad X \leftarrow \phi(x) \in \mathbb{E}$
6. 9 $\quad F_V(\cdot) \leftarrow F(\cdot, v^{(i)})$
10 $\quad \hat{Y} \leftarrow \gcd(F_V(Y) - X, Y^{2^n} - Y)$, by Berlekamp's algorithm
11 $\quad$ **if** $\text{Deg}(\hat{Y}) == 1$ **then**
12 $\quad\quad Y \leftarrow \text{root}(\hat{Y})$, pick the single root of $\hat{Y}$
13 $\quad$ **else**
14 $\quad\quad$ **break** and **jump to** line 4
15 $\quad y \leftarrow \phi^{-1}(Y) \in \mathbb{F}^n$
16 $\quad z \leftarrow R^{-1}(y,\ v_1^{(i)}, ..., v_v^{(i)}) \in \mathbb{F}^{n+v}$,
17 $\quad S_i \leftarrow (z_1, ..., z_{n-a})$
18 $\quad \zeta_i \leftarrow (z_{n-a+1}, ..., z_{n+v})$.
19 **return** $\sigma = (S_k||\zeta_k||...||\zeta_1||r) \in \mathbb{F}^{(n-a)+k(a+v)+s}$

Note that in line 11 of Algorithm 6 they chose to only accept a linear polynomial $\hat{Y}$, as this speeds up the root finding step and the signature generation. Also note that the salt r is appended to the calculated signature, yielding the self-contained signature σ.

Signature Verification The algorithm for verifying a message-signature pair (M, σ) is presented in Algorithm 7.

Algorithm 7: GUI signature verification

Input: $\mathrm{pk} = P$, $M \in \mathbb{F}^*$, $\sigma = (S_k||\zeta_k||...||\zeta_1||r) \in \mathbb{F}^{(n-a)+k(a+v)+s}$
Output: Approve or Reject

$(h_1, ..., h_k) \leftarrow \texttt{preprocessMessage}(M, r)$
for $i \leftarrow k$ **to** 1 **do**
  $w \leftarrow P(S_i||\zeta_i) \in \mathbb{F}^{n-a}$
  $S_{i-1} \leftarrow w \oplus h_i$
if $S_0 == 0^{n-a}$ **then**
  return Approve
else
  return Reject

Signature Correctness For signature correctness, we reintroduce the intermediate public key $P' = L \circ \phi^{-1} \circ F \circ (\phi \times \mathrm{id}_v) \circ R$, i.e. the full transformation where the last a elements of the public key are included. We note for every iteration i in the signature verification,

$$\begin{aligned} w' = P'(S_i||\zeta_i) =& L \circ \phi^{-1} \circ F \circ (\phi \times \mathrm{id}_v) \circ R(S_i||\ \zeta_i) \\ =& L \circ \phi^{-1} \circ F \circ (\phi \times \mathrm{id}_v)(y, v_1^i, ..., v_v^i) \\ =& L \circ \phi^{-1} \circ F_V(Y) \\ =& L \circ \phi^{-1}(X) \\ =& L(x) \\ =& (h_i \oplus S_{i-1}, \gamma_1^i, ..., \gamma_a^i). \end{aligned} \tag{5.8}$$

Then by removing the last a terms of w', we get

$$w = P(S_i||\zeta_i) = h_i \oplus S_{i-1}. \tag{5.9}$$

Then we have indeed

$$w \oplus h_i = h_i \oplus S_{i-1} \oplus h_i = S_{i-1}. \tag{5.10}$$

This shows that in every iteration i in the verification algorithm, we are able to retrieve S_{i-1}. As S_k is indeed given, by recursion S_0 is retrieved.

In particular, the inventors chose three parameter sets for Gui, named Gui-184, Gui-312, Gui-448, presented in Table 5.1.

Table 5.1: Parameter sets for Gui.

	$\mathbb{F}$	n	D	a	v	k	κ
Gui-184	2	184	33	16	16	2	256
Gui-312	2	312	129	24	20	2	384
Gui-448	2	448	513	32	28	2	512

These parameter sets were determined by analyzing the complexity of the most common attacks to multivariate cryptographic systems, to be able to satisfy certain security levels. Estimates of the attacks from Chapter 3 are presented in Section 5.3.

5.2 GeMSS

GeMSS shares some design rationales with Gui. However, there are some differences.

Key Generation The key generation in GeMSS is very similar to the one in Gui, but they use a slightly different approach of sampling random invertible matrices for the transformations L and R. Instead of trial and error, they produce invertible matrices by utilizing the LU decomposition. They generate a lower and an upper triangular matrix, with ones on the diagonal to ensure that the matrices are invertible. The remaining values are chosen randomly. Then, the product of these matrices is indeed invertible, without using any trial and error. However, there are invertible matrices that cannot be produced in this manner, meaning the method is biased toward certain matrices. The authors argue that this approach does not impact the security of the scheme, as the number of unattainable matrices is sufficiently small. We define $\in_{\mathrm{LU}}$ as the process of randomly sampling a matrix using the described LU decomposition technique. The resulting private and public keys are then created in the same way as explained above, and presented in Algorithm 8.

Algorithm 8: GeMSS key generation

Input: $(\mathbb{F}, n, D, a, v)$
Output: Public key pk, Secret key sk

$M_L \in_{\mathrm{LU}} \mathcal{M}_n(\mathbb{F})$
$c_L \in_R \mathbb{F}^n$
$L \leftarrow \mathrm{Aff}(M_L, c_L)$
$M_R \in_{\mathrm{LU}} \mathcal{M}_{n+v}(\mathbb{F})$
$c_R \in_R \mathbb{F}^{n+v}$
$R \leftarrow \mathrm{Aff}(M_R, c_R)$
$F \leftarrow \mathrm{HFEv}(\mathbb{F}, n, D, v)$
$P' \leftarrow L \circ \phi^{-1} \circ F \circ (\phi \times \mathrm{id}_v) \circ R$
$\mathrm{pk} \leftarrow$ the a first components of P'
$\mathrm{sk} \leftarrow (L^{-1}, F, R^{-1})$
return $(\mathrm{pk}, \mathrm{sk})$

Signature Generation The signature generation of GeMSS is similar to Gui, however, they chose to preprocess the message M slightly differently. In GeMSS, the random salt is not utilized before hashing. In the implementation of GeMSS, the salt serves the same purpose as the dummy vectors γ^i in Gui, namely as a means to use the inverse of L [1]. The underlying hash function used in GeMSS is

[1] In [3] the authors also present a version of their algorithm utilizing a salt in somewhat the same manner as Gui, but they have chosen not to implement this version for reasons regarding efficiency.

SHA3. Signature generation is presented in Algorithm 9.

Algorithm 9: GeMSS signature generation

Input: $\text{sk} = (L^{-1}, F, R^{-1}), M$
Output: $\sigma = S_k \in \mathbb{F}^{(n-a)+k(a+v)}$

1 $S_0 \leftarrow 0^{n-a}$, initialization vector comprising of $n-a$ zeros.
2 $H_0 \leftarrow \text{SHA3}(M)$
3 **for** $i \leftarrow 1$ **to** k **do**
4 $\quad h_i \leftarrow$ first $n-a$ elements of the binary representation of H_{i-1}
5 $\quad \gamma \in_R \mathbb{F}^a$
6 $\quad x \leftarrow L^{-1}(h_i \oplus S_{i-1}||\gamma)$
7 $\quad X \leftarrow \phi(x) \in \mathbb{E}$
8 $\quad v \in_R \mathbb{F}^v$
9 $\quad F_V(\cdot) \leftarrow F(\cdot, v)$
10 $\quad \hat{Y} \leftarrow \gcd(F_V(Y) - X, Y^{2^n} - Y)$
11 $\quad$ **if** $\text{Deg}(\hat{Y}) > 0$ **then**
12 $\quad\quad Y \in_R$ *roots*$(\hat{Y})$, pick a random root of the polynomial
13 $\quad$ **else**
14 $\quad\quad$ **break** and **jump to** line 5
15 $\quad y \leftarrow \phi^{-1}(Y) \in \mathbb{F}^n$
16 $\quad z \leftarrow R^{-1}(y, v_1, ..., v_v)$
17 $\quad S_i \leftarrow (z_1, ..., z_{n-a})$
18 $\quad \zeta_i \leftarrow (z_{n-a+1}, ..., z_{n+v})$
19 $\quad H_i \leftarrow \text{SHA3}(H_{i-1})$
20 **return** $\sigma = (S_k||\zeta_k||...||\zeta_1) \in \mathbb{F}^{(n-a)+k(a+v)}$

Signature Verification The algorithm for verifying a message-signature pair (M, σ) is given in Algorithm 10.

Signature Correctness Showing signature correctness is identical to Gui, and is left as an exercise for the reader.

The inventors chose the following three parameter sets for GeMSS, named GeMSS128, GeMSS192, GeMSS256, presented in Table 5.2.

Algorithm 10: GeMSS signature verification

Input: pk $= P$, $M \in \mathbb{F}^*$, $\sigma = (S_k||\zeta_k||...||\zeta_1) \in \mathbb{F}^{(n-a)+k(a+v)}$
Output: Approve or Reject

$H_0 \leftarrow \text{SHA3}(M)$
for $i \leftarrow 1$ **to** k **do**
    $h_i \leftarrow$ first $n-a$ elements of the binary representation of H_{i-1}
    $H_i \leftarrow \text{SHA3}(H_{i-1})$
for $i \leftarrow k$ **to** 1 **do**
    $w \leftarrow P(S_i||\zeta_i) \in \mathbb{F}^{n-a}$
    $S_{i-1} \leftarrow w \oplus h_i$
if $S_0 == 0^{n-a}$ **then**
    return Approve
else
    return Reject

Table 5.2: Parameter sets for GeMSS.

	$\mathbb{F}$	n	D	a	v	k	κ
GeMSS128	2	174	513	12	12	4	128
GeMSS192	2	265	513	22	20	4	192
GeMSS256	2	354	513	30	33	4	256

5.3 Comparison and security assessment

Comparing different cryptosystems is hard, as they each have their own strengths and weaknesses. Before a comparison of their security is given, we first present the sizes of the keys and signatures for the different parameter sets of Gui and GeMSS.

Table 5.3: Key and signature sizes of all given parameter sets of Gui and GeMSS.

	public key (KB)	private key (KB)	signature size (bit)
Gui-184	416.3	19.1	360
GeMSS128	352.18	13.9	384
Gui-312	1955.1	59.3	504
GeMSS192	1273.6	38.5	704
Gui-448	5789.2	155.9	664
GeMSS256	3519.3	80	832

As shown in Table 5.3, GeMSS has slightly larger signatures than Gui for all parameter sets. This is simply because of the repetition factor chosen in each scheme, where Gui chooses to use $k = 2$ for all its parameter sets, while GeMSS choose $k = 4$. Nevertheless, both schemes supply incredibly short signatures.

In order to assess their security levels, NIST has proposed six security categories. These six categories are chosen to be somewhat equivalent to a key search on AES and collision search of SHA3 for different instances of these. The quantum security categories are determined by a factor MAXDEPTH introduced by NIST. This MAXDEPTH parameter is an estimate of the maximum circuit depth of a quantum computer, where they propose the three plausible values $2^{40}, 2^{64}$ and 2^{96}. For further details regarding the security categories, see [38]. For the different values of MAXDEPTH one arrives at the security categories presented in Table 5.4.

Table 5.4: NIST security categories.

Category	$\log_2$ classical complexity	$\log_2$ quantum complexity MAXDEPTH $= (2^{40}, 2^{64}, 2^{96})$
I	143	$(130, 106, 74)$
II	146	
III	207	$(193, 169, 137)$
IV	210	
V	272	$(258, 234, 202)$
VI	274	

Now that the Gui and GeMSS signature schemes have been presented, some analysis of the chosen parameter sets will be given. The inventors have chosen parameter sets for Gui and GeMSS based on the most common attacks to the HFEv- scheme, presented in Section 4.3. In Table 5.5, the complexities of these attacks are explicitly presented for each parameter set of both Gui and GeMSS. Keep in mind that most of the attacks presented in Section 4.3 were attacks to the general HFEv- construction. As both Gui and GeMSS incorporate a repetition factor, the attacks must be scaled accordingly.

As shown in the table, both Gui and GeMSS have certain weaknesses versus the Quantum Boolean Solve algorithm, even when assuming the *lowest* bound of the quantum security. As the algorithm was presented in December 2017, *after* the deadline for the submissions to NIST was passed [2], some more consideration needs to be made. The table also shows that the classical Gröbner basis attack is

[2] Actually, the authors of GeMSS are the same as the authors of [33], and they do mention Quantum Boolean Solve in their proposal of GeMSS.

Table 5.5: Comparison of the $\log_2$ complexities of the attacks presented in Section 4.3 against Gui and GeMSS for the various parameter sets. From left to right, MinRank attack (MR), Brute Force (BF), Fast Exhaustive Search (FES), Gröbner basis (GB), Feistel-Patarin attack (FPA), Quantum Boolean Solve (QBS). The best classical attacks are printed in bold.

Category	MR	BF	FES	GB	FPA	QBS
I, II						
Gui-184	286	191	174	157	**132**	79
GeMSS128	260	186	169	**149**	**149**	77
III, IV						
Gui-312	423	313	294	222	**214**	134
GeMSS192	400	269	250	**211**	216	114
V, VI						
Gui-448	583	443	422	**293**	301	193
GeMSS256	550	351	331	**281**	282	152

the most effective attack for most of the parameter sets for both Gui and GeMSS. In fact, the Gröbner basis complexity was the determining factor when choosing most of the parameters for both schemes. This further motivates the future study of Gröbner basis algorithms, both in the quantum and classical setting.

The most interesting point regarding these attacks is that the generic Feistel-Patarin attack described in Section 4.4 breaks the proposed security level of Gui-184, as it does not satisfy the Category **I** and **II** requirements. The main reason Gui fails to withstand this attack, while GeMSS counters it, is the choice of the repetition factor k. If Gui-184 used a higher repetition factor, e.g. 3 or 4, the Feistel-Patarin attack would have a complexity of approximately

$$C_{\text{Feistel-Patarin Attack}}(k = 3) \sim \mathcal{O}(2^{146}), \tag{5.11}$$

$$C_{\text{Feistel-Patarin Attack}}(k = 4) \sim \mathcal{O}(2^{154}), \tag{5.12}$$

where both would satisfy the Category **I** and **II** requirement. Of course, increasing the repetition factor leads to a decreased efficiency of the scheme, as well as an increase in the size of the signature, by 32 or 64 bits respectively. In the documentation for Gui [2], the authors have not mentioned this attack, even though the attack itself is not particularly new. In April 2018, Ward Beullens, PhD candidate at COSIC, KU Leuven, Belgium, made a comment in the official NIST Post-Quantum Cryptography forum addressing this issue [39]. Two months later, the authors of Gui made an official comment regarding this attack [40]. They proposed to change the repetition factor to $k = 3$ in the Gui-184 scheme while leaving the other parameter sets of Gui unchanged. They claim that this modification will

increase the run time of the signature generation by 50%. By modifying the implementation submitted by the authors, the run times have indeed been investigated, and are presented in Table 5.6. The run times for key generation, signature generation, and signature verification are shown for all the parameter sets for both Gui and GeMSS, including a version of Gui-184 where the repetition factor has been increased to 3 and 4. The average run times for all algorithms were found after performing 10 runs.

Table 5.6: Gui and GeMSS run times for the different parameter sets presented above, measured on a MacBook Pro 13-inch Early 2015 model having 2.7 GHz Intel Core i5 processor, 8GB memory.

Category, name	key generation (s)	generate signature (s)	verify (ms)
I, II			
Gui-184 $(k = 2)$	0.319	0.0070	0.049
Gui-184 $(k = 3)$	0.311	0.0211	0.108
Gui-184 $(k = 4)$	0.316	0.0612	0.075
GeMSS128	0.046	0.422	0.283
III, IV			
Gui-312	1.3965	0.23038	0.249
GeMSS192	0.2506	0.9993	0.376
V, VI			
Gui-448	7.4141	3.8197	0.441
GeMSS256	0.5763	1.2578	0.707

The interesting observation is the increase in run time for signing a message. The run time almost tripled from $k = 2$ to $k = 3$, and increased by nearly a ninefold from $k = 2$ to $k = 4$, which is quite contradictory of the claim made by the authors. One possible explanation for this might be that the root finding step over the extension field, which is the most time-consuming step of the signature generation, needs to be repeated several times. The probability of finding a single root in this step (line 10-12 in Algorithm 6) is about $1/e$, see [8], where e is Euler's number. This means that on average, in order to find a unique root k times, the root finding algorithm is run approximately e^k times. As this step is the most computationally expensive during signature generation, the run time of the signing algorithm can be approximated as

$$T_{\mathrm{Sig}}(k) = e^k \cdot T_{\mathrm{RootFind}} + T_0, \tag{5.13}$$

where T_{RootFind} denotes the run time of the root finding step, and T_0 denotes other calculations. For increasing k, the value T_0 becomes less and less relevant. Some

more experiments were conducted for even higher values of k, to investigate if the exponential run times in k were observed. The results are presented in Figure 5.1, with the chosen $T_{\mathrm{RootFind}} = 0.002559\,\mathrm{s}$ based on the timing results for signature generation in Gui-184 using $k = 1$.

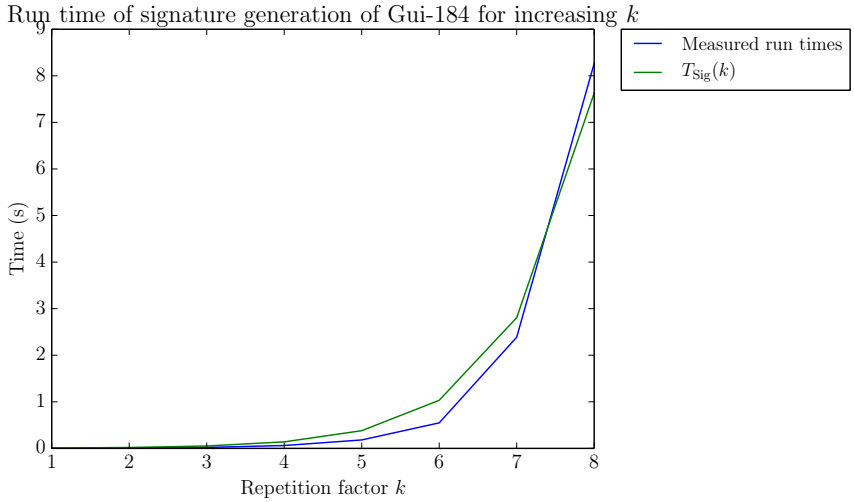

Figure 5.1: Run times for signature generation with respect to the repetition factor k for Gui-184. The figure also shows (5.13), with T_{RootFind} approximated by signing a message using $k = 1$.

Under this argument, the threefold increase in run time for each increase in k seems to be reasonable, and seems supported by the experimental results given. I am curious why the authors of Gui claimed an increase in run time of 50%. I also attempted to modify the submitted code for Gui-312 and Gui-448 to further investigate the run times, but the signatures produced ended up not being accepted by the verification algorithm.

Chapter 6

Closing remarks and further work

The signature schemes Gui and GeMSS have several advantages and limitations. One of the main advantages is the short length of the signatures produced. As a comparison, both ECDSA and DSA signatures are approximately 4λ bits where λ is the given security level, while Gui and GeMSS offer signatures of sizes close to 2λ. Another advantage of these schemes is the simplicity of the signature verification. As this is merely an evaluation of equations, it can be performed extremely rapidly, as shown in Table 5.6. Also, having small private keys, the schemes are suitable for use in small devices with scarce resources, e.g. smartcards. One of the main disadvantages is the size of the public key. For the highest security levels presented, the public keys are of order $4 - 5$ MB, being almost a thousand times as large as an RSA public key at the same classical security level.

The implementation of Gui-184 as proposed by the authors failed to meet the claimed security level, as it was broken by the Feistel-Patarin attack. The solution proposed by the authors to increase the repetition factor k makes said attack infeasible, and the modified Gui-184 is able to withstand all the classical attacks mentioned in this thesis. The authors claimed that the new parameter choice k would increase the run time of signature generation by 50%. However, experimental results and analytical arguments presented in this thesis suggest a time increase of a factor e, Euler's number. The authors of Gui might have some clever trick to circumvent this efficiency drop. Nevertheless, the run times of signature generation for the parameter sets Gui-184 still outperform GeMSS128. Exchanging ideas between GeMSS and Gui seems to be a good approach to find the best possible cryptosystem, as the fast key generation of GeMSS paired with the fast signature generation of Gui might serve as a great tool in post-quantum cryptography.

Bibliography

[1] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26 (5):1484–1509, Oct 1997.

[2] Jintai Ding, Ming-Shing Chen, Albrecht Petzoldt, Dieter Schmidt, and Bo-Yin Yang. Gui documentation, 2017. `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-1-Submissions`.

[3] A. Casanova, J.-C. Faugère, G. Macario-Rat, J. Patarin, L. Perret, and J. Ryckeghem. Gemss: A great multivariate short signature, 2017. `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-1-Submissions`.

[4] David A Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra (Undergraduate Texts in Mathematics)*. Springer, 2015.

[5] E. R. Berlekamp. Factoring polynomials over finite fields. *Bell System Technical Journal*, 46(8):1853–1859, oct 1967.

[6] Eberhard Becker, Teo Mora, Maria Grazia Marinari, and Carlo Traverso. The shape of the shape lemma. In *Proceedings of the international symposium on Symbolic and algebraic computation - ISSAC '94*. ACM Press, 1994.

[7] Jean-Charles Faugère, Françoise Levy-dit Vehel, and Ludovic Perret. *Cryptanalysis of MinRank*, pages 280–296. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-85174-5.

[8] Giordano Fusco and Eric Bach. Phase transition of multivariate polynomial systems. In Jin-Yi Cai, S. Barry Cooper, and Hong Zhu, editors, *Theory and Applications of Models of Computation*, pages 632–645. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-72504-6.

[9] Jean-Charles Faugére. A new efficient algorithm for computing gröbner bases (f4). *Journal of Pure and Applied Algebra*, 139(1-3):61–88, jun 1999.

[10] Jean Charles Faugère. A new efficient algorithm for computing gröbner bases without reduction to zero (f5). In *Proceedings of the 2002 international symposium on Symbolic and algebraic computation - ISSAC '02*. ACM Press, 2002.

[11] J.C. Faugère, P. Gianni, D. Lazard, and T. Mora. Efficient computation of zero-dimensional gröbner bases by change of ordering. *Journal of Symbolic Computation*, 16(4):329–344, oct 1993.

[12] Jean-Charles Faugère and Chenqi Mou. Fast algorithm for change of ordering of zero-dimensional gröbner bases with sparse multiplication matrices. In *Proceedings of the 36th international symposium on Symbolic and algebraic computation - ISSAC '11*. ACM Press, 2011.

[13] Jintai Ding and Bo-Yin Yang. Degree of regularity for hfev and hfev-. In Philippe Gaborit, editor, *Post-Quantum Cryptography*, pages 52–66. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-38616-9.

[14] François Le Gall. Algebraic complexity theory and matrix multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, ISSAC '14, pages 23–23. ACM, 2014. ISBN 978-1-4503-2501-1.

[15] Jean-Charles Faugère and Antoine Joux. Algebraic cryptanalysis of hidden field equation (HFE) cryptosystems using gröbner bases. In *Advances in Cryptology - CRYPTO 2003*, pages 44–60. Springer Berlin Heidelberg, 2003.

[16] Aviad Kipnis and Adi Shamir. Cryptanalysis of the hfe public key cryptosystem by relinearization. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '99, pages 19–30. Springer-Verlag, 1999. ISBN 3-540-66347-9.

[17] Gwénolé Ars, Jean-Charles Faugère, Hideki Imai, Mitsuru Kawazoe, and Makoto Sugita. *Comparison Between XL and Gröbner basis Algorithms*, pages 338–353. Springer Berlin Heidelberg, 2004. ISBN 978-3-540-30539-2.

[18] Joan Daemen and Vincent Rijmen. *The Design of Rijndael*. Springer Berlin Heidelberg, 2002. doi: 10.1007/978-3-662-04722-4.

[19] Tsutomu Matsumoto and Hideki Imai. *Public Quadratic Polynomial-Tuples for Efficient Signature-Verification and Message-Encryption*. Springer Berlin Heidelberg.

[20] Jacques Patarin. Hidden fields equations (HFE) and isomorphisms of polynomials (IP): Two new families of asymmetric algorithms. In *Advances in Cryptology — EUROCRYPT '96*, pages 33–48. Springer Berlin Heidelberg, 1996.

[21] Jacques Patarin, Nicolas Courtois, and Louis Goubin. Flash, a fast multivariate signature algorithm. In David Naccache, editor, *Topics in Cryptology — CT-RSA 2001*, pages 298–307. Springer Berlin Heidelberg, 2001. ISBN 978-3-540-45353-6.

[22] Xin Jiang, Jintai Ding, and Lei Hu. Kipnis-shamir attack on hfe revisited. In Dingyi Pei, Moti Yung, Dongdai Lin, and Chuankun Wu, editors, *Information Security and Cryptology*, pages 399–411. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-79499-8.

[23] Jean-Charles Faugère, Mohab Safey El Din, and Pierre-Jean Spaenlehauer. Computing loci of rank defects of linear matrices using grÖbner bases and applications to cryptology. In *Proceedings of the 2010 International Symposium on Symbolic and Algebraic Computation*, ISSAC '10, pages 257–264. ACM, 2010. ISBN 978-1-4503-0150-3.

[24] Jeremy Vates and Daniel Smith-Tone. Key recovery attack for all parameters of hfe-. In Tanja Lange and Tsuyoshi Takagi, editors, *Post-Quantum Cryptography*, pages 272–288. Springer International Publishing, 2017. ISBN 978-3-319-59879-6.

[25] Albrecht Petzoldt, Ming-Shing Chen, Bo-Yin Yang, Chengdong Tao, and Jintai Ding. Design principles for HFEv- based multivariate signature schemes. In *Advances in Cryptology – ASIACRYPT 2015*, pages 311–334. Springer Berlin Heidelberg, 2015.

[26] Jeremy Vates and Daniel Smith-Tone. Key recovery attack for all parameters of HFE-. In *Post-Quantum Cryptography*, pages 272–288. Springer International Publishing, 2017.

[27] Charles Bouillaguet, Chen-Mou Cheng, Tony (Tung) Chou, Ruben Niederhagen, Adi Shamir, and Bo-Yin Yang. Fast exhaustive search for polynomial systems in $\mathbb{F}_2$. Cryptology ePrint Archive, Report 2010/313, 2010.

[28] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC '96*. ACM Press, 1996.

[29] Bas Westerbaan and Peter Schwabe. Solving binary mq with grover's algorithm. In Claude Carlet, Anwar Hasan, and Vishal Saraswat, editors, *Security, Privacy, and Advanced Cryptography Engineering*, volume 10076 of *Lecture Notes in Computer Science*, pages 303–322. Springer-Verlag Berlin Heidelberg, 2016.

[30] Magali Bardet, Jean-Charles Faugère, and Bruno Salvy. On the complexity of gröbner basis computation of semi-regular overdetermined algebraic equations. 2004.

[31] Jintai Ding, Ray Perlner, Albrecht Petzoldt, and Daniel Smith-Tone. Improved cryptanalysis of hfev- via projection. In Tanja Lange and Rainer Steinwandt, editors, *Post-Quantum Cryptography*, pages 375–395. Springer International Publishing, 2018. ISBN 978-3-319-79063-3.

[32] Albrecht Petzoldt. On the complexity of the hybrid approach on hfev-. *IACR Cryptology ePrint Archive*, 2017:1135, 2017.

[33] Jean-Charles Faugère, Kelsey Horan, Delaram Kahrobaei, Marc Kaplan, Elham Kashefi, and Ludovic Perret. Fast quantum algorithm for solving multivariate quadratic equations. *CoRR*, abs/1712.07211, 2017. URL `http://arxiv.org/abs/1712.07211`.

[34] Marc Girault, Robert Cohen, and 2)Mireille Campana. A generalized birthday attack. In *Lecture Notes in Computer Science*, pages 129–156. Springer Berlin Heidelberg.

[35] Nicolas T. Courtois. Generic attacks and the security of quartz. In Yvo G. Desmedt, editor, *Public Key Cryptography — PKC 2003*, pages 351–364. Springer Berlin Heidelberg, 2002. ISBN 978-3-540-36288-3.

[36] Ralph C. Merkle and Martin E. Hellman. On the security of multiple encryption. *Communications of the ACM*, 24(7):465–467, jul 1981.

[37] Koichi Sakumoto, Taizo Shirai, and Harunaga Hiwatari. On provable security of uov and hfe signature schemes against chosen-message attack. In Bo-Yin Yang, editor, *Post-Quantum Cryptography*, pages 68–82. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-25405-5.

[38] National Institute of Standards and Technology. Submission requirements and evaluation criteria for the post-quantum cryptography standardization process, 2016. https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf.

[39] Ward Beullens. Official comment: Gui, 2018. https://groups.google.com/a/list.nist.gov/forum/#!topic/pqc-forum/8VE6gtFPSH8.

[40] Bo-Yin Yang. Official comment: Gui, 2018. https://groups.google.com/a/list.nist.gov/forum/#!topic/pqc-forum/v7vrS-_jMrA.