

Advances in Information Security 80

Jintai Ding
Albrecht Petzoldt
Dieter S. Schmidt

Multivariate Public Key Cryptosystems

Second Edition



Springer

Advances in Information Security

Volume 80

Series editor

Sushil Jajodia, George Mason University, Fairfax, VA, USA

The purpose of the *Advances in Information Security* book series is to establish the state of the art and set the course for future research in information security. The scope of this series includes not only all aspects of computer, network security, and cryptography, but related areas, such as fault tolerance and software assurance. The series serves as a central source of reference for information security research and developments. The series aims to publish thorough and cohesive overviews on specific topics in Information Security, as well as works that are larger in scope than survey articles and that will contain more detailed background information. The series also provides a single point of coverage of advanced and timely topics and a forum for topics that may not have reached a level of maturity to warrant a comprehensive textbook.


More information about this series at <http://www.springer.com/series/5576>

Jintai Ding • Albrecht Petzoldt • Dieter S. Schmidt

Multivariate Public Key Cryptosystems

Second Edition

 Springer

Jintai Ding 
Department of Mathematical Sciences
University of Cincinnati
Cincinnati, OH, USA

Albrecht Petzoldt
Department of Computer Science
Friedrich-Alexander-Universität
Erlangen-Nürnberg
Erlangen, Germany

Dieter S. Schmidt
Department of Electrical Engineering
and Computer Science
University of Cincinnati
Springboro, OH, USA

ISSN 1568-2633 ISSN 2512-2193 (electronic)
Advances in Information Security
ISBN 978-1-0716-0985-9 ISBN 978-1-0716-0987-3 (eBook)
<https://doi.org/10.1007/978-1-0716-0987-3>

© Springer Science+Business Media LLC, part of Springer Nature 2006, 2020

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Science+Business Media LLC part of Springer Nature.

The registered company address is: 1 New York Plaza, New York, NY 10004, U.S.A.

Preface

For the last three decades, public-key cryptography has completely changed the landscape of our modern communication systems and become an indispensable part of the foundation of our communication networks. In this modern age of global commerce, social networking, remote marketplaces, and cyber-espionage, information security is of critical importance. In addition to the problems of phishing, network intrusion, denial of service attacks, duplicitous crypto, hacking, and plain classical cryptanalysis, there is an additional threat looming just over the horizon: quantum computers.

The Internet and other communication systems rely principally on the Diffie–Hellman key exchange and RSA encryption to provide confidentiality and digital signatures such as DSA or related algorithms to protect the authenticity and integrity of users’ information. The security of these techniques is derived from number-theoretic problems such as integer factorization or the discrete logarithm problem. In 1994, Dr. Peter Shor showed that quantum computers can efficiently solve both of these problems, thus rendering all public-key cryptosystems based on such assumptions insecure. If a sufficiently powerful quantum computer can be built, it will put all modern communication systems into peril, when they use key exchanges or encryption or digital authentication.

In the years since Shor’s discovery, we have seen steady progress in quantum computing technologies. In 2001, Dr. I. Chuang led an effort at IBM that implemented Shor’s algorithm on a 7-qubit quantum computer to factor $15 = 3 * 5$. This demonstrated that Shor’s algorithm actually works. In a very recent presentation, M. Mosca, deputy director of the Institute for Quantum Computing, estimated a 1/7 chance that quantum computers will break RSA-2048 (one of the most secure of the encryption and authentication algorithms currently in use) by 2026. A large international community has emerged to address this issue in the hope that our public-key infrastructure may remain intact by utilizing new **quantum-resistant** primitives. In the academic world, this new science is denoted as “**Post-Quantum Cryptography**” (PQC).

Many cryptographers around the world, both in academia and beyond, have devoted themselves to the search for these alternative public-key systems. In May of

2006, the first International Workshop on Post-Quantum Cryptography (PQCrypto) was organized by the European Network of Excellence for Cryptology (ECRYPT). This major conference has been held regularly.

The need for quantum-resistant public-key cryptosystems has also received attention within the standardization and policy spectrum. The National Institute of Standards and Technology (NIST) has held workshops on post-quantum cryptography and the European Telecommunications Standards Institute (ETSI) has held “Quantum-Safe Cryptography” workshops.

Even intelligence organizations have broken the silence on post-quantum cryptography. In the UK, the Government Communications Headquarters (GCHQ) has publicly acknowledged and published work on quantum-resistant cryptography. In the USA, in August 2015, the National Security Agency (NSA) published a webpage stating

Currently, Suite B cryptographic algorithms are specified by the National Institute of Standards and Technology (NIST) and are used by NSA’s Information Assurance Directorate in solutions approved for protecting classified and unclassified National Security Systems (NSS). Below, we announce preliminary plans for transitioning to **quantum-resistant** algorithms. (<https://web.archive.org/web/20160101091229/>; https://www.nsa.gov/ia/programs/suiteb_cryptography/)

In February 2016, at the 7th PQCrypto Workshop in Japan, NIST announced a call for proposals for quantum-resistant algorithms and in July of 2016 published the first draft of a call for quantum-resistant standards. NIST made a very strong justification for the timeliness of the PQC project.

While in the past it was less clear that large quantum computers are a physical possibility, many scientists now believe it to be merely a significant engineering challenge. Some engineers even predict that within the next 20 or so years, sufficiently large quantum computers will be built to break essentially all public-key schemes currently in use. Historically, it has taken almost two decades to deploy our modern public-key cryptography infrastructure. Therefore, regardless of whether we can estimate the exact time of the arrival of the quantum computing era, we must begin now to prepare our information security systems to be able to resist quantum computing. (<https://csrc.nist.gov/projects/post-quantum-cryptography/>)

Historically, NIST standards have often been endorsed by other standard-setting organizations around the world, and so the development of a post-quantum standard at NIST will set the stage for the maintenance of our e-society in the next generation. Considering all of these facts, it is not surprising that the effort to develop quantum-resistant technologies has become an increasingly central research topic in the area of information security.

Just as current public-key cryptosystems such as the Diffie–Hellman key exchange and RSA encryption relied heavily on number-theoretic advances, new post-quantum cryptosystems rely on a solid understanding of the underlying mathematical hard problems and the related mathematical structures and theories. Nonetheless, this is a relatively new field of study that has brought many new challenging mathematical problems, which are important both theoretically and practically. Research towards developing a new quantum-resistant NIST

cryptographic standard brings a great and unique opportunity and even greater challenges for the mathematical community.

This relatively young nature of post-quantum cryptography presents a particularly alarming situation for NIST, an organization which has based its asymmetric recommendations noticeably on the longevity of public-key cryptosystems. This strategy is less applicable in the quantum-resistant milieu in which most of the longest-lived schemes suffer horrendous parameters to achieve practical security and would require a dramatic overhaul of the public-key infrastructure.

Multivariate public-key cryptography, or MPKC for short, is one of the main families of post-quantum cryptosystems and has increasingly been seen as a possible alternative to classical public-key schemes such as RSA and DSA.

A result from complexity theory states that solving a set of randomly chosen nonlinear multivariate polynomial equations over a finite field is NP-hard. So far, quantum computers have not yet been shown to be able to solve a set of multivariate polynomial equations efficiently, and the consensus is that quantum computers are unlikely to provide an advantage for this type of problem.

In general, a multivariate public-key cryptosystem (MPKC) is a public-key cryptosystem in which the public key is a set of multivariate polynomials $p^{(1)}, \dots, p^{(m)}$ in $\mathbb{F}[x_1, \dots, x_n]$, where \mathbb{F} is a finite field. If Alice wants to send the message $(x'_1, \dots, x'_n) \in \mathbb{F}^n$ to Bob, she looks up Bob's public key, computes $y'_i = p^{(i)}(x'_1, \dots, x'_n)$ for $i = 1, \dots, m$, and sends the encrypted message (y'_1, \dots, y'_m) to Bob. Bob's secret key will be some information about the construction of the polynomials $p^{(i)}$ which enables him to solve the system $p^{(1)}(x_1, \dots, x_n) = y'_1, \dots, p^{(m)}(x_1, \dots, x_n) = y'_m$ for x_1, \dots, x_n . Of course, since Bob, with the help of his secret key, must be able to recover Alice's message efficiently, the polynomial system $p^{(1)}, \dots, p^{(m)}$ must contain some structure. Therefore, the NP-hardness of the multivariate polynomial equation solving problem does not necessarily guarantee the security of practical schemes, though intuitively it does suggest that the more we can make the system $p^{(1)}, \dots, p^{(m)}$ appear to be "random," the more secure the scheme is likely to be.

Research on MPKCs has undergone rapid development in the last two decades, providing many interesting results in designing and attacking the MPKCs. In addition, the study of MPKCs has also resulted in new ideas in solving systems of multivariate polynomial equations over a finite field, a purely mathematical problem that lies in the area of algebraic geometry. This has also attracted a lot of attention.

There are several multivariate schemes submitted to compete in the NIST post-quantum standardization competition. Due to the relatively long history of study and the solid theoretical and experimental support, we believe that some of these schemes are very strong candidates. One example is the Rainbow signature scheme, which is very simple, very efficient, and has very small signatures, although it has relatively large public keys. Rainbow was invented more than 15 years ago by two of the authors of this book and has sustained attacks for all these years without any change of the basic design. Rainbow is now one of the nine signature schemes in the second round of the NIST Post-Quantum standardization process.

Here we would like to point out that developing a new MPKC is a very delicate task. LUOV, a NIST second-round candidate, was broken by the first author and his students by a new subfield differential-algebraic attack.

This book is intended to systematically present the subject matter to a broad audience, including information security experts in industry, computer scientists, and mathematicians. We hope that this book can be used in the following ways: by industry experts as a guide for understanding the basic mathematical structures needed to implement these cryptosystems for practical applications, as a starting point for researchers in both computer science and mathematics looking to explore this exciting new field, or as a textbook for a course in MPKC suitable for beginning graduate students in mathematics or computer science. Due to the above considerations, this book has been written more from the computational perspective, though we have tried to provide the necessary mathematical background.

It should be noted that there are usually several improvements on the schemes that we present, in particular in terms of the efficiency of the computation in both implementation and attacks. However, to keep the size of this book reasonable and to keep the book more focused, we have chosen not to cover some of these details. Instead, we have tried to present the essential ideas, methods, and examples so that a reader will not be distracted by technical details that can be found in the references provided. Nevertheless, for those readers interested in the practical side of the MPKCs, we highly recommend reading through the details in order to discover improvements. Improving the performance of a cryptosystem by even a small factor may not be significant from a mathematical perspective, but can be very important in practice.

Due to the fast development in MPKC, though this book bears the same title as our previous book, we have totally rewritten this book with much more new ideas and research results.

The materials in the book can be used as a text for a year-long course in advanced topics in cryptography or applied algebra, or as a supplementary text for a first course in cryptography. Students with some previous exposure to abstract algebra (groups, rings, fields, and ideals) will be more than well-prepared to read and understand the various topics. For those with a programming background, our relevant software is available for public use at <https://scholar.uc.edu/show/b5644s654>.

The same website can also be reached via the more permanent URL <http://dx.doi.org/10.7945/5sqr-g734>.

This will provide interested readers a starting point to further develop their understanding and computational intuition by experimenting with the software. Those readers new to the field of MPKC will be best served by first reading the introductory chapter (Chap. 2), after which the chapters are written so as to be essentially self-contained. Readers with previous exposure to MPKC may use the text to learn more about a given scheme and as a guide to related articles.

Although it was our intention to include all related references, we apologize to those we have missed. Also, the amount of space devoted to a given topic is not necessarily related to how important we consider it. Rather it is likely due to space

constraints or to maintain the consistency and convenience of the structure and flow of the book.

We plan to maintain at the website given above a PDF-file with the name **Corrections** where we will list corrections to the book. Readers are encouraged to submit their findings to any (or all) authors.

We would like to thank many people who supported our book project, and the list is too long to be added here. Many thanks go to the Taft foundation and to the Department of Mathematical Sciences at the University of Cincinnati for their support. Finally, we would like to thank our families for their constant support and encouragement.

Changes to the Previous Edition

Although this book can be viewed as the second edition of the book with the same title, we fundamentally rewrote it in accordance with the new research and new perspective, which were developed after the publication of the previous book. So, in comparison to the first edition, this book contains several new sections/chapters:

- in Chap. 4 (Hidden Field Equations), a section about the ZHFE encryption scheme and a section about the signature scheme Gui;
- in Chap. 5 (Oil and Vinegar), new techniques to reduce the key sizes of UOV and Rainbow, including the signature scheme LUOV, as well as a section on the efficient key generation of these schemes;
- a new Chap. 6 on the MQ-based identification scheme and the MQDSS signature scheme;
- a new Chap. 7 on the simple matrix encryption scheme and its variants;
- in Chap. 8 (Solving Polynomial Systems), new sections about solving univariate polynomials of high degree (important for the implementation of HFE and variants) and on the solution of over- and underdetermined multivariate systems, which is important, e.g. for the security analysis for UOV. Furthermore, this chapter contains a section about recent work on estimating the degree of regularity of HFE like systems.

On the other hand, we have removed the chapter on Triangular Systems, since we do not think that these schemes will be of great relevance for both practice and future research. Due to the rapidly developing field, we apologize that not all references are included.

Jintai Ding

Cincinnati, OH, USA

Albrecht Petzoldt

Nürnberg, Germany

Dieter S. Schmidt

Springboro, OH, USA

Contents

1	Introduction	1
1.1	Cryptography	1
1.2	Public Key Cryptography	2
1.3	Post-Quantum Cryptography	4
	References	4
2	Multivariate Cryptography	7
2.1	Multivariate Polynomials	7
2.1.1	Matrix Representation	12
2.1.2	Symmetric Matrices Corresponding to a Multivariate Quadratic Polynomial	13
2.2	Construction Methods for MPKC's	14
2.2.1	The Bipolar Construction	14
2.2.2	Mixed Systems	16
2.2.3	IP Based Identification	17
2.2.4	MQ Based Identification	19
2.3	Underlying Problems	19
2.3.1	The MQ Problem	19
2.3.2	The IP Problem	20
2.4	Security and Standard Attacks	20
2.4.1	Security Categories	21
2.5	Advantages and Disadvantages	22
	References	23
3	The Matsumoto-Imai Cryptosystem	25
3.1	The Basic Matsumoto-Imai Cryptosystem	26
3.1.1	MI as an Encryption Scheme	27
3.1.2	MI as a Signature Scheme	28
3.1.3	Degree of the Public Key Components	28
3.1.4	Key Sizes and Efficiency	29
3.1.5	Toy Example	30

3.2	The Linearization Equations Attack	32
3.2.1	Linearization Equations Attack on Matsumoto-Imai	33
3.2.2	Toy Example	38
3.3	Encryption Schemes Based on MI	41
3.3.1	Internal Perturbation	41
3.3.2	Differential Attack on PMI	43
3.3.3	Preventing the Differential Attack and PMI+	45
3.3.4	Toy Example	46
3.4	Signature Schemes Based on MI	50
3.4.1	The Minus Variation and SFLASH	50
3.4.2	Toy Example	51
3.4.3	Differential Attack on SFLASH	53
3.4.4	Preventing the Differential Attack and PFLASH	56
3.4.5	Toy Example	57
	References	59
4	Hidden Field Equations	61
4.1	The Basic HFE Cryptosystem	62
4.1.1	HFE as an Encryption Scheme	63
4.1.2	HFE as a Signature Scheme	63
4.1.3	Key Sizes and Efficiency	64
4.1.4	Toy Example	65
4.2	Attacks on HFE	67
4.2.1	The Direct Attack on HFE	67
4.2.2	Rank Attacks of the Kipnis–Shamir Type	67
4.2.3	Summary of the Security of HFE	71
4.3	Encryption Schemes Based on HFE	72
4.3.1	The IPHFE+ Encryption Scheme	72
4.3.2	Security and Efficiency	73
4.3.3	The ZHFE Encryption Scheme	73
4.3.4	Key Sizes and Efficiency	75
4.3.5	Cryptanalysis of ZHFE	76
4.4	Signature Schemes Based on HFE	76
4.4.1	The HFEv- Signature Scheme	77
4.4.2	Key Sizes and Efficiency	79
4.4.3	Toy Example	79
4.4.4	Security of HFEv-	82
4.4.5	The Gui Signature Scheme	84
4.4.6	Security	86
4.4.7	Key Sizes and Efficiency	86
	References	87
5	Oil and Vinegar	89
5.1	The Oil and Vinegar Signature Scheme	90
5.1.1	Properties of the Central Map	91

5.1.2	Key Sizes and Efficiency	92
5.1.3	Toy Example	92
5.2	The Kipnis–Shamir Attack on Balanced Oil and Vinegar and UOV ..	94
5.2.1	The Case of q Odd	97
5.2.2	Toy Example	99
5.2.3	The Case of q Even	101
5.2.4	Toy Example	104
5.2.5	The Unbalanced Oil and Vinegar Signature Scheme (UOV) ..	109
5.3	Other Attacks on UOV	111
5.3.1	The Direct Attack	111
5.3.2	Practical Parameters	115
5.4	The Rainbow Signature Scheme	116
5.4.1	Key Sizes and Efficiency	118
5.4.2	Toy Example	119
5.5	Attacks on Rainbow	121
5.5.1	The Direct Attack	121
5.5.2	Rank Attacks	122
5.5.3	The MinRank Attack	122
5.5.4	The HighRank Attack	124
5.5.5	Attacks Using the Underlying UOV Structure	125
5.5.6	The Rainbow Band Separation Attack	125
5.5.7	Practical Parameters	129
5.6	Reducing the Public Key Size	130
5.6.1	StructuredUOV	130
5.6.2	The Case of Rainbow	134
5.6.3	Key Sizes and Efficiency	137
5.6.4	LUOV	138
5.6.5	Security	140
5.6.6	Key Sizes and Efficiency	141
5.7	Efficient Key Generation of Rainbow	142
5.7.1	First Step: Compute the Matrices $Q^{(i)}$ of the First Layer	144
5.7.2	Second Step: Compute the Matrices $Q^{(i)}$ of the Second Layer	144
5.7.3	Third Step: Compute the Public Key	145
5.7.4	Efficient Key Generation for StructuredRainbow	145
	References	150
6	MQDSS	153
6.1	The MQ Based Identification Scheme	153
6.1.1	Reducing the Communication Cost	158
6.1.2	Security	158
6.1.3	Key Sizes and Efficiency	159
6.1.4	Toy Example	160
6.2	The Fiat-Shamir Transformation	162
6.2.1	Security Analysis	163

6.3	The MQDSS Signature Scheme	164
6.3.1	Security	167
6.3.2	Key Sizes and Efficiency	167
	References	168
7	The SimpleMatrix Encryption Scheme	169
7.1	The Basic SimpleMatrix Encryption Scheme	170
7.1.1	Key Sizes and Efficiency	171
7.1.2	Toy Example	171
7.2	The Rectangular SimpleMatrix Encryption Scheme	174
7.2.1	Solving the Quadratic Systems	176
7.2.2	Probability of Decryption Failures	176
7.2.3	Key Sizes and Efficiency	176
7.2.4	Toy Example	177
7.3	Attacks on SimpleMatrix	180
7.3.1	Direct Attack	180
7.3.2	Rank Attacks	181
7.3.3	Practical Parameters	182
	References	183
8	Solving Polynomial Systems	185
8.1	History of Solving Polynomial Equations	186
8.1.1	Solving Nonlinear Univariate Polynomial Equations	187
8.1.2	Solving Systems of Multivariate Polynomial Equations	189
8.2	Solving Univariate Polynomials of High Degree	190
8.2.1	Berlekamp's Algorithm	191
8.2.2	Toy Example	192
8.2.3	The Cantor–Zassenhaus Algorithm	194
8.2.4	Toy Example	197
8.3	The XL-Algorithm	198
8.3.1	Toy Example	199
8.4	Gröbner Bases	202
8.4.1	Reduction of Polynomials	203
8.5	Buchberger's Algorithm and F_4	204
8.5.1	Buchberger's Algorithm	204
8.5.2	Improvements of Buchberger's Algorithm	206
8.5.3	Faugère's F_4 -Algorithm	206
8.5.4	Toy Example	208
8.6	Estimating the Degree of Regularity	212
8.6.1	Semi-Regular Systems	212
8.6.2	HFE and Variants	215
8.7	Algorithms for Solving Over- and Underdetermined Systems	225
8.7.1	Solving Overdetermined Systems with $m \sim n^2$	225
8.7.2	Solving Underdetermined Systems with $n \sim m^2$	226
8.7.3	Analysis of the Algorithm	233
8.7.4	Toy Example	234

8.7.5 Solving Underdetermined Systems with $n = \nu m$	240
8.7.6 Toy Example	243
References	246
Software	249
Index	251
List of Toy Examples	253

Notations

Throughout this book, we use the following notation:

$\mathbb{F} = \mathbb{F}_q$	Finite field with q elements
\mathbb{F}^n	Vector space of dimension n over \mathbb{F}
$\mathbb{F}[X]$	Univariate polynomial ring over \mathbb{F}
$\mathbb{F}[x_1, \dots, x_n]$	Multivariate polynomial ring in n variables over \mathbb{F}
$\mathbb{E} = \mathbb{F}_{q^n}$	Degree n extension field of \mathbb{F}
$\Phi : \mathbb{F}^n \rightarrow \mathbb{E}$	Isomorphism between the vector space \mathbb{F}^n and the extension field \mathbb{E}
\mathcal{F}	Central map of a multivariate cryptosystem
$\mathcal{L}, \mathcal{S}, \mathcal{T}$	Linear or affine transformations
\mathcal{P}	Public key of a multivariate cryptosystem
$\mathbf{w} = (w_1, \dots, w_m)$	Ciphertext/message (hash value) to be signed
$\mathbf{z} = (z_1, \dots, z_n)$	Plaintext/signature
G_i	Symmetric matrix representing the homogeneous quadratic part of the i -th component of the central map
H_i	Symmetric matrix representing the homogeneous quadratic part of the i -th component of the public key
\mathcal{DP}	Differential/polar form of a system \mathcal{P} of multivariate quadratic system \mathcal{P}

Moreover, small Latin or Greek letters represent integers or elements of the base field, small **bold** letters stand for vectors, and capital letters represent matrices or elements of an extension field.

List of Algorithms

3.1	Linearization equations attack	38
4.1	Signature generation process of Gui	85
4.2	Signature verification process of Gui	86
5.1	Attack on balanced oil and vinegar (q odd)	98
5.2	Attack on balanced oil and vinegar (q even)	103
5.3	Inversion of the map $\tilde{\mathcal{F}}$	104
5.4	UOV-reconciliation attack	115
5.5	MinRank attack	123
5.6	HighRank attack	125
5.7	Key generation of StructuredUOV	133
5.8	Key generation of StructuredRainbow	138
5.9	Efficient key generation of rainbow	146
5.10	Efficient Key Generation of Structured Rainbow.....	150
8.1	Square-free algorithm.....	191
8.2	Berlekamp's algorithm.....	192
8.3	Distinct degree factoring algorithm (DDF)	195
8.4	Fixed degree factoring algorithm (FDF)	195
8.5	Cantor–Zassenhaus algorithm	196
8.6	XL-Algorithm	198
8.7	Buchberger's algorithm	205
8.8	F_4 -algorithm.....	208
8.9	Reduction algorithm of F_4	208
8.10	Transforming the system \mathcal{P} into a system \mathcal{F} of form (8.8)	232
8.11	Transforming the system \mathcal{P} into a system \mathcal{F} of the form of (8.15)	241
8.12	Solving the system \mathcal{F}	243

List of Figures

Fig. 1.1	Workflow of symmetric cryptography	2
Fig. 1.2	Workflow of public key cryptography	3
Fig. 2.1	General workflow of bipolar schemes	15
Fig. 2.2	The IP based identification scheme	18
Fig. 3.1	Construction of a multivariate BigField scheme	26
Fig. 3.2	Construction of the Matsumoto-Imai cryptosystem	27
Fig. 3.3	Addition and multiplication table of $GF(2^2)$	30
Fig. 3.4	Key generation of PMI	43
Fig. 4.1	Structure of the matrix \mathbf{F}^{*0} for HFE	70
Fig. 4.2	Structure of the matrix \mathbf{F}^{*0} for HFEv-	83
Fig. 4.3	Signature generation process of Gui	85
Fig. 5.1	Quadratic forms of the rainbow central map	117
Fig. 5.2	Alternative key generation of UOV	133
Fig. 5.3	Layout of the matrices M_P , M_Q and M_F	135
Fig. 5.4	Alternative key generation of rainbow	135
Fig. 5.5	Matrices $F^{(i)}$ and $Q^{(i)}$	143
Fig. 5.6	Computing the public key	145
Fig. 5.7	Two layer StructuredRainbow signature scheme	146
Fig. 5.8	Computing the central polynomials of the first layer	147
Fig. 5.9	Computing the central polynomials of the second layer	148
Fig. 5.10	Building the matrix M_Q of StructuredRainbow	149
Fig. 6.1	The MQ based identification scheme (3-pass version)	155
Fig. 6.2	The MQ based identification scheme (5-pass version)	156
Fig. 8.1	Structure of the homogeneous part of the system \mathcal{F}	227
Fig. 8.2	System \mathcal{P} after the First Iteration of the Algorithm	230
Fig. 8.3	System \mathcal{P} after the Second Iteration of the Algorithm	231
Fig. 8.4	System \mathcal{P} after the Third Iteration of the Algorithm	231
Fig. 8.5	Structure of the system \mathcal{F}	241
Fig. 8.6	Structure of the System $\tilde{\mathcal{F}}$	243

List of Tables

Notations and Terminology	xix
Table 2.1 Security categories	22
Table 4.1 Modifications of BigField schemes.....	77
Table 4.2 Parameters and key sizes of Gui	87
Table 5.1 Parameters and key sizes of UOV	116
Table 5.2 Parameters and key sizes of rainbow	130
Table 5.3 Parameters and key sizes of StructuredUOV	133
Table 5.4 Parameters and key sizes of StructuredRainbow	139
Table 5.5 Parameters and key sizes of LUOV.....	142
Table 5.6 Running times of the key generation process of rainbow	149
Table 6.1 Number of Rounds of the MQ Based Identification Scheme.....	159
Table 6.2 Key Sizes and Comm. Cost of the MQ Based ID Scheme	160
Table 6.3 Parameters and key sizes of MQDSS	168
Table 7.1 Parameters and Key Sizes of the SimpleMatrix Scheme	182
Table 7.2 Parameters and Key Sizes of Rectangular SimpleMatrix.....	183
Table 8.1 History of solving (systems of) polynomial equations	186

Chapter 1

Introduction



Abstract This chapter gives a short introduction into the field of cryptography. After defining the protection goals of cryptography, we briefly describe the main cryptographic primitives used to achieve these goals. We describe why the currently used public key cryptosystems become insecure in the presence of quantum computers and thereby motivate the field of post-quantum cryptography. We discuss current research activities in this field and introduce the main families of post-quantum cryptosystems.

1.1 Cryptography

Cryptography is widely seen as the science (or sometimes art) to encrypt messages. While, in the past, this was mainly important for politicians and the military, nowadays cryptography has become an essential part of everyday communication. Besides encrypting messages, cryptographic techniques are used for access control on websites (e.g. email providers, online banking) as well as the authentication and integrity of data sent over the Internet (e.g. software updates).

Using these application scenarios, we can identify the following protection goals of cryptography

- **Confidentiality:** An illegitimate user should not get any information about secret messages.
- **Entity authentication:** Corroboration of an entity's identity.
- **Data authentication:** Corroboration of the origin of data.
- **Data integrity:** Ensuring that information has not been changed by unauthorized entities.

Another important cryptographic protection goal is **Non Repudiation**, which prevents the signer of a digital document or contract from denying his authorship. An important technique to achieve these protection goals are public key cryptosystems.

1.2 Public Key Cryptography

Until the 1970s, the only way to reach the cryptographic goals discussed in the previous section was by means of **symmetric cryptography**. In a symmetric cryptosystem, the two users Alice and Bob exchange a secret key k using a secure channel. In order to send a secret message m to Alice, Bob encrypts his message using the previously shared secret key (thus obtaining a ciphertext $c = \mathcal{E}_k(m)$) and sends the ciphertext c to Alice. Alice uses the secret key k to decrypt the ciphertext c and therefore can read the plain message m . On the other hand, the non-legitimate user Eve can eavesdrop only the encrypted message which will give her (in the case of a secure encryption scheme) no information about the content of the message itself (see Fig. 1.1).

The main problem of symmetric cryptosystems is the exchange of the secret key, which must be performed over a secure channel. **Public Key Cryptography** solves this problem by making use of two types of keys, which are denoted as the private and the public key. These two keys are related via a hard mathematical problem such as integer factorization or the discrete logarithm problem. In fact, computing the public key from the private key is easy, while the inversion of this step should be infeasible.

In order to use a public key cryptosystem, Alice first chooses her private key and computes from it the corresponding public key, which can be published e.g.

Fig. 1.1 Workflow of symmetric cryptography

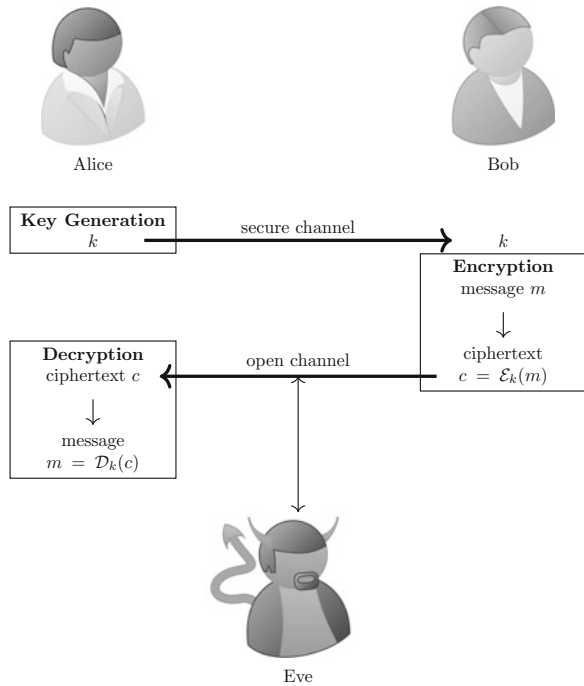
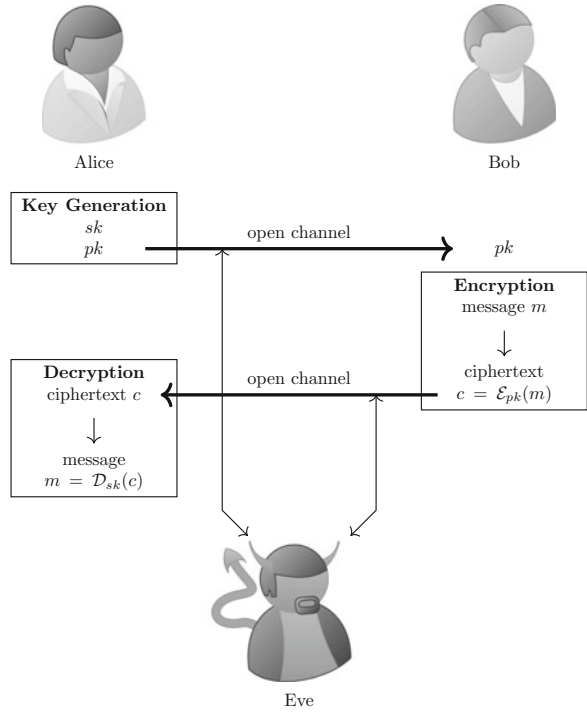


Fig. 1.2 Workflow of public key cryptography



via the Internet. In order to send an encrypted message to Alice, Bob encrypts his message using Alice’s public key and sends the encrypted message to Alice via an open channel. Alice, with the help of her private key, can decrypt the message, while the illegitimate eavesdropper Eve has only access to the encrypted message and the public key (see Fig. 1.2). However, since it is infeasible to compute the private out of the public key, the knowledge of the public key does not help Eve to recover the message.

Additionally to encrypting messages, public key cryptography also enables another very interesting primitive, namely **digital signatures**. In a digital signature scheme, Alice computes a hash value of her message and creates a signature by encrypting this hash value using her private key. After that, she publishes message and signature together. By using Alice’s public key, anybody can check if the decrypted signature is really the hash value of the message. By creating a digital signature for a document d , Alice can therefore prove that she is the author of the document d .

Currently, the most used public key cryptosystems are the factoring based RSA scheme [7] (for encryption and digital signatures) as well as the discrete logarithm based Digital Signature Algorithm (DSA) [5] and its elliptic curve analogue ECDSA.

1.3 Post-Quantum Cryptography

In 1994, Peter Shor proposed an algorithm which solves number theoretic problems such as the integer factorization problem and the discrete logarithm problem in polynomial time on a quantum computer [8]. This algorithm has therefore the potential to solve the mathematical problems underlying RSA and DSA efficiently. As soon as sufficiently large quantum computers are built, the currently used public key cryptosystems will become insecure.

In the last years much research has been done on quantum computing and much progress in building a large scale quantum computer has been achieved. Besides academic institutions, many large companies such as Google and IBM work on this topic and have already created prototypes of quantum computers with up to 50 qubits [2]. In fact, many researchers believe that building a large quantum computer is nowadays mainly an engineering problem, which might be solved within the next 10–15 years.

It is therefore necessary to develop alternatives to the classical cryptographic schemes RSA and DSA, which are not affected by quantum computer attacks and therefore can replace these schemes in a post-quantum era. These are the so-called post-quantum cryptosystems [1]. The importance of research in the area of post-quantum cryptography has been realized by national and international organizations. This includes the EU, which is financing research programs such as PQCRYPTO [4] and SAFEcrypto, and the Japanese Society for the Promotion of Science (JSPS), which is supporting the program CryptoMathCREST [3]. Also the American National Institute for Standards and Technology is preparing to develop standards for post-quantum public key cryptosystems [6]. The main families of post-quantum cryptosystems are:

- lattice-based cryptosystems,
- code-based cryptosystems,
- hash-based signatures,
- isogeny-based key exchange schemes and
- multivariate cryptography, which is the topic of this book.

References

1. D. Bernstein, J. Buchmann, E. Dahmen, *Post-Quantum Cryptography* (Springer, Berlin, Heidelberg, 2008)
2. Futurism.com, MIT Technical Review, 2017. <https://www.technologyreview.com/f/614346/ibms-new-53-qubit-quantum-computer-is-the-most-powerful-machine-you-can-use/>
3. JSPS, Cryptomathcrest website, 2018. <http://crypto.mist.i.u-tokyo.ac.jp/crest/english/index.html>
4. T. Lange, PQCrypto website, 2018. <http://www.pqcrypto.eu/index.html>
5. National Institute of Standards and Technology. Digital signature standard (DSS). FIPS 186 -4, 2013.

6. National Institute of Standards and Technology. Post-quantum cryptography, 2018. <https://csrc.nist.gov/projects/post-quantum-cryptography/>
7. R.L. Rivest, A. Shamir, L.M. Adleman, A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* **21**(2), 120–126 (1978)
8. P. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.* **26**(5), 1484–1509 (1997)

Chapter 2

Multivariate Cryptography



Abstract This chapter gives an overview of the basic concepts of multivariate cryptography. After recalling the basic definitions on (systems of) multivariate polynomials, we present the main construction methods of multivariate public key cryptosystems. We discuss the mathematical problems underlying the security of multivariate cryptography and give an overview of the main attacks against these schemes. Finally, we discuss the advantages and disadvantages of multivariate schemes compared to other (post-quantum) cryptosystems.

In this chapter we give an overview of the basic concepts of multivariate cryptography needed in the following chapters of this book. After recalling the basic definitions of multivariate polynomials in Sect. 2.1, Sect. 2.2 describes the basic construction techniques of multivariate public key cryptosystems. Section 2.3 introduces the mathematical problems on which the security of multivariate cryptosystems is based, whereas Sect. 2.4 gives an overview on the main attacks against multivariate schemes. Finally, Sect. 2.5 discusses the advantages and disadvantages of multivariate schemes compared to other public key cryptosystems.

2.1 Multivariate Polynomials

In this section we recall the basic definitions and introduce notations about multivariate polynomials needed in the later parts of this book.

Definition 2.1 Let $\mathbb{F} = \mathbb{F}_q$ be a finite field with q elements. We define the **ring of multivariate polynomials** in n variables over \mathbb{F} as

$$\mathbb{F}[x_1, \dots, x_n] = \left\{ \sum_{i=1}^s c_i t_{\mathbf{a}_i} \mid s \in \mathbb{N}, \mathbf{a}_i = (a_{i,1}, \dots, a_{i,n}) \in \mathbb{N}_0^n \right\}. \quad (2.1)$$

We call $c_i \in \mathbb{F}$ a **coefficient** and $t_{\mathbf{a}_i} = x_1^{a_{i,1}} x_2^{a_{i,2}} \cdots x_n^{a_{i,n}}$ a **monomial**. The product $c_i t_{\mathbf{a}_i}$ is called a **term**. The set of all monomials in $\mathbb{F}[x_1, \dots, x_n]$ is denoted by T^n .

Definition 2.2 For a polynomial $p = \sum_{i=1}^s c_i t_{\mathbf{a}_i} \in \mathbb{F}[x_1, \dots, x_n]$, we define the **support** of p as the set of all monomials appearing in p , i.e.

$$\text{Supp}(p) = \{t_{\mathbf{a}_i} | c_i \neq 0\}.$$

For a system $\mathcal{P} = (p^{(1)}, \dots, p^{(m)})$, the support of \mathcal{P} is defined as

$$\text{Supp}(\mathcal{P}) = \bigcup_{i=1}^m \text{Supp}(p^{(i)}).$$

Definition 2.3 The **degree** of a monomial $t_{\mathbf{a}} = x_1^{a_1} x_2^{a_2} \cdots x_n^{a_n} \in \mathbb{F}[x_1, \dots, x_n]$ is defined as

$$\deg(t_{\mathbf{a}}) = |\mathbf{a}| = \sum_{j=1}^n a_j. \quad (2.2)$$

The degree of a polynomial $p = \sum_{i=1}^s c_i t_{\mathbf{a}_i}$ is defined as

$$\deg(p) = \max_{i \in \{1, \dots, s\}} \deg(t_{\mathbf{a}_i}) = \max_{i \in \{1, \dots, s\}} |\mathbf{a}_i|. \quad (2.3)$$

Theorem 2.4 Let \mathbb{F} be a finite field with q elements and $d < q$. Then there exist

$$\binom{n+d-1}{d}$$

monomials of degree d in $\mathbb{F}[x_1, \dots, x_n]$. The number of monomials of degree $\leq d$ in $\mathbb{F}[x_1, \dots, x_n]$ is given by

$$\binom{n+d}{d}.$$

Proof

- (1) Number of monomials of degree d : Choose d out of the n elements x_1, \dots, x_n , with possible repetition.
- (2) Number of polynomials of degree $\leq d$: Now we choose d elements from the set $\{x_1, \dots, x_n, 1\}$, again with possible repetition.

□

For $d \geq q$ the number of monomials gets smaller, since we have $x_i^q = x_i$ for $i = 1, \dots, n$. In the important case of $q = 2$ we get

Theorem 2.5 *The number of monomials of degree d in $\text{GF}(2)[x_1, \dots, x_n]$ is given by*

$$\binom{n}{d}.$$

The number of monomials of degree $\leq d$ in $\text{GF}(2)[x_1, \dots, x_n]$ is given by

$$\sum_{i=0}^d \binom{n}{i}.$$

Proof Same as in the proof of Theorem 2.4, only without repetition. \square

As Theorem 2.4 shows, the number of monomials in $\mathbb{F}[x_1, \dots, x_n]$ increases rapidly with higher d . For efficiency reasons, multivariate public key cryptosystems therefore restrict (in most cases) to polynomials of degree 2. In this case we get

Corollary 2.6 *The number of monomials of degree ≤ 2 in $\mathbb{F}[x_1, \dots, x_n]$ is given by*

$$\begin{cases} \frac{(n+1)(n+2)}{2} & \text{if } q > 2 \\ \frac{n(n+1)}{2} + 1 & \text{if } q = 2 \end{cases}.$$

Proof Set $d = 2$ in Theorems 2.4 and 2.5. \square

Corollary 2.6 plays an important role when computing the public key size of a multivariate public key cryptosystem.

Definition 2.7 For a monomial $t_a = x_1^{a_1} x_2^{a_2} \dots x_n^{a_n} \in \mathbb{F}[x_1, \dots, x_n]$ we define

$$\log(t_a) = (a_1, a_2, \dots, a_n) \in \mathbb{N}_0^n. \quad (2.4)$$

Definition 2.8 An **order of monomials** is a complete relationship $\sigma \subset T^n \times T^n$ on T^n . Instead of $(t_1, t_2) \in \sigma$ we write $t_1 >_\sigma t_2$. An order of monomials is called **admissible**, if for all $t_1, t_2, t_3 \in T^n$ we have

- (1) $t_1 >_\sigma 1$,
- (2) $t_1 >_\sigma t_2 \Rightarrow t_1 t_3 >_\sigma t_2 t_3$.

Often used examples for (admissible) orders of monomials are the pure *lexicographical order* (lex), the *graded lexicographical order* (glex) and the *graded reverse lexicographical order* (grevlex).

- In the pure *lexicographical order* we have

$$t_1 >_{\text{lex}} t_2 \Leftrightarrow \text{the first non zero component of } \log(t_1) - \log(t_2) \text{ is greater than 0.}$$

- In the *graded lexicographical order* we have

$$t_1 >_{\text{glex}} t_2 \Leftrightarrow \deg(t_1) > \deg(t_2) \text{ or } \deg(t_1) = \deg(t_2) \wedge t_1 >_{\text{lex}} t_2.$$

- In the *graded reverse lexicographical order* we have

$$t_1 >_{\text{grevlex}} t_2 \Leftrightarrow \deg(t_1) > \deg(t_2)$$

or $\deg(t_1) = \deg(t_2)$ and the last non zero component of $\log(t_1) - \log(t_2)$ is negative.

After having fixed an order of monomials σ , we can give the following definitions:

Definition 2.9 For a multivariate polynomial

$$p(x_1, \dots, x_n) = \sum_{i=1}^s c_i t_{a_i} \in \mathbb{F}[x_1, \dots, x_n],$$

with $t_{a_1} >_{\sigma} t_{a_2} >_{\sigma} \dots >_{\sigma} t_{a_s}$ we define the coefficient vector $\Pi(p)$ by

$$\Pi(p) = (c_1, \dots, c_s) \in \mathbb{F}^s. \quad (2.5)$$

Definition 2.10 For a system \mathcal{P} of m multivariate polynomials¹

$$p^{(1)}(x_1, \dots, x_n) = \sum_{i=1}^s c_i^{(1)} t_{a_i}$$

$$p^{(2)}(x_1, \dots, x_n) = \sum_{i=1}^s c_i^{(2)} t_{a_i}$$

$$\vdots$$

$$p^{(m)}(x_1, \dots, x_n) = \sum_{i=1}^s c_i^{(m)} t_{a_i}$$

with $t_{a_1} >_{\sigma} t_{a_2} >_{\sigma} \dots >_{\sigma} t_{a_s}$, we define the **Macaulay matrix** of \mathcal{P} by

¹Without loss of generality we assume that each of the polynomials of \mathcal{P} contains the same monomials.

$$M_P = (\Pi(p^{(1)}), \dots, \Pi(p^{(m)}))^T = \begin{pmatrix} c_1^{(1)} & c_2^{(1)} & \dots & c_s^{(1)} \\ c_1^{(2)} & c_2^{(2)} & \dots & c_s^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ c_1^{(m)} & c_2^{(m)} & \dots & c_s^{(m)} \end{pmatrix}. \quad (2.6)$$

We get

$$\mathcal{P}(x_1, \dots, x_n) = M_P \cdot (t_{a_1}, \dots, t_{a_s})^T.$$

In the context of multivariate cryptography, we mostly deal with systems of multivariate quadratic polynomials. The number of equations in the system will be denote by m and the number of variables by n . Equation (2.7) shows such a system $\mathcal{P} = (p^{(1)}, \dots, p^{(m)})$ of multivariate quadratic polynomials.

$$\begin{aligned} p^{(1)}(x_1, \dots, x_n) &= \sum_{i=1}^n \sum_{j=i}^n p_{ij}^{(1)} x_i x_j + \sum_{i=1}^n p_i^{(1)} x_i + p_0^{(1)} \\ p^{(2)}(x_1, \dots, x_n) &= \sum_{i=1}^n \sum_{j=i}^n p_{ij}^{(2)} x_i x_j + \sum_{i=1}^n p_i^{(2)} x_i + p_0^{(2)} \\ &\vdots \\ p^{(m)}(x_1, \dots, x_n) &= \sum_{i=1}^n \sum_{j=i}^n p_{ij}^{(m)} x_i x_j + \sum_{i=1}^n p_i^{(m)} x_i + p_0^{(m)} \end{aligned} \quad (2.7)$$

If $m = n$, we call \mathcal{P} a *determined* system. For $m < n$, \mathcal{P} is called *underdetermined* and for $m > n$, we speak of an *overdetermined* system.

For the system \mathcal{P} of (2.7) and the graded lexicographical order of monomials, the Macaulay matrix (see Definition 2.10) has the form

$$M_P = \begin{pmatrix} p_{11}^{(1)} & p_{12}^{(1)} & \dots & p_{nn}^{(1)} & p_1^{(1)} & \dots & p_n^{(1)} & p_0^{(1)} \\ p_{11}^{(2)} & p_{12}^{(2)} & \dots & p_{nn}^{(2)} & p_1^{(2)} & \dots & p_n^{(2)} & p_0^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ p_{11}^{(m)} & p_{12}^{(m)} & \dots & p_{nn}^{(m)} & p_1^{(m)} & \dots & p_n^{(m)} & p_0^{(m)} \end{pmatrix}. \quad (2.8)$$

We get

$$\begin{pmatrix} p^{(1)} \\ p^{(2)} \\ \vdots \\ p^{(m)} \end{pmatrix} (x_1, \dots, x_n) = M_P \cdot (x_1^2, x_1 x_2, \dots, x_n^2, x_1, \dots, x_n, 1)^T. \quad (2.9)$$

2.1.1 Matrix Representation

For a multivariate quadratic system \mathcal{P} , as shown in (2.7), we can write each component $p^{(k)}$ ($k = 1, \dots, m$) as a matrix-vector product using an upper triangular $(n+1) \times (n+1)$ matrix $M P^{(k)}$ of the form

$$M P^{(k)} = \begin{pmatrix} p_{11}^{(k)} & p_{12}^{(k)} & p_{13}^{(k)} & \cdots & p_{1n}^{(k)} & p_1^{(k)} \\ 0 & p_{22}^{(k)} & p_{23}^{(k)} & \cdots & p_{2n}^{(k)} & p_2^{(k)} \\ 0 & 0 & p_{33}^{(k)} & & p_{3n}^{(k)} & p_3^{(k)} \\ \vdots & & \ddots & & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & p_{nn}^{(k)} & p_n^{(k)} \\ 0 & 0 & \cdots & 0 & 0 & p_0^{(k)} \end{pmatrix}. \quad (2.10)$$

We get

$$p^{(k)}(x_1, \dots, x_n) = (x_1, \dots, x_n, 1) \cdot M P^{(k)} \cdot (x_1, \dots, x_n, 1)^T \quad (k = 1, \dots, m). \quad (2.11)$$

Definition 2.11 We call a quadratic system \mathcal{P} **homogeneous quadratic** if it has no linear and constant terms. This means that all coefficients $p_i^{(k)}$ ($i = 0, \dots, n$, $k = 1, \dots, m$) in (2.7) are zero.

We can write a homogeneous quadratic system \mathcal{P} as

$$p^{(k)}(x_1, \dots, x_n) = (x_1, \dots, x_n) \cdot M P^{(k)} \cdot (x_1, \dots, x_n)^T \quad (2.12)$$

with matrices $M P^{(k)}$ ($k = 1, \dots, m$) of the form

$$MP^{(k)} = \begin{pmatrix} p_{11}^{(k)} & p_{12}^{(k)} & p_{13}^{(k)} & \cdots & p_{1n}^{(k)} \\ 0 & p_{22}^{(k)} & p_{23}^{(k)} & \cdots & p_{2n}^{(k)} \\ 0 & 0 & p_{33}^{(k)} & & p_{3n}^{(k)} \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \cdots & 0 & p_{nn}^{(k)} \end{pmatrix} \in \mathbb{F}^{n \times n}. \quad (2.13)$$

2.1.2 Symmetric Matrices Corresponding to a Multivariate Quadratic Polynomial

For some attacks against multivariate public key cryptosystems, we consider the symmetric matrices associated to (the homogeneous part of) the polynomials of the private or public key. These matrices are defined slightly different depending on the characteristic of the underlying field.

For an underlying field of odd characteristic, the symmetric matrix corresponding to a multivariate quadratic polynomial $p \in \mathbb{F}[x_1, \dots, x_n]$ is the $n \times n$ matrix $Q = (q_{ij})$, whose elements are given by

$$q_{ij} = \begin{cases} \text{MonomialCoefficient}(p, x_i^2) & i = j \\ \text{MonomialCoefficient}(p, x_i x_j)/2 & \text{otherwise.} \end{cases}$$

In the case of an underlying field of even characteristic, we can not define the matrix Q as above, since the division by two is not defined. Here, we define first an upper triangular matrix $P = (p_{ij})$ by

$$p_{ij} = \begin{cases} \text{MonomialCoefficient}(p, x_i x_j) & i \leq j \\ 0 & \text{otherwise} \end{cases}$$

and set

$$Q = P + P^T.$$

Note that the matrix Q is a symmetric matrix with zeros on the main diagonal.

Definition 2.12 For a multivariate quadratic system \mathcal{P} , we define the **differential** or **polar form** as

$$\mathcal{DP}(\mathbf{x}, \mathbf{y}) = \mathcal{P}(\mathbf{x} + \mathbf{y}) - \mathcal{P}(\mathbf{x}) - \mathcal{P}(\mathbf{y}) + \mathcal{P}(0).$$

Note that the differential of a multivariate quadratic system is symmetric and bilinear in \mathbf{x} and \mathbf{y} . In the case of a homogeneous quadratic system, the differential is given as

$$\mathcal{DP}(\mathbf{x}, \mathbf{y}) = \mathcal{P}(\mathbf{x} + \mathbf{y}) - \mathcal{P}(\mathbf{x}) - \mathcal{P}(\mathbf{y}),$$

since $\mathcal{P}(\mathbf{0}) = 0$ holds.

2.2 Construction Methods for MPKC's

In this section we present the basic methods for constructing multivariate public key cryptosystems (MPKC's). Basically, there are two different methods

- the standard (bipolar) construction and
- the mixed systems construction.

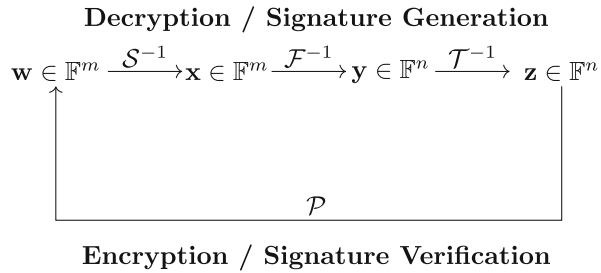
Additionally, there exist two methods to obtain public key identification schemes on the basis of multivariate polynomials: the IP based identification scheme and the MQ based identification scheme. Here, IP and MQ denote the mathematical problems underlying the security of the schemes, which are the *Isomorphism of Polynomials Problem* and the *MQ Problem* of solving a system of multivariate quadratic polynomials. A formal definition of these problems can be found in the next section.

2.2.1 The Bipolar Construction

The basic idea behind the standard construction of multivariate public key cryptosystems is to choose a system $\mathcal{F} : \mathbb{F}^n \rightarrow \mathbb{F}^m$ of m multivariate quadratic polynomials in n variables which can be easily inverted (*central map*). After that, one chooses two affine (or linear) invertible maps $\mathcal{S} : \mathbb{F}^m \rightarrow \mathbb{F}^m$ and $\mathcal{T} : \mathbb{F}^n \rightarrow \mathbb{F}^n$ to hide the structure of the central map \mathcal{F} in the public key. The *public key* of the cryptosystem is the composed quadratic map $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T} : \mathbb{F}^n \rightarrow \mathbb{F}^m$ which is supposed to be hardly distinguishable from a random system and therefore to be difficult to invert. The *private key* consists of \mathcal{S} , \mathcal{F} and \mathcal{T} and therefore allows to invert the public key.

The standard process for encryption and decryption or for signature generation and verification works as shown in Fig. 2.1.

Fig. 2.1 General workflow of bipolar schemes



2.2.1.1 Encryption Schemes ($m \geq n$)

Encryption To encrypt a message $\mathbf{z} \in \mathbb{F}^n$, one simply computes $\mathbf{w} = \mathcal{P}(\mathbf{z})$. The ciphertext of the message \mathbf{z} is $\mathbf{w} \in \mathbb{F}^m$.

Decryption To decrypt the ciphertext $\mathbf{w} \in \mathbb{F}^m$, one computes recursively $\mathbf{x} = \mathcal{S}^{-1}(\mathbf{w})$, $\mathbf{y} = \mathcal{F}^{-1}(\mathbf{x})$ and $\mathbf{z} = \mathcal{T}^{-1}(\mathbf{y})$. $\mathbf{z} \in \mathbb{F}^n$ is the plaintext corresponding to the ciphertext \mathbf{w} . Since $m \geq n$ holds, the pre-image of \mathbf{x} under \mathcal{F} and therefore the resulting plaintext is unique.

2.2.1.2 Signature Schemes ($m \leq n$)

Signature Generation To sign a document d , we use a hash function $\mathcal{H} : \{0, 1\}^* \rightarrow \mathbb{F}^m$ to compute the value $\mathbf{w} = \mathcal{H}(d) \in \mathbb{F}^m$. Then we compute $\mathbf{x} = \mathcal{S}^{-1}(\mathbf{w})$, $\mathbf{y} = \mathcal{F}^{-1}(\mathbf{x})$ and $\mathbf{z} = \mathcal{T}^{-1}(\mathbf{y})$. The signature of the document d is $\mathbf{z} \in \mathbb{F}^n$.

Here, $\mathcal{F}^{-1}(\mathbf{x})$ means finding one (of the possibly many) pre-image of \mathbf{x} under the central map \mathcal{F} . Since we have $n \geq m$, we can be sure that such a pre-image exists. Therefore every message has a signature.

Signature Verification To verify the authenticity of a document, one simply computes $\mathbf{w}' = \mathcal{P}(\mathbf{z})$ and the hash value $\mathbf{w} = \mathcal{H}(d)$ of the document. If $\mathbf{w}' = \mathbf{w}$ holds, the signature is accepted, otherwise it is rejected.

The security of bipolar schemes is based on two different problems. In particular, these are the MQ Problem and some version of the IP Problem. If the central map \mathcal{F} is publicly known (as for the Matsumoto-Imai cryptosystem (see Chap. 3)), this is the IP2S Problem. If \mathcal{F} is a part of the private key (as for UOV and Rainbow (see Chap. 5)), it is the EIP Problem. More information regarding these underlying problems can be found in Sect. 2.3.

2.2.2 Mixed Systems

To build a multivariate scheme of the mixed systems type, one starts with a quadratic map $\mathcal{F} : \mathbb{F}^{m+n} \rightarrow \mathbb{F}^m$ of the form

$$\mathcal{F}(y_1, \dots, y_n, x_1, \dots, x_m) = (h_1, \dots, h_m). \quad (2.14)$$

\mathcal{F} has to fulfill the following two conditions:

(C1) For each fixed element $(\bar{y}_1, \dots, \bar{y}_n) \in \mathbb{F}^n$ the map

$$\mathcal{F}(\bar{y}_1, \dots, \bar{y}_n, x_1, \dots, x_m) : \mathbb{F}^m \rightarrow \mathbb{F}^m$$

becomes linear.

(C2) For each fixed element $(\bar{x}_1, \dots, \bar{x}_m) \in \mathbb{F}^m$ the map

$$\mathcal{F}(y_1, \dots, y_n, \bar{x}_1, \dots, \bar{x}_m) : \mathbb{F}^n \rightarrow \mathbb{F}^m$$

is an efficiently invertible system of multivariate quadratic equations.

The *public key* of the scheme is defined as $\mathcal{P} = \mathcal{L} \circ \mathcal{F} \circ (\mathcal{S} \times \mathcal{T})$, where $\mathcal{S} : \mathbb{F}^n \rightarrow \mathbb{F}^n$ and $\mathcal{T} : \mathbb{F}^m \rightarrow \mathbb{F}^m$ are invertible affine and $\mathcal{L} : \mathbb{F}^m \rightarrow \mathbb{F}^m$ is an invertible linear map. Therefore, \mathcal{P} is a quadratic map from \mathbb{F}^{m+n} to \mathbb{F}^m . But, for any fixed $(\bar{z}_1, \dots, \bar{z}_n) \in \mathbb{F}^n$, the system

$$\mathcal{P}(\bar{z}_1, \dots, \bar{z}_n, w_1, \dots, w_m)$$

becomes a linear map from \mathbb{F}^m to itself (due to condition (C1)).

The *private key* consists of the maps \mathcal{L} , \mathcal{F} , \mathcal{S} and \mathcal{T} . Though \mathcal{L} is in general not necessary to decrypt ciphertexts or sign messages, it should be kept secret.

2.2.2.1 Encryption Schemes ($m \geq n$)

Encryption To encrypt a message $\mathbf{z} = (z_1, \dots, z_n) \in \mathbb{F}^n$, one solves the linear system

$$\mathcal{P}(z_1, \dots, z_n, w_1, \dots, w_m) = (0, \dots, 0)$$

for w_1, \dots, w_m . $\mathbf{w} = (w_1, \dots, w_m) \in \mathbb{F}^m$ is the ciphertext of the message \mathbf{z} .

Decryption To decrypt a ciphertext $\mathbf{w} \in \mathbb{F}^m$, one first computes $\mathbf{x} = (x_1, \dots, x_m) = \mathcal{T}(\mathbf{w})$. Then one solves

$$\mathcal{F}(y_1, \dots, y_n, x_1, \dots, x_m) = (0, \dots, 0) \quad (2.15)$$

for $\mathbf{y} = y_1, \dots, y_n$. Note that (2.15) is, due to condition (C2), efficiently invertible. Finally, one computes the plaintext $\mathbf{z} = \mathcal{S}^{-1}(\mathbf{y})$. Since we have $m \geq n$, the resulting plaintext is unique.

2.2.2.2 Signature Schemes ($m \leq n$)

Signature Generation To generate a signature for a document d , one uses a hash function $\mathcal{H} : \{0, 1\}^* \rightarrow \mathbb{F}^m$ to compute a hash value $\mathbf{w} = \mathcal{H}(d) = (w_1, \dots, w_m) \in \mathbb{F}^m$. Then one computes $\mathbf{x} = (x_1, \dots, x_m) = \mathcal{T}(\mathbf{w})$ and solves

$$\mathcal{F}(y_1, \dots, y_n, x_1, \dots, x_m) = (0, \dots, 0). \quad (2.16)$$

for $\mathbf{y} = y_1, \dots, y_n$. Again, (2.16) is a system of multivariate quadratic equations which, due to condition (C2), is efficiently invertible. The signature of the message d is $\mathbf{z} = \mathcal{S}^{-1}(\mathbf{y}) \in \mathbb{F}^n$. Since we have $n \geq m$, we can be sure that every message has a signature.

Signature Verification To verify the authenticity of a signature $\mathbf{z} = (z_1, \dots, z_n) \in \mathbb{F}^n$, one computes the hash value $\mathbf{w} = \mathcal{H}(d) = (w_1, \dots, w_m)$ and evaluates $\mathcal{P}(z_1, \dots, z_n, w_1, \dots, w_m)$. If the result is $(0, \dots, 0) \in \mathbb{F}^m$, the signature is accepted, otherwise it is rejected.

There exist only very few schemes of the mixed systems type. Examples for such schemes are the Dragon cryptosystems of Patarin [4]. We do not handle these schemes in this book.

As for bipolar schemes, the security of schemes of the mixed systems type is based on the MQ Problem and some type of the IP Problem (see Sect. 2.3).

Additionally to these constructions for multivariate public key encryption and signature schemes, there exist two different constructions for multivariate public key identification schemes.

In an identification scheme, a prover P wants to prove his identity to a verifier V . This is done using a zero knowledge proof in which the prover shows his knowledge of a secret s . The central property of such a proof is that the verifier does not get any information about the secret s , which prevents him from impersonating P . In this book, we describe two different constructions of such a proof based on multivariate quadratic polynomials.

2.2.3 IP Based Identification

The IP based identification scheme of Patarin [5] can be described as follows:

Key Generation The prover P randomly chooses a system $\mathcal{A} : \mathbb{F}^n \rightarrow \mathbb{F}^m$ of multivariate quadratic polynomials and two affine maps $\mathcal{S} : \mathbb{F}^m \rightarrow \mathbb{F}^m$ and $\mathcal{T} :$

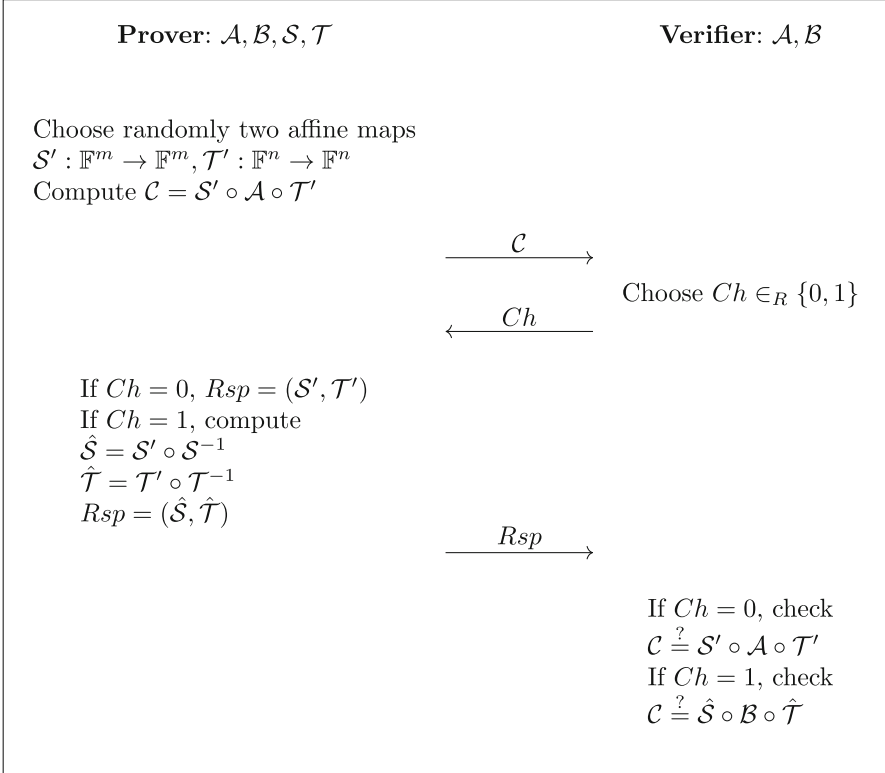


Fig. 2.2 The IP based identification scheme

$\mathbb{F}^n \rightarrow \mathbb{F}^n$. He computes $\mathcal{B} = \mathcal{S} \circ \mathcal{A} \circ \mathcal{T}$. The *public key* consists of \mathcal{A} and \mathcal{B} , the *private key* of \mathcal{S} and \mathcal{T} .

To prove his identity to a verifier, P now performs one or more rounds of the identification protocol. Figure 2.2 shows one round of the protocol.

The scheme is a zero knowledge argument of knowledge that P knows two affine maps \mathcal{S} and \mathcal{T} such that $\mathcal{B} = \mathcal{S} \circ \mathcal{A} \circ \mathcal{T}$. The cheating probability per round is $\frac{1}{2}$. Therefore, one needs 30 rounds to reduce the impersonation probability to 2^{-30} .

Using the Fiat Shamir transformation (see Sect. 6.2), the IP based identification scheme can be extended to a signature scheme. The security of the scheme is based on the IP2S Problem (see Sect. 2.3).

One major disadvantage of the IP based identification scheme is the large communication cost. In every round of the protocol, the prover has to send a multivariate quadratic system \mathcal{C} of m equations in n variables (as well as two linear systems) to the verifier, which, especially for higher levels of security, leads to an immense communication cost. This problem is solved by the MQ identification scheme.

2.2.4 MQ Based Identification

In [7], Sakumoto et al. proposed an identification scheme, whose security is solely based on the MQ Problem of solving a system of multivariate quadratic equations. This scheme is the basis of the MQDSS signature scheme which is described in Chap. 6. The MQ based identification scheme of [7] is described in Sect. 6.1.

2.3 Underlying Problems

In this section we describe the mathematical problems underlying the security of multivariate cryptosystems.

2.3.1 The MQ Problem

Solving multivariate nonlinear polynomial systems is the central problem for the security of all multivariate cryptosystems.

Definition 2.13 (Problem of Solving Polynomial Systems (PoSSo)) Given a system $\mathcal{P} = (p^{(1)}, \dots, p^{(m)})$ of m nonlinear polynomial equations in the variables x_1, \dots, x_n , find values $\bar{x}_1, \dots, \bar{x}_n$ such that

$$p^{(1)}(\bar{x}_1, \dots, \bar{x}_n) = \dots = p^{(m)}(\bar{x}_1, \dots, \bar{x}_n) = 0.$$

Solving multivariate polynomial systems is proven to be NP-complete even for the simplest case of quadratic polynomials over $\text{GF}(2)$ [3] (in its decisional variant). More precisely, it can be shown to be equivalent to the 3SAT problem.

For efficiency reasons, most multivariate schemes use only quadratic polynomials. For this special case when all polynomials $p^{(1)}, \dots, p^{(m)}$ have degree two, the polynomial solving-problem is called the **MQ Problem** (for Multivariate Quadratic).

Since nearly all cryptographic schemes can be written as systems of nonlinear polynomial equations, this problem is important to all of them. Especially for the cryptanalysis of symmetric ciphers like block and stream ciphers this so called *algebraic cryptanalysis* plays a major role [6].

In Chap. 8 we present known algorithms to solve the MQ Problem. In contrast to the case of integer factorization, all of these algorithms have exponential running time (for $m \sim n$).

2.3.2 The IP Problem

Due to their construction, the security of most multivariate schemes is not solely based on the MQ Problem, but also on (some variant of) the IP (Isomorphism of Polynomials) Problem. In particular, there exist three versions of this problem.

Definition 2.14 (Problem IP1S (Isomorphism of Polynomials with One Secret))

Given nonlinear multivariate systems \mathcal{A} and \mathcal{B} such that $\mathcal{B} = \mathcal{A} \circ \mathcal{T}$ for a linear or affine map \mathcal{T} , find a map \mathcal{T}' such that $\mathcal{B} = \mathcal{A} \circ \mathcal{T}'$.

Definition 2.15 (Problem IP2S (Isomorphism of Polynomials with Two Secrets))

Given nonlinear multivariate systems \mathcal{A} and \mathcal{B} such that $\mathcal{B} = \mathcal{S} \circ \mathcal{A} \circ \mathcal{T}$ for some linear or affine maps \mathcal{S} and \mathcal{T} , find two maps \mathcal{S}' and \mathcal{T}' such that $\mathcal{B} = \mathcal{S}' \circ \mathcal{A} \circ \mathcal{T}'$.

Definition 2.16 (Problem EIP (Extended Isomorphism of Polynomials))

Given a special class \mathcal{C} of nonlinear multivariate systems and a nonlinear multivariate system \mathcal{P} which can be written as $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$ with affine maps \mathcal{S} and \mathcal{T} and $\mathcal{F} \in \mathcal{C}$, find a decomposition of \mathcal{P} of the form $\mathcal{P} = \mathcal{S}' \circ \mathcal{F}' \circ \mathcal{T}'$ with affine maps \mathcal{S}' and \mathcal{T}' and $\mathcal{F}' \in \mathcal{C}$.

The IP2S Problem is used for the construction of multivariate schemes where the central map is publicly known (e.g. for Matsumoto-Imai (see Chap. 3)) as well as the IP based identification scheme of Sect. 2.2.

When the central map of the scheme is part of the private key, the security of the scheme is based on the EIP Problem. This is the case for most SingleField schemes such as UOV and Rainbow (see Chap. 5).

In contrast to the MQ Problem, there is not much known about the hardness of the IP Problem. In fact, for some multivariate schemes (e.g. the balanced Oil and Vinegar signature scheme) the decomposition of the public key \mathcal{P} turned out to be very easy (see Sect. 5.2). This fact prevented researchers to give security proofs for multivariate public key schemes. In fact, the only existing provably secure multivariate public key cryptosystems are based on the MQ based identification scheme of Sect. 6.1.

2.4 Security and Standard Attacks

For most multivariate public key cryptosystems, there exists no security proof which reduces the security of the scheme to the hardness of a well known mathematical problem (e.g. the MQ Problem). In fact, the only provable secure multivariate public key schemes are the MQ based identification scheme of Sect. 6.1 and the related signature scheme MQDSS (see Chap. 6).

Since there are no security proofs for multivariate public key cryptosystems, the security of these schemes is estimated by the complexities of the relevant attacks

against the schemes. Note that this is also done for most schemes with a security proof, e.g. lattice based schemes. Since the reductions are rarely tight, deriving the parameters from the security proof would lead to an inefficient scheme. So, despite of having a security proof for the scheme, it is not used for the practical parameter choice.

The standard attacks against multivariate public key schemes can be divided into two groups.

- **Direct attacks:** In a direct attack, one considers the public equation $\mathcal{P}(\mathbf{z}) = \mathbf{w}$ as an instance of the MQ Problem. In Chap. 8, we describe the most important algorithms to solve this problem.

Direct attacks can be used to attack any multivariate public key cryptosystem. However, they appeared to be most efficient against systems with a huge algebraic structure such as HFE and its variants (see Chap. 4) and SimpleMatrix (see Chap. 7).

Besides of their use to attack multivariate schemes, the algorithms used in direct attacks can also be used in different applications, for example the algebraic cryptanalysis of symmetric ciphers.

- **Structural attacks:** Structural attacks try to use the structure of the central map of a multivariate public key cryptosystem to find the composition $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$ of the public key. The most important examples of structural attacks against multivariate schemes are
 - Linearization equations attack: The most popular application of this attack is the linearization equations attack against the Matsumoto Imai cryptosystem (see Sect. 3.2).
 - Rank attacks: Rank attacks come in two flavors: The MinRank and the HighRank attack. The MinRank attack plays an important role in the security analysis of the HFE cryptosystem and its variants (see Chap. 4). For the parameter choice of Rainbow (see Sect. 5.5), both types of Rank attacks have to be taken into account.
 - Differential attacks: Differential attacks look for invariants or symmetries in the differential of the public key of an MPKC to find some information about the central map. This type of attack is used in the cryptanalysis of SFLASH (see Sect. 3.4) and PMI (see Sect. 3.3).
 - UOV related attacks: These attacks use some properties of the central map of UOV and related schemes to get information about the private key. As the name indicates, these attacks have to be considered in the security analysis of UOV and Rainbow (see Chap. 5).

2.4.1 Security Categories

When proposing concrete parameter sets for a multivariate public key cryptosystem, we consider three different security categories. Since our schemes are supposed to

Table 2.1 Security categories

Security category	Considered attacks	Breaking the scheme is as hard as breaking	Scheme provides security for
I	Classical and quantum attacks	AES-128	Today and near future
II		AES-192	Medium future
III		AES -256	Foreseeable future
Ia	Only classical attacks	SHA-256	Today and near future
IIa	attacks	SHA-384	Medium future ^a
IIIa		SHA-512	Foreseeable future ^a

^aAs long as no quantum computers exist

be resistant against quantum computer attacks, the security categories are chosen according to this fact. Note that our security categories are defined in the same way as those of the NIST standardization process.

The instances in the first category I are supposed to provide as much security as AES-128, breaking those in the second security category II is as hard as breaking AES-192 and those in category III provide as much security as AES-256.

Additionally to these basic security categories, we give (in some cases) also parameter sets which are secure only against attacks on classical computers. We denote these security categories by Ia, IIa and IIIa. Breaking the schemes in security category Ia requires as much computational effort as finding a collision of SHA-256, schemes in category IIa are as secure as SHA-384 and schemes in security category IIIa are (supposedly) as hard to break as SHA-512. Table 2.1 gives an overview of the security categories considered in this book.

2.5 Advantages and Disadvantages

Multivariate cryptosystems offer, besides their (believed) resistance against quantum computer attacks, a number of advantages:

- **Modest computational requirements:** First, the computations required by multivariate schemes are mainly simple arithmetic operations over relatively small finite fields. Therefore, running a multivariate public key cryptosystem requires much less computational power than for example performing RSA, which makes multivariate schemes attractive for the use on low cost devices such as smart cards and RFID chips.
- **Many practical signature schemes:** In the area of digital signatures, there exist multivariate schemes which offer signature sizes of little more than 100 bit (e.g. Gui; see Sect. 4.4). Other multivariate signature schemes produce signatures of length a few hundred bits. Therefore, the signatures of multivariate schemes are much shorter than those of classical schemes such as RSA and those of other post quantum signature schemes.

- **Efficient implementations:** Multivariate schemes can be implemented very efficiently. Indeed, there are many hints that multivariate schemes can be much faster than classical schemes such as RSA and ECC [1, 2].
- **Underlying hard problem:** The MQ Problem underlying the security of multivariate schemes is one of the fundamental problems of cryptography. Every (symmetric or asymmetric) cryptosystem can be written as a system of nonlinear polynomial equations. Therefore, if somebody finds an efficient algorithm for the MQ Problem, all of the other cryptographic schemes will be broken, too. Following this argumentation, the MQ Problem is one of the strongest problems of all.

However, multivariate schemes have a number of disadvantages, too:

- **Lack of security proofs:** There exist no security proofs for multivariate public key schemes and some schemes believed to be secure have been broken. On the other hand, the behavior of attacks against multivariate schemes is well understood and the theoretical complexity of these attacks matches very well with computer experiments.
- **Large public and private keys:** The key sizes of multivariate schemes are relatively large. In fact, the public key of a multivariate scheme is a system of m quadratic equations in n variables. Therefore, the public key size is cubic in n , leading to public key sizes which are much larger than those of classical schemes such as RSA.

References

1. A. Bogdanov, T. Eisenbarth, A. Rupp, C. Wolf, Time-area optimized public-key engines: Mq-cryptosystems as replacement for elliptic curves? in *CHES 2008. Lecture Notes in Computer Science*, vol. 5154 (Springer, Berlin, 2008), pp. 45–61
2. A.I.-T. Chen, M.-S. Chen, T.-R. Chen, C.-M. Cheng, J. Ding, E.L.-H. Kuo, F.Y.-S. Lee, B.-Y. Yang, SSE implementation of multivariate PKCs on modern x86 CPUs, in *CHES 2009. Lecture Notes in Computer Science*, vol. 5747 (Springer, Berlin, 2009), pp. 33–48
3. M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (W. H. Freeman, San Francisco, 1979)
4. J. Patarin, Asymmetric cryptography with a hidden monomial, in *CRYPTO 1996. Lecture Notes in Computer Science*, vol. 1109 (Springer, Berlin, 1996), pp. 45–60
5. J. Patarin, Hidden field equations (HFE) and isomorphisms of polynomials (IP): two new families of asymmetric algorithms, in *EUROCRYPT. Lecture Notes in Computer Science*, vol. 1070 (Springer, Berlin, 1996), pp. 33–48
6. A. Pyshkin, Algebraic cryptanalysis of block ciphers using Gröbner bases. PhD thesis, Darmstadt University of Technology, Germany, 2008
7. K. Sakumoto, T. Shirai, H. Hiwatari, Public-key identification schemes based on multivariate quadratic polynomials, in *CRYPTO 2011. Lecture Notes in Computer Science*, vol. 6841 (Springer, Berlin, 2011), pp. 706–723

Chapter 3

The Matsumoto-Imai Cryptosystem



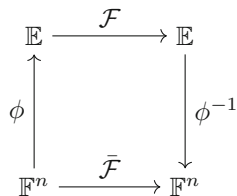
Abstract This chapter describes the Matsumoto-Imai cryptosystem, which is one of the oldest multivariate public key cryptosystems. After introducing the plain scheme, we discuss its cryptanalysis by the linearization equations attack of Patarin. We then consider variants of the basic scheme which are immune against this attack. In the area of encryption schemes, these are the PMI and PMI+ schemes, which are obtained by a concept named internal perturbation and the plus modifier. In the area of signature schemes based on MI, we present the MI-Minus or SFLASH scheme and its extension PFLASH.

The Matsumoto-Imai cryptosystem (short MI or C^*) as proposed by Tsutomu Matsumoto and Hideki Imai in 1988 [8], was one of the first multivariate cryptosystems and has attracted a lot of attention. The MI scheme is also the first example of a multivariate scheme from the BigField family. Instead of looking for an easily invertible quadratic map over the vector space \mathbb{F}^n , a multivariate BigField scheme uses an easily invertible map \mathcal{F} of Hamming weight degree 2 over a degree n extension field \mathbb{E} of \mathbb{F} as well as an isomorphism $\phi : \mathbb{F}^n \rightarrow \mathbb{E}$ to transform this map \mathcal{F} into a quadratic map $\bar{\mathcal{F}} = \phi^{-1} \circ \mathcal{F} \circ \phi$ over the vector space \mathbb{F}^n (see Fig. 3.1).

By using this strategy, Matsumoto and Imai were able to develop a very efficient cryptosystem. Since the public key of the Matsumoto-Imai scheme is a bijective map, the scheme can be used both for encryption and digital signatures.

Although the basic MI scheme was broken by the linearization equations attack of Patarin [9], the scheme had a great influence on the research in the area of multivariate cryptography. Many variants of MI, which are immune against Patarin's attack, have been proposed. The best known example for such a scheme is the MI-Minus or SFLASH [1] signature scheme, which provides short signatures and is very efficient. Indeed, one version of the SFLASH scheme (SFLASH^{v2}) was selected by the NESSIE project (New European Schemes for Signatures, Identification and Encryption) [10] as a standard for digital signatures on low cost devices. However, the SFLASH scheme was broken by a differential attack of Dubois et al. [5]. To prevent this attack, it was recommended by Ding et al. to project the public key of the scheme to a subspace of \mathbb{F}^n (PFLASH [4]).

Fig. 3.1 Construction of a multivariate BigField scheme



In the area of encryption schemes, Ding et al. proposed in [2] a technique called internal perturbation (PMI). After this scheme was broken by a differential attack by Fouque [7], another variation called PMI+ was proposed to fix this problem [3].

However, the Matsumoto-Imai scheme influenced the research in the area of multivariate cryptography in a much wider sense, and many of the techniques presented in this book are inspired by the MI scheme. Examples for this are the HFE cryptosystem (see Chap. 4) and the Oil and Vinegar signature scheme (see Chap. 5).

This chapter is organized as follows: After introducing the basic Matsumoto-Imai scheme in Sect. 3.1, Sect. 3.2 describes the linearization equations attack of Patarin. In Sect. 3.3, we give an overview of encryption schemes on the basis of MI. After introducing Ding's technique of internal perturbation, we describe Fouques differential attack against the PMI scheme and how this attack can be prevented by the plus method. In the last section of this chapter (Sect. 3.4), we finally deal with signature schemes on the basis of the MI scheme. In particular, this section introduces the SFLASH signature scheme (MI-Minus), describes the differential attack of Dubois et al. to break this scheme and finally the last section presents the projecting technique of Ding et al. to prevent this attack.

3.1 The Basic Matsumoto-Imai Cryptosystem

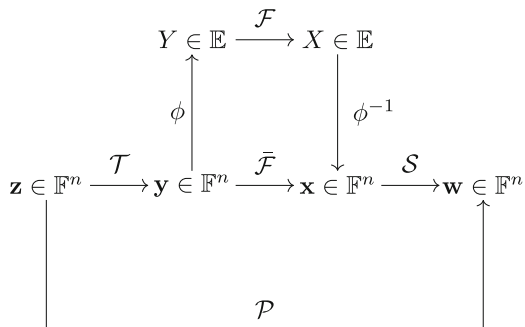
The basic Matsumoto-Imai (short MI or C^*) cryptosystem as proposed in [8] can be described as follows.

Let \mathbb{F} be a finite field of characteristic 2 with q elements and let $g(X) \in \mathbb{F}[X]$ be an irreducible polynomial of degree n over \mathbb{F} .¹ Therefore, the field $\mathbb{E} = \mathbb{F}[X]/g(X)$ is a degree n extension field of \mathbb{F} . Let $\phi : \mathbb{F}^n \rightarrow \mathbb{E}$ be the standard isomorphism between the vector space \mathbb{F}^n and the extension field \mathbb{E} , i.e.

$$\phi(x_1, \dots, x_n) = \sum_{i=1}^n x_i X^{i-1}.$$

¹For the functionality of the MI scheme, the condition of $\text{char}(\mathbb{F}) = 2$ is not necessarily required. However, one has to make some changes to the scheme to guarantee the bijectivity of the central map.

Fig. 3.2 Construction of the Matsumoto-Imai cryptosystem



The central map $\mathcal{F} : \mathbb{E} \rightarrow \mathbb{E}$ of the C^* scheme is a bijective map over the extension field \mathbb{E} , which is defined as

$$\mathcal{F}(Y) = Y^{q^\theta + 1} \quad (3.1)$$

with $0 < \theta < n$ and $\gcd(q^n - 1, q^\theta + 1) = 1$.²

In order to invert the central map \mathcal{F} , we use the extended Euclidean algorithm to compute an integer h with $h(q^\theta + 1) = 1 \pmod{q^n - 1}$. Therefore, we get

$$\mathcal{F}^{-1}(X) = X^h = Y^{h(q^\theta + 1)} = Y^{k(q^n - 1) + 1} = Y.$$

The public key of the scheme is the composed map $\mathcal{P} = \mathcal{S} \circ \bar{\mathcal{F}} \circ \mathcal{T} = \mathcal{S} \circ \phi^{-1} \circ \mathcal{F} \circ \phi \circ \mathcal{T} : \mathbb{F}^n \rightarrow \mathbb{F}^n$ with two invertible linear maps $\mathcal{S} : \mathbb{F}^n \rightarrow \mathbb{F}^n$ and $\mathcal{T} : \mathbb{F}^n \rightarrow \mathbb{F}^n$, the *private* key consists of \mathcal{S} , h and \mathcal{T} . However, we can also assume that h is public since θ is in a very small range.

Figure 3.2 shows a graphical illustration of the Matsumoto-Imai scheme. Due to the bijectivity of the central map \mathcal{F} , the basic MI scheme can be used both for en/decryption and digital signatures.

3.1.1 MI as an Encryption Scheme

Encryption To encrypt a plaintext $\mathbf{z} = (z_1, \dots, z_n) \in \mathbb{F}^n$, one simply computes $\mathbf{w} = \mathcal{P}(\mathbf{z}) \in \mathbb{F}^n$.

Decryption To decrypt a ciphertext $\mathbf{w} \in \mathbb{F}^n$, one computes recursively $\mathbf{x} = \mathcal{S}^{-1}(\mathbf{w}) \in \mathbb{F}^n$, $X = \phi(\mathbf{x}) \in \mathbb{E}$, $Y = \mathcal{F}^{-1}(X) \in \mathbb{E}$, $\mathbf{y} = \phi^{-1}(Y) \in \mathbb{F}^n$ and $\mathbf{z} = \mathcal{T}^{-1}(\mathbf{y})$. The plaintext corresponding to the ciphertext \mathbf{w} is given by $\mathbf{z} \in \mathbb{F}^n$.

²Together with $\text{char}(\mathbb{F}) = 2$, the condition on θ guarantees that the central map \mathcal{F} is bijective.

3.1.2 MI as a Signature Scheme

Signature Generation To generate a signature for a document d , one uses a hash function $\mathcal{H} : \{0, 1\}^* \rightarrow \mathbb{F}^n$ to compute the hash value $\mathbf{w} = \mathcal{H}(d) \in \mathbb{F}^n$. After that, one computes $\mathbf{x} = \mathcal{S}^{-1}(\mathbf{w}) \in \mathbb{F}^n$, $X = \phi(\mathbf{x}) \in \mathbb{E}$, $Y = \mathcal{F}^{-1}(X) \in \mathbb{E}$, $\mathbf{y} = \phi^{-1}(Y) \in \mathbb{F}^n$ and $\mathbf{z} = \mathcal{T}^{-1}(\mathbf{y})$. The signature of the document d is given by $\mathbf{z} \in \mathbb{F}^n$.

Signature Verification To check, if $\mathbf{z} = (z_1, \dots, z_n) \in \mathbb{F}^n$ is indeed a valid signature for the document d , one computes $\mathbf{w} = \mathcal{H}(d) \in \mathbb{F}^n$ and $\mathbf{w}' = \mathcal{P}(\mathbf{z}) \in \mathbb{F}^n$. If $\mathbf{w}' = \mathbf{w}$ holds, the signature is accepted, otherwise rejected.

3.1.3 Degree of the Public Key Components

The central equation $\mathcal{F}(Y) = Y^{q^\theta+1}$ of the Matsumoto-Imai scheme can be written as a product $\mathcal{F}(Y) = \mathcal{F}_1(Y) \cdot \mathcal{F}_2(Y)$ with $\mathcal{F}_1(Y) = Y^{q^\theta}$ and $\mathcal{F}_2(Y) = Y = Y^{q^0}$. Note that both \mathcal{F}_1 and \mathcal{F}_2 are so called Frobenius maps. In fact, these maps are elements of the Galois group $G = \text{Gal}(\mathbb{E}/\mathbb{F})$, and therefore are \mathbb{F} -linear maps on \mathbb{E} . From this it is easy to see that both $\phi^{-1} \circ \mathcal{F}_1 \circ \phi$ and $\phi^{-1} \circ \mathcal{F}_2 \circ \phi$ are \mathbb{F} -linear maps over \mathbb{F}^n . Hence, each component of $\phi^{-1} \circ \mathcal{F} \circ \phi$ has total degree two in $\mathbb{F}[x_1, \dots, x_n]$.

In order to better see the relationship between the degree of a map $\mathcal{G}(X) \in \mathbb{E}[X]$ and the degree of the components of $\phi^{-1} \circ \mathcal{G} \circ \phi$ in $\mathbb{F}[x_1, \dots, x_n]$, we introduce the notion of the q -Hamming weight degree. The q -Hamming weight degree of a monomial X^e ($0 \leq e < q^n$) is defined as the sum of the coefficients in the base- q expansion of e . The q -Hamming weight degree of a function $\mathcal{G}(X) \in \mathbb{E}[X]$ is the largest q -Hamming weight degree of all monomials of $\mathcal{G}(X)$.

Example Let $q = 4$ and $\mathcal{G}(X) = X^{25} + X^{12} + X^5$. We write the exponents of X in base q and compute the 4-Hamming weight degree of each term separately, obtaining

Exponent	In base q	q -Hamming weight degree
25	$1 * 4^2 + 2 * 4^1 + 1 * 4^0$	$1 + 2 + 1 = 4$
12	$3 * 4^1$	3
5	$1 * 4^1 + 1 * 4^0$	$1 + 1 = 2$

Therefore, we get

$$q\text{-Hamming weight degree}(\mathcal{G}(X)) = q\text{-Hamming weight degree}(X^{25}) = 4.$$

Now suppose that we have a function $\mathcal{G}(X) \in \mathbb{E}[X]$ of q -Hamming weight degree d . Then the components of $\phi^{-1} \circ \mathcal{G} \circ \phi$ will be of total degree d . In

particular, since in the case of the MI scheme, the q -Hamming weight degree of $\tilde{\mathcal{F}} = \phi^{-1} \circ \mathcal{F} \circ \phi$ is two, it follows that the total degree of each of the components $\tilde{f}^{(1)}, \dots, \tilde{f}^{(n)}$ is two. Since \mathcal{S} and \mathcal{T} are invertible affine transformations, the same also holds for the components $p^{(1)}, \dots, p^{(n)}$ of the public key.

3.1.4 Key Sizes and Efficiency

The public key of the Matsumoto-Imai scheme consists of n multivariate quadratic polynomials in n variables. Each of these polynomials has therefore $n(n+1)/2 + n + 1 = (n+1)(n+2)/2$ terms and therefore the same number of coefficients which have to be stored in the public key. Therefore, the size of the whole public key is

$$\text{size}_{\text{pk MI}} = n \cdot \frac{(n+1)(n+2)}{2} \text{ field elements.} \quad (3.2)$$

If we have $\mathbb{F} = \text{GF}(2)$, this number is a bit smaller, since we have the field equations $x_i^2 = x_i$. We get

$$\text{size}_{\text{pk MI, GF}(2)} = n \cdot \frac{n^2 + n + 2}{2} \text{ field elements.} \quad (3.3)$$

The private key consists of the two affine maps \mathcal{S} and $\mathcal{T} : \mathbb{F}^n \rightarrow \mathbb{F}^n$ and the parameter h . It therefore has a size of

$$\text{size}_{\text{sk MI}} = 2n(n+1) \text{ field elements} + \text{Log}_2(h) \text{ bit.} \quad (3.4)$$

For the parameters proposed by Matsumoto and Imai ($q = 2^8$, $n = 32$), this leads to a public key size of 17.5 kB, which is much larger than the key size of e.g. RSA.

This problem of large key sizes is one of the major problems in multivariate cryptography. On the other hand, the MI scheme can be implemented much more efficiently than RSA. If the cardinality q of the ground field \mathbb{F} is reasonably small, we can store the multiplication table of \mathbb{F} in cache memory. This allows much faster arithmetic operations than in the case of RSA, where we have to deal with large integers.

The most costly step during the decryption process of the Matsumoto-Imai scheme is the exponentiation of the polynomial \tilde{X} to its h -th power. In our implementation, we performed this step using the well known square-and-multiply method. To be even more efficient, one looks for exponents θ for which the binary representation of the inverse h has a simple structure.

Fig. 3.3 Addition and multiplication table of $\text{GF}(2^2)$

+	0	1	α	α^2		*	0	1	α	α^2
0	0	1	α	α^2		0	0	0	0	
1	1	0	α^2	α		1	0	1	α	α^2
α	α	α^2	0	1		α	0	α	α^2	1
α^2	α^2	α	1	0		α^2	0	α^2	1	α

3.1.5 Toy Example

In this section we illustrate the workflow of the Matsumoto-Imai cryptosystem using a toy example with small parameters.

Let $\mathbb{F} = \text{GF}(2^2)$ be the finite field with $q = 2^2 = 4$ elements. The multiplicative group for the nonzero elements of this field can be generated by the field element α which satisfies the relation $\alpha^2 + \alpha + 1 = 0$. The field elements of \mathbb{F} can be represented as $\{0, 1, \alpha, \alpha^2\}$ and the addition and multiplication tables are given in Fig. 3.3.

Next, we choose $n = 3$ and $g(X) = X^3 + \alpha$ as the irreducible polynomial in $\mathbb{F}[X]$. Set $\mathbb{E} = \mathbb{F}[X]/(X^3 + \alpha)$. There are only two possible choices for the MI exponent θ ; namely $\theta = 1$ or $\theta = 2$. We will use $\theta = 2$. Thus, the map \mathcal{F} and its inverse are given by

$$\mathcal{F}(Y) = Y^{1+4^2} \quad \text{and} \quad \mathcal{F}^{-1}(X) = X^{26}.$$

Let \mathcal{S} and \mathcal{T} be given by

$$\mathcal{S} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} \alpha & \alpha & \alpha^2 \\ 0 & 0 & \alpha \\ \alpha^2 & \alpha & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} 0 \\ \alpha^2 \\ \alpha \end{pmatrix}$$

and

$$\mathcal{T} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ \alpha & \alpha^2 & 1 \\ \alpha & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} 1 \\ \alpha^2 \\ \alpha^2 \end{pmatrix}.$$

To obtain the public key polynomials in terms of the plaintext variables x_1, x_2, x_3 , we begin by computing $\phi \circ \mathcal{T}(x_1, x_2, x_3)^T$, which we find to be

$$(\alpha x_1 + \alpha^2)X^2 + (\alpha x_1 + \alpha^2 x_2 + x_3 + \alpha^2)X + x_1 + x_2 + x_3 + 1.$$

If we denote this by \tilde{X} , then we can compute $\mathcal{F}(\tilde{X}) = \tilde{X}^{1+4^2} = \tilde{X} \cdot \tilde{X}^{16}$. Thus, $\mathcal{F}(\tilde{X})$ is given by

$$(\alpha^2 x_1^2 + x_1 x_2 + x_1 x_3 + \alpha^2 x_1 + x_2^2 + \alpha x_2 + \alpha^2 x_3^2 + \alpha x_3 + \alpha^2)X^2$$

$$\begin{aligned}
& + (x_1^2 + \alpha x_1 x_2 + x_1 x_3 + \alpha x_1 + x_2^2 + \alpha^2 x_2 x_3 + \alpha x_3^2 + \alpha^2 x_3)X \\
& + \alpha x_1 x_2 + \alpha^2 x_1 x_3 + x_2^2 + \alpha^2 x_2 + x_3^2 + x_3 + \alpha.
\end{aligned}$$

Finally, we compute $\mathcal{S} \circ \phi^{-1}(\mathcal{F}(\tilde{X}))$ to get the public key polynomials

$$\begin{aligned}
p^{(1)}(x_1, x_2, x_3) &= \alpha^2 x_1 x_2 + x_1 + \alpha^2 x_2^2 + x_2 x_3 + \alpha^2 x_3^2 + \alpha x_3 + 1, \\
p^{(2)}(x_1, x_2, x_3) &= x_1^2 + \alpha x_1 x_2 + \alpha x_1 x_3 + x_1 + \alpha x_2^2 + \alpha^2 x_2 + x_3^2 + \alpha^2 x_3 + \alpha, \\
p^{(3)}(x_1, x_2, x_3) &= x_1^2 + \alpha^2 x_1 x_2 + x_1 x_3 + x_2 x_3 + \alpha^2 x_3^2.
\end{aligned}$$

We want to encrypt the plaintext $\mathbf{z} = (z_1, z_2, z_3) = (0, \alpha, \alpha^2)$. To do this, we compute

$$\begin{aligned}
w_1 &= p^{(1)}(0, \alpha, \alpha^2) = \alpha, \\
w_2 &= p^{(2)}(0, \alpha, \alpha^2) = \alpha, \\
w_3 &= p^{(3)}(0, \alpha, \alpha^2) = 0
\end{aligned}$$

to get the ciphertext $\mathbf{w} = (\alpha, \alpha, 0)$.

In order to decrypt this ciphertext, we need the private key which consists of the two maps \mathcal{S} and \mathcal{T} , and the parameter h . First we compute

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \mathcal{S}^{-1} \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ \alpha & 0 & 1 \\ 0 & \alpha^2 & 0 \end{pmatrix} \begin{pmatrix} w_1 - 0 \\ w_2 - \alpha^2 \\ w_3 - \alpha \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ \alpha^2 \end{pmatrix}$$

and lift $\mathbf{x} \in \mathbb{F}^3$ to the extension field \mathbb{E} , obtaining $\tilde{X} = \alpha^2 X^2 + X + 1$. In the next step, we invert the central map \mathcal{F} by raising \tilde{X} to the $(h = 26)$ -th power. We obtain

$$\tilde{Y} = \mathcal{F}^{-1}(\tilde{X}) = \tilde{X}^{26} = \alpha^2 X^2 + X.$$

We move \tilde{Y} back to the vector space (obtaining $\mathbf{y} = (0, 1, \alpha^2)$) and compute the plaintext by

$$\mathbf{z} = \begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} = \mathcal{T}^{-1} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 0 & 0 & \alpha^2 \\ \alpha^2 & \alpha^2 & 1 \\ \alpha & \alpha^2 & \alpha \end{pmatrix} \begin{pmatrix} y_1 - 1 \\ y_2 - \alpha^2 \\ y_3 - \alpha^2 \end{pmatrix} = \begin{pmatrix} 0 \\ \alpha \\ \alpha^2 \end{pmatrix}.$$

3.2 The Linearization Equations Attack

Definition 3.1 Let $\mathcal{P} = (p_1, \dots, p_m)$ be the public key of a multivariate public key cryptosystem. A **linearization equation** is a polynomial equation in $\mathbb{F}[w_1, \dots, w_m, z_1, \dots, z_n]$ of the form

$$\sum_{i=1}^m \sum_{j=1}^n \alpha_{ij} w_i z_j + \sum_{i=1}^m \beta_i w_i + \sum_{j=1}^n \gamma_j z_j + \delta, \quad (3.5)$$

which, when substituting a valid plaintext/ciphertext pair (\mathbf{z}/\mathbf{w}) , evaluates to 0. Note that, when substituting a ciphertext (w_1, \dots, w_m) into the linearization equation, we obtain a linear equation in the plaintext variables z_1, \dots, z_n .

Remark 3.2 In order to be used in cryptanalysis, (3.5) does not have to be bilinear in the elements of \mathbf{z} and \mathbf{w} . In fact, we can define **higher order linearization equations** of the form

$$\sum_{i=1}^n g_i(w_1, \dots, w_m) z_i + g(w_1, \dots, w_m). \quad (3.6)$$

The degree of the HOLE is given by

$$d = \deg(\text{HOLE}) = \max(\deg(g_1, \dots, g_n, g)).$$

However, for degree $d > 2$, finding a higher order linearization equation gets very difficult, since the number of coefficients in the polynomials g_i increases exponentially.

So, how do we use linearization equations in the cryptanalysis of a multivariate public key cryptosystem?

Suppose that we have an MPKC with public key \mathcal{P} and we know that the (plaintext/ciphertext) pairs (\mathbf{z}/\mathbf{w}) fulfill some linearization equations of the form

$$\sum_{i=1}^m \sum_{j=1}^n \alpha_{ij} w_i z_j + \sum_{i=1}^m \beta_i w_i + \sum_{j=1}^n \gamma_j z_j + \delta = 0.$$

By computing a large number (approximately $(m+1) \cdot (n+1)$) of (plaintext/ciphertext) pairs³ and substituting them into the linearization equation, we get a system of linear equations in the coefficients α_{ij} , β_i , γ_j and δ which can be solved by Gaussian elimination. By doing so, we get a system \mathcal{L} of bilinear equations in the plaintext/ciphertext elements $z_1, \dots, z_n, w_1, \dots, w_m$.

³Note that, for a public key encryption scheme, this can be performed easily.

Now, let us assume that we are given a ciphertext $\mathbf{w}^* = (w_1^*, \dots, w_m^*)$ that we want to decrypt. By substituting the ciphertext elements into the bilinear equations computed in the previous step we get linear equations in the plaintext variables z_1^*, \dots, z_n^* . In some cases, we get enough linear equations to recover the whole plaintext. However, even if this is not the case, the linear equations can be used to speed up e.g. direct attacks against the cryptosystem.

3.2.1 Linearization Equations Attack on Matsumoto-Imai

In [9], Patarin proposed a linearization equations attack against the Matsumoto-Imai cryptosystem. Recall that, for the standard Matsumoto-Imai scheme, we have $m = n$ and

$$X = \mathcal{F}(Y) = Y^{q^\theta+1}.$$

Then we have

$$\begin{aligned} X^{q^\theta-1} &= (Y^{q^\theta+1})^{q^\theta-1} \\ &= Y^{(q^\theta+1)(q^\theta-1)} \\ &= Y^{q^{2\theta}-1}. \end{aligned}$$

By multiplying both sides by XY we get

$$YX^{q^\theta} = Y^{q^{2\theta}}X$$

or

$$YX^{q^\theta} - Y^{q^{2\theta}}X = 0.$$

Finally, we define

$$\tilde{R} : \mathbb{E}^2 \rightarrow \mathbb{E}, \tilde{R}(X, Y) = YX^{q^\theta} - Y^{q^{2\theta}}X = 0.$$

and

$$R : \mathbb{F}^{2n} \rightarrow \mathbb{F}^n, R(x_1, \dots, x_n, y_1, \dots, y_n) = \phi^{-1} \circ \tilde{R} \circ (\phi \times \phi)(x_1, \dots, x_n, y_1, \dots, y_n).$$

Note that, due to the Frobenius isomorphism, the map R is a bilinear map in x_1, \dots, x_n and y_1, \dots, y_n which evaluates to 0 when substituting a (plaintext/ciphertext) pair. Therefore, R yields a set \mathcal{L} of linearization equations for the central map \mathcal{F} of the Matsumoto-Imai scheme. In fact, since \mathcal{L} is closed under addition and scalar multiplication, it is a vector space.

In the following we consider the question how many linear independent linear equations arise from R by substituting a ciphertext $\hat{\mathbf{y}}$ into R .

Lemma 3.3 *For a fixed $\hat{Y} \in \mathbb{E}$, there exist at most $q^{\gcd(\theta, n)}$ different values of $X \in \mathbb{E}$ such that*

$$\tilde{R}(X, \hat{Y}) = 0.$$

Proof We have

$$X\hat{Y}^{q^\theta} = X^{q^{2\theta}}\hat{Y} \quad (3.7)$$

or, if $\hat{Y} \neq 0$,

$$X^{q^{2\theta}-1} = \hat{Y}^{q^{\theta-1}},$$

which has at most $\gcd(q^{2\theta} - 1, q^n - 1)$ different solutions for X . Furthermore, due to the condition $\gcd(q^\theta + 1, q^n - 1) = 1$ we have

$$\gcd(q^{2\theta} - 1, q^n - 1) = \gcd(q^\theta - 1, q^n - 1).$$

Hence, (3.7) has at most $\gcd(q^\theta - 1, q^n - 1) + 1$ solutions, including the trivial solution. Since we further have

$$\gcd(q^\theta - 1, q^n - 1) = q^{\gcd(\theta, n)} - 1,$$

the maximal number of solutions of (3.7) is given by $q^{\gcd(\theta, n)}$. \square

As the Lemma shows, the linear system we obtain by substituting a fixed ciphertext $\hat{\mathbf{y}} \in \mathbb{F}^n$ into R has at most $q^{\gcd(n, \theta)}$ solutions, which form a linear space of dimension $\leq \gcd(n, \theta)$. Therefore, the linear system arising from R contains at least $n - \gcd(\theta, n)$ linearly independent equations.

However, for cryptanalytic purposes, it is more important how many linearly independent equations in the plaintext variables we get after substituting the challenge ciphertext into the linearization equations obtained from the public key. This question is answered by Theorem 3.4.

Theorem 3.4 *Let \mathcal{P} be a Matsumoto-Imai public key. Then, after substituting the challenge ciphertext $\mathbf{w}^* \in \mathbb{F}^n \setminus \{0\}$ in the linearization equations obtained from \mathcal{P} , we get at least*

$$n - \gcd(n, \theta) \geq \frac{2n}{3}$$

linearly independent linear equations in the plaintext variables z_1, \dots, z_n . In particular, if $\gcd(n, \theta) = 1$ holds, there remain only q plaintext candidates.

Proof For the proof, we introduce the following notation:

- \mathcal{L} is the space of linearization equations obtained from the Matsumoto-Imai central map \mathcal{F} .
- $\tilde{\mathcal{L}}$ is the space of linearization equations obtained from the Matsumoto-Imai public key \mathcal{P} .
- \mathcal{L}_{X^*} is the space of linear equations in the plaintext variables we obtain after substituting a ciphertext X^* into the linearization equations from \mathcal{L} .
- $\tilde{\mathcal{L}}_{\mathbf{w}^*}$ is the space of linear equations in the plaintext variables we obtain after substituting a ciphertext \mathbf{w}^* into the linearization equations of $\tilde{\mathcal{L}}$.

With this notation, the linearization equations obtained from R are contained in the space \mathcal{L} . Lemma 3.3 states that the space \mathcal{L}_{X^*} , obtained by substituting the ciphertext X^* into the linearization equations of \mathcal{L} , has dimension at least $n - \gcd(\theta, n)$. We have to show that

$$\dim(\tilde{\mathcal{L}}_{\mathbf{w}^*}) = n - \gcd(n, \theta) \geq \frac{2n}{3}$$

holds. The proof is provided by Lemmas 3.5–3.7. \square

First, in Lemma 3.5, we show that the affine transformations \mathcal{S} and \mathcal{T} do not have any effect on the dimension of the space of the linearization equations.

Lemma 3.5 *With the notation of Theorem 3.4, we have*

$$\dim \mathcal{L} = \dim \tilde{\mathcal{L}}.$$

Proof First, we assume that \mathcal{T} is the identity map, so that $\mathcal{P} = \mathcal{S} \circ \mathcal{F}$ or

$$p^{(i)}(x_1, \dots, x_n) = \sum_{j=1}^n s_{ij} f^{(j)}(x_1, \dots, x_n) + s_{i0}.$$

Let

$$r = \sum_{i,j=1}^n a_{ij} x_i y_j + \sum_{i=1}^n b_i x_i + \sum_{j=1}^n c_j y_j + d$$

be a linearization equation for the public polynomials. Therefore, when substituting a (plaintext/ciphertext) pair $(\hat{\mathbf{x}}, \mathcal{P}(\hat{\mathbf{x}}))$ into it, r evaluates to 0. We get

$$\begin{aligned} 0 &= \sum_{i,j=1}^n a_{ij} \hat{x}_i p^{(j)}(\hat{\mathbf{x}}) + \sum_{i=1}^n b_i \hat{x}_i + \sum_{j=1}^n c_j p^{(j)}(\hat{\mathbf{x}}) + d \\ &= \sum_{i,j=1}^n a_{ij} \hat{x}_i \left(\sum_{k=1}^n s_{jk} f^{(k)}(\hat{\mathbf{x}}) + s_{j0} \right) + \sum_{i=1}^n b_i \hat{x}_i \end{aligned}$$

$$\begin{aligned}
& + \sum_{j=1}^n c_j \left(\sum_{k=1}^n s_{jk} f^{(k)}(\hat{\mathbf{x}}) + s_{j0} \right) + d \\
& = \sum_{i,j=1}^n a'_{ij} \hat{x}_i f^{(j)}(\hat{\mathbf{x}}) + \sum_{i=1}^n b'_i \hat{x}_i + \sum_{j=1}^n c'_j f^{(j)}(\hat{\mathbf{x}}) + d',
\end{aligned}$$

which is a linearization equation for the central map. Similarly, by looking at $\mathcal{F} = \mathcal{S}^{-1} \circ \mathcal{P}$ and starting with a linearization equation for the central polynomials, we can get a linearization equation for the polynomials of $\mathcal{S} \circ \mathcal{F}$.

We therefore have an isomorphism between the space of linearization equations of \mathcal{F} and those of $\mathcal{S} \circ \mathcal{F}$, which implies that both spaces have the same dimension.

Now, let us suppose that the first affine map \mathcal{S} is the identity map. Let

$$\bar{x}_i = \mathcal{T}(\mathbf{x})_i = \sum_{j=1}^n t_{ij} x_j + t_{i0}, \quad (3.8)$$

so that

$$p^{(i)}(x_1, \dots, x_n) = f^{(i)}(\bar{x}_1, \dots, \bar{x}_n).$$

Let r be a linearization equation for the central polynomials, i.e.

$$0 = r(\mathbf{x}, \mathcal{F}(\mathbf{x})) = \sum_{i,j=1}^n a_{ij} x_i f^{(j)}(\mathbf{x}) + \sum_{i=1}^n b_i x_i + \sum_{j=1}^n c_j f^{(j)}(\mathbf{x}) + d$$

which, since the invertible change of variables (3.8) amounts to a perturbation of \mathbb{F}^n , yields

$$0 = \sum_{i,j=1}^n a_{ij} \bar{x}_i f^{(j)}(\bar{\mathbf{x}}) + \sum_{i=1}^n b_i \bar{x}_i + \sum_{j=1}^n c_j f^{(j)}(\bar{\mathbf{x}}) + d.$$

But then, we have

$$0 = \sum_{i,j=1}^n a_{ij} \bar{x}_i p^{(j)}(\mathbf{x}) + \sum_{i=1}^n b_i \bar{x}_i + \sum_{j=1}^n c_j p^{(j)}(\mathbf{x}) + d,$$

which, using (3.8), can be rewritten as

$$0 = \sum_{i,j=1}^n a'_{ij} x_i p^{(j)}(\mathbf{x}) + \sum_{i=1}^n b_i x_i + \sum_{j=1}^n c_j p^{(j)}(\mathbf{x}) + d,$$

a linearization equation for the public polynomials.

Similarly, by looking at $\mathcal{F} = \mathcal{P} \circ \mathcal{T}^{-1}$ and starting with a linearization equation for the public polynomials, we can derive a linearization equation for the central map.

We therefore have an isomorphism between the space of linearization equations of \mathcal{F} and the space of linearization equations of $\mathcal{F} \circ \mathcal{T}$, which implies that both spaces have the same dimension.

We can combine these two isomorphism to an isomorphism between the two spaces \mathcal{L} and $\tilde{\mathcal{L}}$. From this we derive

$$\dim \mathcal{L} = \dim \tilde{\mathcal{L}}.$$

□

Lemma 3.6 shows that the same holds after substituting the ciphertext into the linearization equations.

Lemma 3.6 *Using the notation of Theorem 3.4, we have*

$$\dim \mathcal{L}_{X^*} = \dim \tilde{\mathcal{L}}_{\mathbf{w}^*}.$$

Proof Lemma 3.5 provides a bijection between the two spaces \mathcal{L} and $\tilde{\mathcal{L}}$. This implies a bijection also between \mathcal{L}_{X^*} and $\tilde{\mathcal{L}}_{\mathbf{w}^*}$. □

From Lemma 3.3 we know that $\dim \mathcal{L}_{X^*} = n - \gcd(\theta, n)$ holds. According to Lemma 3.6, this is also the dimension of the space $\tilde{\mathcal{L}}_{\mathbf{w}^*}$ obtained by substituting the ciphertext $\mathbf{w}^* \in \mathbb{F}^n$ into the linearization equations obtained from the public key. Finally, Lemma 3.7 provides a lower bound for this dimension.

Lemma 3.7 *In the case of Matsumoto-Imai we have*

$$n - \gcd(n, \theta) \geq \frac{2n}{3}.$$

Proof The three largest values of $\gcd(n, \theta)$ are n , $\frac{n}{2}$ (if n is even) and $\frac{n}{3}$ (if n is divisible by three). So, if we can show that the first two cases are impossible, we are done.

The first case ($\theta = n$) is impossible, because θ was explicitly chosen to be in the set $\{1, \dots, n-1\}$. Second, if $\gcd(n, \theta) = n/2$, θ must be $n/2$ itself. But then we have

$$\gcd(q^{n-1}, q^{\theta+1}) = \gcd(q^{(n/2-1)(n/2+1)}, q^{n/2+1}) = q^{n/2+1} > 1,$$

which contradicts the choice of θ in Sect. 3.1.

We therefore have $\gcd(n, \theta) \leq \frac{n}{3}$, which, together with Lemmas 3.3, 3.5 and 3.6, proves Theorem 3.4. □

Algorithm 3.1 summarizes the steps of the linearization equations attack.

Algorithm 3.1 Linearization equations attack

Input: (Matsumoto-Imai) public key $\mathcal{P} : \mathbb{F}^n \rightarrow \mathbb{F}^m$, challenge ciphertext $\mathbf{w}^* \in \mathbb{F}^m$

Output: linear equations in the plaintext elements z_1^*, \dots, z_n^*

- 1: Compute $(m+1) \cdot (n+1)$ (plaintext/ciphertext) pairs $(\mathbf{z}_1, \mathbf{w}_1), \dots, (\mathbf{z}_{(m+1)(n+1)}, \mathbf{w}_{(m+1)(n+1)})$ and substitute them into the linearization equation (3.5).
 - 2: Use Gaussian elimination to solve the resulting linear system for the coefficients α_{ij} , β_i , γ_j and δ . The result is a set of bilinear equations b_1, \dots, b_k in the variables z_1, \dots, z_n and w_1, \dots, w_m .
 - 3: Substitute the challenge ciphertext \mathbf{w}^* into the equations b_1, \dots, b_k to obtain a set l_1, \dots, l_ℓ of linear equations in the plaintext variables z_1^*, \dots, z_n^* .
 - 4: **return** l_1, \dots, l_ℓ .
-

Complexity of the Attack

- Evaluating a multivariate quadratic system of $m = O(n)$ equations in n variables takes time $O(n^2)$. Therefore, computing the $(m+1) \cdot (n+1)$ (plaintext/ciphertext) pairs in step 1 of the algorithm takes time $O(n^4)$.
- Solving the linear system of $(m+1) \cdot (n+1)$ equations in the same number of variables takes time $O(n^6)$.
- The remaining steps of the attack have a significantly lower complexity. Therefore, the cost of the whole attack is $O(n^6)$.

3.2.2 Toy Example

We now illustrate Patarin's linearization equations attack against the Matsumoto-Imai cryptosystem using a toy example. Let us assume that we have $\mathbb{F} = \text{GF}(4)$, $n = 5$ and we are given a Matsumoto-Imai public key of the form

$$\begin{aligned} p^{(1)} &= \alpha x_1 x_2 + \alpha^2 x_1 x_3 + x_1 x_4 + \alpha x_1 x_5 + \alpha x_2 x_3 + \alpha^2 x_2 x_4 + \alpha^2 x_2 x_5 + x_2 \\ &= \alpha^2 x_3^2 + x_3 x_5 + x_4 x_5 + \alpha^2 x_4 + x_5^2 + x_5 + \alpha, \end{aligned}$$

$$\begin{aligned} p^{(2)} &= \alpha x_1 x_2 + x_1 x_3 + \alpha^2 x_1 x_4 + x_1 x_5 + \alpha x_2^2 + x_2 x_3 \\ &\quad + x_2 x_5 + \alpha x_2 + \alpha x_3^2 + \alpha x_3 x_4 + x_3 + \alpha^2 x_4^2 + x_4 x_5 + \alpha^2 x_5^2 + \alpha^2, \end{aligned}$$

$$\begin{aligned} p^{(3)} &= \alpha^2 x_1^2 + x_1 x_2 + \alpha^2 x_1 x_3 + \alpha^2 x_1 x_4 + \alpha^2 x_1 x_5 + \alpha x_1 + \alpha^2 x_2 x_3 + \alpha^2 x_2 x_4 \\ &\quad + x_2 x_5 + \alpha x_2 + \alpha^2 x_3^2 + \alpha x_3 x_5 + \alpha^2 x_3 + x_4 x_5 + \alpha x_5^2 + \alpha^2 x_5 + \alpha^2, \end{aligned}$$

$$\begin{aligned} p^{(4)} &= x_1^2 + \alpha^2 x_1 x_2 + \alpha x_1 x_5 + \alpha x_1 + \alpha^2 x_2 x_3 + \alpha x_2 x_4 + \alpha x_2 + \alpha x_3 x_4 \\ &\quad + \alpha x_3 x_5 + x_3 + \alpha x_4^2 + \alpha^2 x_5^2 + \alpha^2 x_5 + \alpha, \end{aligned}$$

$$\begin{aligned} p^{(5)} &= \alpha x_1^2 + x_1 x_4 + \alpha x_1 + \alpha x_2 x_3 + \alpha x_2 x_4 + x_2 x_5 + \alpha x_2 + \alpha x_3^2 + \alpha^2 x_3 x_4 \\ &\quad + x_3 + \alpha^2 x_4^2 + x_4 x_5 + \alpha^2 x_4 + \alpha^2 x_5^2 + \alpha x_5 + \alpha. \end{aligned}$$

First we compute $(n + 1)^2 = 36$ plaintext/ciphertext pairs

\mathbf{z}	$\mathbf{w} = \mathcal{P}(\mathbf{z})$
$(1, 0, 0, 0, \alpha)$	$(0, \alpha^2, \alpha^2, 1, \alpha^2)$
$(\alpha, 1, \alpha^2, \alpha^2, 0)$	$(\alpha, \alpha^2, 1, 1, 1)$
$(\alpha, 1, 0, \alpha, \alpha)$	$(1, 0, \alpha, 1, 1)$
$(\alpha, \alpha^2, \alpha, 1, \alpha^2)$	$(\alpha, \alpha, \alpha, \alpha, \alpha^2)$
$(\alpha^2, 0, \alpha^2, 0, \alpha^2)$	$(\alpha, \alpha, \alpha, 1, 0)$
$(\alpha, 1, 0, 1, 0)$	$(\alpha, \alpha, 0, 1, \alpha)$
$(1, \alpha, 0, 1, \alpha)$	$(\alpha, \alpha^2, 1, \alpha, \alpha^2)$
$(\alpha^2, \alpha^2, \alpha^2, \alpha, 0)$	$(\alpha, 1, \alpha^2, \alpha, 1)$
$(\alpha^2, 0, \alpha^2, 0, 0)$	$(\alpha, 1, 1, \alpha, 0)$
$(\alpha^2, \alpha^2, \alpha, 1, \alpha^2)$	$(\alpha, 1, 0, \alpha, \alpha)$
$(\alpha, 0, 0, \alpha^2, 0)$	$(1, 1, 1, 1, \alpha)$
$(\alpha^2, 0, \alpha, \alpha^2, 1)$	$(\alpha^2, 0, 1, 0, \alpha^2)$
$(0, 1, \alpha^2, \alpha, 1)$	$(1, 0, \alpha, 0, 0)$
$(\alpha, \alpha^2, 1, \alpha^2, \alpha)$	$(\alpha, \alpha^2, \alpha, \alpha^2, 1)$
$(\alpha, \alpha^2, \alpha, 1, 1)$	$(0, 0, \alpha, 1, 1)$
$(\alpha^2, \alpha, \alpha, 1, \alpha)$	$(\alpha^2, \alpha, \alpha^2, \alpha^2, 0)$
$(\alpha^2, 0, 1, \alpha^2, \alpha^2)$	$(0, \alpha^2, \alpha^2, \alpha^2, \alpha)$
$(0, 0, 1, 0, 1)$	$(0, \alpha^2, 0, 1, 0)$
$(\alpha, \alpha, \alpha^2, 1, 1)$	$(1, 1, \alpha^2, 1, 0)$
$(0, 1, \alpha, 0, 0)$	$(\alpha, \alpha, \alpha^2, \alpha^2, 0)$
$(\alpha, \alpha^2, 0, 0, \alpha^2)$	$(1, 1, \alpha, 1, \alpha^2)$
$(0, 1, \alpha, 0, 0)$	$(\alpha, \alpha, \alpha^2, \alpha^2, 0)$
$(0, \alpha, 1, 0, \alpha^2)$	$(1, 0, 0, \alpha^2, 0)$
$(1, 0, 1, \alpha, \alpha^2)$	$(\alpha^2, \alpha, \alpha^2, 1, 0)$
$(\alpha^2, 1, \alpha, 1, 0)$	$(\alpha^2, 0, \alpha, 1, \alpha)$
$(1, 0, 1, 1, 1)$	$(\alpha, 0, \alpha^2, 0, \alpha^2)$
$(0, \alpha^2, \alpha^2, \alpha, \alpha)$	$(1, \alpha^2, \alpha^2, 1, 1)$
$(\alpha^2, \alpha, \alpha^2, \alpha, 1)$	$(\alpha^2, \alpha^2, \alpha^2, \alpha^2, 0)$
$(\alpha^2, 0, \alpha, \alpha, 0)$	$(\alpha^2, 1, 0, \alpha^2, \alpha^2)$
$(1, 0, 0, \alpha^2, \alpha^2)$	$(1, 0, 1, 0, 0)$
$(0, \alpha^2, 1, 1, 0)$	$(\alpha, 0, \alpha, 1, \alpha^2)$
$(1, 0, \alpha^2, 1, \alpha^2)$	$(\alpha, 0, \alpha^2, 0, \alpha)$
$(\alpha, 1, \alpha, 1, \alpha)$	$(1, \alpha, \alpha^2, \alpha, \alpha)$
$(\alpha, \alpha, \alpha, 1, \alpha)$	$(\alpha^2, \alpha, \alpha^2, \alpha, 1)$
$(1, 0, \alpha^2, 0, \alpha)$	$(\alpha, 0, 0, 0, \alpha^2)$
$(\alpha^2, 1, 1, \alpha, 0)$	$(0, 1, 1, \alpha, \alpha^2)$

and substitute them into the linearization equation

$$\sum_{i=1}^5 \sum_{j=1}^5 a_{ij} z_i w_j + \sum_{i=1}^5 b_i z_i + \sum_{j=1}^5 c_j w_j + d$$

By solving the resulting linear system for the 36 variables a_{ij}, b_i, c_j, d we find five linearly independent linearization equations

$$\begin{aligned} L_1 = & z_1 w_1 + z_1 + z_2 w_3 + \alpha z_2 w_4 + \alpha z_2 w_5 + \alpha^2 z_2 + z_3 w_1 + \alpha^2 z_3 w_2 \\ & + z_3 w_3 + z_3 w_4 + \alpha^2 z_3 w_5 + z_3 + z_4 w_1 + \alpha z_4 w_2 + \alpha^2 z_4 w_3 + \alpha z_4 w_5 \\ & + z_4 + z_5 w_1 + \alpha z_5 w_2 + \alpha^2 z_5 w_4 + \alpha^2 z_5 w_5 + \alpha^2 z_5 + w_1 \\ & + w_2 + w_3 + \alpha^2 w_4 + \alpha^2, \end{aligned}$$

$$\begin{aligned} L_2 = & z_1 w_2 + \alpha z_2 w_3 + \alpha z_2 w_5 + \alpha z_2 + z_3 w_1 + z_3 w_2 + \alpha^2 z_3 w_5 + \alpha^2 z_3 \\ & + z_4 w_1 + z_4 w_2 + \alpha z_4 w_3 + \alpha z_4 w_4 + \alpha z_4 + z_5 w_1 + \alpha^2 z_5 w_4 + z_5 w_5 \\ & + \alpha z_5 + \alpha^2 w_1 + \alpha^2 w_3 + w_5 + 1, \end{aligned}$$

$$\begin{aligned} L_3 = & z_1 w_3 + \alpha z_1 w_5 + \alpha^2 z_1 + \alpha^2 z_2 w_3 + z_2 w_4 + z_2 w_5 + \alpha z_2 + \alpha z_3 w_1 \\ & + \alpha z_3 w_2 + z_3 w_3 + z_3 w_4 + \alpha^2 z_3 w_5 + \alpha z_3 + z_4 w_2 + z_4 w_3 + z_4 w_5 \\ & + z_4 + \alpha z_5 w_1 + z_5 w_2 + \alpha^2 z_5 w_3 + \alpha z_5 w_4 + \alpha z_5 w_5 + \alpha z_5 + \alpha w_1 \\ & + \alpha^2 w_2 + \alpha^2 w_3 + \alpha w_4 + \alpha^2 w_5 + 1, \end{aligned}$$

$$\begin{aligned} L_4 = & z_1 w_4 + \alpha^2 z_1 w_5 + \alpha z_1 + \alpha^2 z_2 w_3 + z_2 w_4 + z_2 + z_3 w_1 + \alpha^2 z_3 w_2 \\ & + \alpha z_3 w_3 + \alpha z_3 w_4 + \alpha z_3 w_5 + \alpha^2 z_3 + z_4 w_1 + z_4 w_3 + \alpha^2 z_4 w_4 + \alpha z_4 w_5 \\ & + \alpha z_4 + z_5 w_1 + \alpha z_5 w_2 + \alpha^2 z_5 w_4 + \alpha^2 z_5 w_5 + \alpha^2 z_5 + \alpha^2 w_1 + \alpha w_2 \\ & + \alpha w_3 + w_5 + \alpha^2, \end{aligned}$$

$$\begin{aligned} L_5 = & z_2 w_1 + \alpha^2 z_2 w_4 + \alpha z_2 w_5 + z_3 w_1 + \alpha z_3 w_2 + \alpha z_3 w_3 + \alpha^2 z_3 w_5 \\ & + \alpha^2 z_4 w_2 + \alpha z_4 w_3 + \alpha^2 z_4 w_5 + z_4 + \alpha^2 z_5 w_1 + \alpha^2 z_5 w_2 + z_5 w_4 \\ & + \alpha^2 z_5 w_5 + z_5 + \alpha^2 w_1 + \alpha w_2 + \alpha^2 w_3 + w_4 + \alpha w_5 + \alpha^2 \end{aligned}$$

Until now, the computations have been completely independent of the challenge ciphertext \mathbf{w}^* . So, if we want to find the plaintexts corresponding to multiple challenge ciphertexts $\mathbf{w}_1^*, \dots, \mathbf{w}_\ell^*$, we have to perform this step only once.

Let us assume that we want to find the plaintext corresponding to the ciphertext $\mathbf{w}^* = (\alpha^2, \alpha^2, \alpha, \alpha^2, \alpha^2)$.

By substituting the ciphertext \mathbf{w}^* into these linearization equations, we obtain 4 linear equations in the plaintext variables z_1, \dots, z_5

$$\begin{aligned}
z_1 + z_5 + \alpha^2 &= 0, \\
z_2 + \alpha z_5 + \alpha &= 0, \\
z_3 + \alpha^2 z_5 + \alpha^2 &= 0, \\
z_4 + z_5 + \alpha^2 &= 0.
\end{aligned}$$

We therefore find the four possible plaintexts

\mathbf{z}	$\mathbf{w} = \mathcal{P}(\mathbf{z})$
$(0, \alpha^2, 1, 0, \alpha^2)$	$(0, \alpha, \alpha, \alpha, \alpha^2)$
$(1, 1, \alpha, 1, \alpha)$	$(\alpha, 1, \alpha, 1, \alpha^2)$
$(\alpha, 0, 0, \alpha, 1)$	$(1, 0, \alpha, 0, \alpha^2)$
$(\alpha^2, \alpha, \alpha^2, \alpha^2, 0)$	$(\alpha^2, \alpha^2, \alpha, \alpha^2, \alpha^2)$

The plaintext corresponding to the ciphertext $\mathbf{w}^* = (\alpha^2, \alpha^2, \alpha, \alpha^2, \alpha^2)$ is therefore $\mathbf{z}^* = (\alpha^2, \alpha, \alpha^2, \alpha^2, 0)$.

3.3 Encryption Schemes Based on MI

After the basic Matsumoto-Imai scheme had been broken by Patarin's linearization equations attack (see Sect. 3.2), several variants of the scheme have been suggested which prevent this attack. In this section we concentrate on variants of the Matsumoto-Imai scheme used for encryption. The main design goal of these schemes was to prevent Patarin's linearization equations attack against the basic MI scheme. However, as we will see, the modifications used to do so make the schemes much slower than the original MI scheme.

In this section we first introduce the Internal Perturbation technique of Ding to prevent the linearization equations attack, leading to the PMI scheme. Then, we describe a differential attack of Fouque et al. to break this scheme and finally present the PMI+ scheme which prevents Fouque's attack.

3.3.1 Internal Perturbation

The basic idea of internal perturbation can be described as follows. To hide the structure of the central map \mathcal{F} of a multivariate BigField scheme, we add to \mathcal{F} a perturbation map $\hat{\mathcal{F}}$, consisting of n randomly chosen quadratic polynomials in a small number of variables.

In the following we show how to apply this idea to the Matsumoto-Imai cryptosystem.

Key Generation Let $((S, \mathcal{F}, \mathcal{T}), \mathcal{P})$ be a key pair of the standard Matsumoto-Imai cryptosystem (see Sect. 3.1). For a small integer r , we randomly choose an affine map $\mathcal{Z} : \mathbb{F}^n \rightarrow \mathbb{F}^r$, i.e.

$$\mathcal{Z}(x_1, \dots, x_n) = \underbrace{\begin{pmatrix} z_{11} & z_{12} & \dots & z_{1n} \\ z_{21} & z_{22} & \dots & z_{2n} \\ \vdots & & \ddots & \vdots \\ z_{r1} & z_{r2} & \dots & z_{rn} \end{pmatrix}}_Z \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} + \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_r \end{pmatrix}$$

with a full rank $r \times n$ matrix Z . Additionally, we choose randomly a multivariate quadratic map $\tilde{\mathcal{F}} : \mathbb{F}^r \rightarrow \mathbb{F}^n$, and define the perturbation map $\hat{\mathcal{F}}$ as

$$\hat{\mathcal{F}} = \tilde{\mathcal{F}} \circ \mathcal{Z} : \mathbb{F}^n \rightarrow \mathbb{F}^n.$$

Furthermore, the owner of the private key computes a set

$$P = \{(\lambda, \mu) \mid \tilde{\mathcal{F}}(\lambda) = \mu, \lambda \in \mathbb{F}^r\}.$$

The set P has q^r elements (λ, μ) .

Remark 3.8 The reason for the special construction of the perturbation map $\hat{\mathcal{F}}$ is that, during the decryption process, we have to iterate over all possible output values of $\hat{\mathcal{F}}$. Due to the use of the linear transformation $\mathcal{Z} : \mathbb{F}^n \rightarrow \mathbb{F}^r$, the number of these values is limited to q^r for a small integer r .

With this notation, we can describe the PMI scheme as follows:

Public Key $\mathcal{P} = S \circ (\mathcal{F} + \hat{\mathcal{F}}) \circ \mathcal{T} : \mathbb{F}^n \rightarrow \mathbb{F}^n$.

Private Key The private key of the PMI scheme consists of

- the three maps S , \mathcal{F} and \mathcal{T}
- the linear map $\mathcal{Z} : \mathbb{F}^n \rightarrow \mathbb{F}^r$
- the set of points

$$P = \{(\lambda, \mu) \mid \tilde{\mathcal{F}}(\lambda) = \mu\}.$$

Figure 3.4 illustrates this key generation process.

Encryption To encrypt a message $\mathbf{z} \in \mathbb{F}^n$ using the PMI scheme, one simply computes $\mathbf{w} = \mathcal{P}(\mathbf{z}) \in \mathbb{F}^n$.

Decryption To decrypt a ciphertext $\mathbf{w} \in \mathbb{F}^n$, the owner of the private key performs the following two steps:

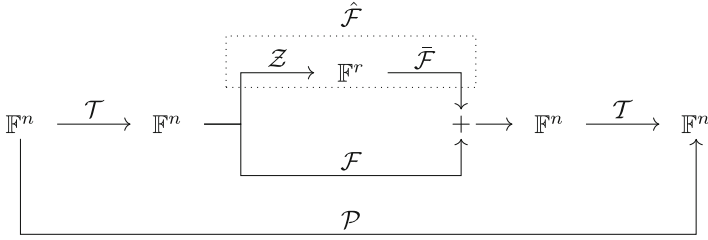


Fig. 3.4 Key generation of PMI

1. Invert the first affine map \mathcal{S} by computing $\mathbf{x} = \mathcal{S}^{-1}(\mathbf{w})$.
2. Compute for each pair $(\lambda, \mu) \in P$, $\mathbf{y}_\lambda = \phi^{-1} \circ \mathcal{F}^{-1} \circ \phi(\mathbf{x} + \mu_\lambda)$ and check, if $\mathcal{Z}(\mathbf{y}_\lambda) = \lambda$ holds.
 - If this is not the case, discard \mathbf{y}_λ .
 - Otherwise, compute the plaintext candidate \mathbf{z}_λ by $\mathbf{z}_\lambda = \mathcal{T}^{-1}(\mathbf{y}_\lambda)$.

It might happen that the decryption process of PMI produces several possible plaintexts \mathbf{z}_λ . In this case, one has to use special techniques (e.g. hash functions, redundancy in the plaintext) to make the decryption unique.

Due to the guessing process in the decryption step (there are approximately q^r possible choices of λ), the decryption process of the PMI scheme is not very efficient.

3.3.2 Differential Attack on PMI

In [7], Fouque proposed an attack against the PMI encryption scheme. The basic idea of this attack is to test which plaintexts produce noise and which do not. This allows to remove the noise from the scheme, which can then be broken by the linearization equations attack (see Sect. 3.2).

Recall that the differential \mathcal{G} of a multivariate quadratic system \mathcal{P} is defined as

$$\mathcal{G}(\mathbf{x}, \mathbf{y}) = \mathcal{P}(\mathbf{x} + \mathbf{y}) - \mathcal{P}(\mathbf{x}) - \mathcal{P}(\mathbf{y}) + \mathcal{P}(\mathbf{0})$$

and is bilinear in \mathbf{x} and \mathbf{y} . When we fix \mathbf{y} to some random value $\bar{\mathbf{y}} \in \mathbb{F}^n$, the map $\mathcal{G}_{\bar{\mathbf{y}}}(\mathbf{x}) = \mathcal{G}(\mathbf{x}, \bar{\mathbf{y}})$ is a linear map in \mathbf{x} .

Definition 3.9 We define the noise kernel \mathcal{K} to be the kernel of the linear part of the transformation $\mathcal{Z} \circ \mathcal{T}$, i.e.

$$\mathcal{K} = \{\mathbf{x} \in \mathbb{F}^n : \mathcal{Z} \circ \mathcal{T}(\mathbf{x}) = \mathbf{0} \in \mathbb{F}^r\}.$$

The term “noise kernel” is inspired by the fact that, for a plaintext $\mathbf{z} \in \mathcal{K}$, the internal perturbation generates no “noise”. Therefore, for such plaintexts, the PMI scheme just works as the standard Matsumoto-Imai encryption scheme.

With the definition of the noise kernel, we can prove

Proposition 3.10 *Let $\mathbf{x} \in \mathcal{K}$. Then*

$$\dim(\ker(\mathcal{G}_{\mathbf{x}})) = \begin{cases} \gcd(n, \theta) & \text{if } \mathbf{x} \neq \mathbf{0} \\ n & \text{if } \mathbf{x} = \mathbf{0} \end{cases}$$

Proof Since $\mathbf{v} \in \mathcal{K}$, we can assume that the central map $\hat{\mathcal{F}}$ of PMI is just the standard Matsumoto-Imai central map. Furthermore, since \mathcal{S} and \mathcal{T} are invertible affine transformations, they do not have any effect on the dimension of the kernel of the differential. So, in order to prove the proposition, we have to count the number of solutions $V \in \mathbb{E}$ of the equation

$$\mathcal{F}(X + V) - \mathcal{F}(X) - \mathcal{F}(V) = 0,$$

where $\mathcal{F}(Y) = Y^{q^\theta+1}$ is the standard Matsumoto-Imai central map and X is a fixed element of \mathbb{E} (note that $\mathcal{F}(0) = 0$).

We have

$$\begin{aligned} 0 &= \mathcal{F}(X + V) - \mathcal{F}(X) - \mathcal{F}(V) \\ &= (X + V)^{q^\theta+1} - X^{q^\theta+1} - V^{q^\theta+1} \\ &= X^{q^\theta} V + X V^{q^\theta}. \end{aligned}$$

or

$$X^{q^\theta} V = X V^{q^\theta} \tag{3.9}$$

Here, we made use of the fact that $q = 0$ holds in \mathbb{E} .

For $X = 0$, clearly any element $V \in \mathbb{E}$ fulfills (3.9) and we have $\dim(\ker(\mathcal{G}_X)) = n$.

For $X \neq 0$, we divide both sides of (3.9) by XV and obtain

$$X^{q^\theta-1} = V^{q^\theta-1}$$

It can be shown that this equation has exactly $q^{\gcd(n, \theta)} - 1$ different non-zero solutions. Adding the trivial solution $V = 0$, the total number of solutions is exactly $q^{\gcd(n, \theta)}$, which implies $\dim(\ker(\mathcal{G}_X)) = \gcd(n, \theta)$. \square

Based on this proposition, we can formulate the following *test* to check the membership of a plaintext $\mathbf{z} \in \mathbb{F}^n$ in the noise kernel: For each $\mathbf{z} \in \mathbb{F}^n$, define the function $\tilde{\mathcal{T}}$ by

$$\tilde{T}(\mathbf{z}) = \begin{cases} 1 & \text{if } \dim(\ker(\mathcal{G}_{\mathbf{z}})) \neq \gcd(n, \theta) \\ 0 & \text{otherwise.} \end{cases}$$

For an element \mathbf{z} of the noise kernel, we have $\dim(\ker(\mathcal{G}_{\mathbf{z}})) = \gcd(n, \theta)$, and therefore $\tilde{T}(\mathbf{z}) = 0$. For an element $\mathbf{z} \notin \mathcal{K}$, the value of $\dim(\ker(\mathcal{G}_{\mathbf{z}}))$ is, due to the randomness of the noise, a random number, and therefore $\tilde{T}(\mathbf{z}) = 1$ with high probability.

We can therefore use this test to identify the elements of the noise kernel and to find a basis of \mathcal{K} .

After having found a basis of the noise kernel \mathcal{K} , we can perform Patarin's linearization equations attack for each affine subspace \mathcal{A} of the form $\mathcal{A} = a + \mathcal{K}$ separately.

3.3.3 Preventing the Differential Attack and PMI+

Surprisingly, preventing Fouque's differential attack on PMI has been proven to be very simple. The only thing we have to do is to add some random quadratic equations to the private key of the PMI scheme (plus modification). If we add s of these random equations, then the public key of the cryptosystem becomes a multivariate quadratic map from \mathbb{F}^n to \mathbb{F}^{n+s} .

The resulting encryption scheme, PMI+, can be described as follows. As in the case of the standard Matsumoto-Imai scheme, we use a finite field \mathbb{F} with q elements and a degree n extension field \mathbb{E} of \mathbb{F} . We have an isomorphism $\phi : \mathbb{F}^n \rightarrow \mathbb{E}$ between the vector space \mathbb{F}^n and the extension field \mathbb{E} .

Furthermore, we use two integers r and s denoting the dimension of the permutation space (see above) and the number of added random equations respectively.

Key Generation In order to generate a key pair for PMI+, Alice chooses

- an MI central map $\mathcal{F} : \mathbb{F}^n \rightarrow \mathbb{F}^n$,
- a linear map $\mathcal{Z} : \mathbb{F}^n \rightarrow \mathbb{F}^r$,
- a perturbation map $\tilde{\mathcal{F}} : \mathbb{F}^r \rightarrow \mathbb{F}^n$,
- a random quadratic map $\mathcal{Q} : \mathbb{F}^n \rightarrow \mathbb{F}^s$,
- two invertible affine maps $\mathcal{S} : \mathbb{F}^{n+s} \rightarrow \mathbb{F}^{n+s}$ and $\mathcal{T} : \mathbb{F}^n \rightarrow \mathbb{F}^n$.

As in the case of the PMI scheme, Alice defines $\hat{\mathcal{F}} = \tilde{\mathcal{F}} \circ \mathcal{Z} : \mathbb{F}^n \rightarrow \mathbb{F}^n$. Furthermore, she sets $\tilde{\mathcal{F}} = \mathcal{F} + \hat{\mathcal{F}}$.

Public Key $\mathcal{P} = \mathcal{S} \circ (\tilde{\mathcal{F}} || \mathcal{Q}) \circ \mathcal{T} : \mathbb{F}^n \rightarrow \mathbb{F}^{n+s}$

Private Key The private key of PMI+ consists of

- the three maps \mathcal{S} , \mathcal{F} and \mathcal{T}
- the linear map \mathcal{Z}
- the set of points $P = \{(\lambda, \mu) | \tilde{\mathcal{F}}(\lambda) = \mu\}$.

- the random quadratic map $\mathcal{Q} : \mathbb{F}^n \rightarrow \mathbb{F}^s$ (to distinguish between false and correct plaintext candidates)

Encryption In order to encrypt a plaintext $\mathbf{z} \in \mathbb{F}^n$, the sender computes $\mathbf{w} = \mathcal{P}(\mathbf{z}) \in \mathbb{F}^{n+s}$ and sends it to the receiver.

Decryption In order to decrypt a ciphertext $\mathbf{w} \in \mathbb{F}^{n+s}$, the receiver performs the following steps.

1. Compute $\mathbf{x}' = \mathcal{S}^{-1}(\mathbf{w}) \in \mathbb{F}^{n+s}$. Define $\mathbf{x} \in \mathbb{F}^n$ to be the vector containing the first n elements of \mathbf{x}' .
2. Compute for each pair $(\lambda, \mu) \in P$, $\mathbf{y}_\lambda = \phi^{-1} \circ \mathcal{F}^{-1} \circ \phi(\mathbf{x} + \mu_\lambda)$ and check, if $\mathcal{Z}(\mathbf{y}_\lambda) = \lambda$ holds.
 - If this is not the case, discard \mathbf{y}_λ .
 - Otherwise, compute the plaintext candidate \mathbf{z}_λ by $\mathbf{z}_\lambda = \mathcal{T}^{-1}(\mathbf{y}_\lambda)$.

In the case of PMI+, we can use the added polynomials of \mathcal{Q} to decide which of the plaintext candidates \mathbf{z}_λ is the correct one. We just evaluate the map $\mathcal{Q} \circ \mathcal{T}$ for each of the plaintext candidates \mathbf{z}_λ . For the correct plaintext, the value $\mathcal{Q} \circ \mathcal{T}(\mathbf{z}_\lambda)$ must be equal to the last s components of the vector \mathbf{x}' . Therefore, for increasing s , the probability of getting several possible plaintexts gets negligible.

As in the case of PMI, the decryption process of PMI+ contains a guessing step. Therefore, the scheme is not very efficient and hardly used in practice.

3.3.4 Toy Example

In the following we illustrate the workflow of the PMI encryption scheme using a toy example. We use $\mathbb{F} = \text{GF}(4)$ as the underlying field and the PMI parameters $(n, r) = (5, 1)$. To generate the extension field \mathbb{E} , we use the irreducible polynomial $f(X) = X^5 + X + \alpha$. We use $\theta = 2$ as the MI-exponent to encrypt a message. Correspondingly, the exponent used during decryption is $h = 662$.

The private key of the scheme consists of the two affine maps $\mathcal{S} : \mathbb{F}^5 \rightarrow \mathbb{F}^5$,

$$\mathcal{S}(x_1, \dots, x_5) = \begin{pmatrix} \alpha^2 & 0 & 1 & 0 & 0 \\ \alpha^2 & 1 & \alpha & 1 & \alpha^2 \\ \alpha & \alpha & 0 & 0 & \alpha \\ 1 & \alpha & 0 & \alpha^2 & \alpha \\ \alpha^2 & 0 & 1 & 1 & \alpha \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ \alpha \\ \alpha \end{pmatrix}$$

and $\mathcal{T} : \mathbb{F}^5 \rightarrow \mathbb{F}^5$,

$$\mathcal{T} \begin{pmatrix} 1 & \alpha & 0 & \alpha & \alpha^2 \\ \alpha & \alpha^2 & \alpha^2 & \alpha^2 & \alpha^2 \\ 1 & 1 & 0 & \alpha^2 & 1 \\ 1 & \alpha & \alpha^2 & 0 & \alpha \\ \alpha & 1 & \alpha^2 & \alpha & \alpha^2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} + \begin{pmatrix} \alpha \\ 0 \\ 0 \\ \alpha^2 \\ 0 \end{pmatrix},$$

as well as the perturbation maps $\mathcal{Z} : \mathbb{F}^5 \rightarrow \mathbb{F}^1$,

$$\mathcal{Z}(x_1, \dots, x_5) = \alpha^2 x_1 + \alpha x_4 + \alpha x_5$$

and $\bar{\mathcal{F}} : \mathbb{F}^1 \rightarrow \mathbb{F}^5$,

$$\bar{\mathcal{F}}(y_1) = \begin{cases} \alpha^2 \\ \alpha^2 y_1^2 \\ \alpha^2 y_1^2 + \alpha y_1 + \alpha^2 \\ y_1^2 + \alpha y_1 + \alpha^2 \\ \alpha y_1^2 + \alpha^2 y_1 + \alpha \end{cases}.$$

Using $\bar{\mathcal{F}}$, we compute the set $P = \{(\lambda, \mu) | \bar{\mathcal{F}}(\lambda) = \mu\}$. We get

λ	$\mu_\lambda = \bar{\mathcal{F}}(\lambda)$
1	$(\alpha^2, \alpha^2, \alpha, 0, \alpha^2)$
α	$(\alpha^2, \alpha, \alpha, \alpha^2, \alpha)$
α^2	$(\alpha^2, 1, \alpha^2, 0, \alpha^2)$
0	$(\alpha^2, 0, \alpha^2, \alpha^2, \alpha)$

In order to compute the public key, we first lift $\mathcal{T}(x_1, \dots, x_5)$ to the extension field, obtaining

$$\begin{aligned} A &= (\alpha x_1 + x_2 + \alpha^2 x_3 + \alpha x_4 + \alpha^2 x_5)X^4 \\ &\quad + (x_1 + \alpha x_2 + \alpha^2 x_3 + \alpha x_5 + \alpha^2)X^3 \\ &\quad + (x_1 + x_2 + \alpha^2 x_4 + x_5)X^2 \\ &\quad + (\alpha x_1 + \alpha^2 x_2 + \alpha^2 x_3 + \alpha^2 x_4 + \alpha^2 x_5)X \\ &\quad + x_1 + \alpha x_2 + \alpha x_4 + \alpha^2 x_5 + \alpha \end{aligned}$$

and compute $B = A^{q^2+1} = A^{17} \bmod f(X)$, obtaining

$$\begin{aligned} B &= (\alpha^2 x_1^2 + x_1 x_2 + x_1 x_3 + \alpha^2 x_1 x_4 + x_2^2 + \alpha x_2 x_4 + x_2 x_5 + \alpha^2 x_2 + \alpha x_3^2 + x_3 x_4 \\ &\quad + \alpha^2 x_3 x_5 + x_3 + \alpha^2 x_4 + \alpha^2 x_5^2 + x_5)X^4 \end{aligned}$$

$$\begin{aligned}
& + (x_1x_3 + \alpha^2x_1x_4 + \alpha^2x_1x_5 + \alpha^2x_2^2 + \alpha x_2x_4 + \alpha^2x_2x_5 + \alpha x_2 + \alpha x_3^2 + x_3x_4 \\
& + x_3x_5 + \alpha x_3 + \alpha^2x_4^2 + x_4x_5 + x_4 + \alpha x_5^2 + 1)X^3 \\
& + (x_1^2 + x_1x_3 + \alpha^2x_1x_4 + \alpha^2x_1x_5 + \alpha x_2x_3 + \alpha x_2x_4 + \alpha^2x_2 + \alpha x_3^2 + \alpha x_3x_4 \\
& + x_3x_5 + x_3 + x_4^2 + x_4x_5 + x_4 + x_5^2 + \alpha^2x_5 + \alpha^2)X^2 \\
& + (x_1^2 + x_1x_2 + x_1x_3 + \alpha x_1x_5 + \alpha x_1 + \alpha x_2^2 + \alpha x_2x_3 + \alpha x_2x_4 + \alpha^2x_2x_5 + \alpha x_3^2 \\
& + x_3x_4 + \alpha x_3 + \alpha^2x_4^2 + \alpha x_4x_5 + x_4 + \alpha x_5^2 + \alpha^2x_5 + \alpha)X \\
& + \alpha x_1^2 + \alpha^2x_1x_2 + \alpha^2x_1x_3 + \alpha^2x_1x_4 + \alpha x_1 + \alpha^2x_2^2 + x_2x_3 + \alpha x_2x_4 + \alpha x_2x_5 \\
& + x_2 + \alpha^2x_3^2 + \alpha^2x_3x_4 + x_3 + x_4x_5 + x_4 + x_5^2 + 1.
\end{aligned}$$

After that, we move B down to the vector space \mathbb{F}^5 and add the permutation map $\hat{F} = \bar{F} \circ \mathcal{Z}$. We get

$$\begin{aligned}
c^{(1)} &= \alpha x_1^2 + \alpha^2x_1x_2 + \alpha^2x_1x_3 + \alpha^2x_1x_4 + \alpha x_1 + \alpha^2x_2^2 + x_2x_3 + \alpha x_2x_4 \\
&+ \alpha x_2x_5 + x_2 + \alpha^2x_3^2 + \alpha^2x_3x_4 + x_3 + x_4x_5 + x_4 + x_5^2 + \alpha, \\
c^{(2)} &= x_1x_2 + x_1x_3 + \alpha x_1x_5 + \alpha x_1 + \alpha x_2^2 + \alpha x_2x_3 + \alpha x_2x_4 + \alpha^2x_2x_5 + \alpha x_3^2 \\
&+ x_3x_4 + \alpha x_3 + x_4^2 + \alpha x_4x_5 + x_4 + \alpha^2x_5 + \alpha, \\
c^{(3)} &= x_1x_3 + \alpha^2x_1x_4 + \alpha^2x_1x_5 + x_1 + \alpha x_2x_3 + \alpha x_2x_4 + \alpha^2x_2 + \alpha x_3^2 + \alpha x_3x_4 \\
&+ x_3x_5 + x_3 + \alpha^2x_4^2 + x_4x_5 + \alpha x_4 + \alpha^2x_5^2, \\
c^{(4)} &= \alpha x_1^2 + x_1x_3 + \alpha^2x_1x_4 + \alpha^2x_1x_5 + x_1 + \alpha^2x_2^2 + \alpha x_2x_4 + \alpha^2x_2x_5 \\
&+ \alpha x_2 + \alpha x_3^2 + x_3x_4 + x_3x_5 + \alpha x_3 + x_4x_5 + \alpha x_4 + x_5^2 + \alpha^2x_5 + \alpha, \\
c^{(5)} &= x_1x_2 + x_1x_3 + \alpha^2x_1x_4 + \alpha x_1 + x_2^2 + \alpha x_2x_4 + x_2x_5 + \alpha^2x_2 + \alpha x_3^2 \\
&+ x_3x_4 + \alpha^2x_3x_5 + x_3 + x_4^2 + \alpha x_4 + \alpha x_5^2 + \alpha.
\end{aligned}$$

Finally, we apply the second affine map \mathcal{S} and obtain the public key $\mathcal{P} = (p^{(1)}, \dots, p^{(5)}) : \mathbb{F}^5 \rightarrow \mathbb{F}^5$ as

$$\begin{aligned}
p^{(1)} &= x_1^2 + \alpha x_1x_2 + \alpha^2x_1x_3 + x_1x_4 + \alpha^2x_1x_5 + \alpha x_2^2 + x_2x_3 + \alpha^2x_2x_4 \\
&+ x_2x_5 + x_3x_5 + \alpha x_3 + \alpha^2x_4^2 + \alpha x_4x_5 + x_4 + 1, \\
p^{(2)} &= \alpha^2x_1^2 + \alpha^2x_1x_3 + \alpha x_1x_4 + x_1 + \alpha x_2x_3 + \alpha^2x_2x_4 + \alpha x_2x_5 + \alpha x_2 \\
&+ \alpha x_3x_4 + x_3x_5 + \alpha x_3 + \alpha^2x_4^2 + \alpha x_4x_5 + \alpha x_4 + \alpha x_5^2, \\
p^{(3)} &= \alpha^2x_1^2 + x_1x_2 + x_1x_3 + \alpha^2x_1x_5 + \alpha^2x_1 + x_2x_3 + \alpha^2x_2x_4 + \alpha^2x_2 + x_3^2 \\
&+ x_3x_4 + x_3x_5 + \alpha^2x_3 + x_4x_5 + \alpha^2x_4 + x_5^2 + x_5 + \alpha,
\end{aligned}$$

$$\begin{aligned}
p^{(4)} &= \alpha^2 x_1^2 + \alpha^2 x_1 x_2 + x_1 x_5 + x_1 + \alpha x_2 x_3 + \alpha^2 x_2 x_4 + \alpha^2 x_2 x_5 + x_2 + \alpha x_3^2 \\
&\quad + \alpha x_3 x_5 + x_3 + x_4 x_5 + x_4 + x_5^2 + \alpha^2 x_5 + 1, \\
p^{(5)} &= \alpha^2 x_1^2 + \alpha^2 x_1 x_4 + \alpha x_1 + \alpha^2 x_2^2 + x_2 x_3 + \alpha x_2 x_4 + \alpha^2 x_2 + x_3^2 + \alpha^2 x_3 x_4 \\
&\quad + x_3 x_5 + \alpha x_3 + x_4^2 + \alpha^2 x_4 x_5 + \alpha x_5^2 + \alpha^2 x_5 + \alpha.
\end{aligned}$$

In order to encrypt a message $\mathbf{z} = (\alpha, 0, 1, 0, \alpha^2) \in \mathbb{F}^5$, we simply evaluate the public key to get the ciphertext

$$\mathbf{w} = \mathcal{P}(\mathbf{z}) = (1, \alpha^2, 0, \alpha, \alpha).$$

In order to decrypt the ciphertext $\mathbf{w} = (1, \alpha^2, 0, \alpha, \alpha) \in \mathbb{F}^5$, we first compute

$$\mathbf{x} = \mathcal{S}^{-1}(\mathbf{w}) = (1, \alpha^2, \alpha, \alpha^2, 1).$$

In order to invert the central map \mathcal{F} , we compute for every possible value of λ the vector $\mathbf{y}_\lambda = \mathcal{F}^{-1}(\mathbf{x} + \mu_\lambda)$, obtaining

λ	μ_λ	$\mathbf{y}_\lambda = \mathcal{F}^{-1}(\mathbf{x} + \mu_\lambda)$	$\mathcal{Z}(\mathbf{y}_\lambda)$
1	$(\alpha^2, \alpha^2, \alpha, 0, \alpha^2)$	$(\alpha, 0, 0, 1, \alpha^2)$	α
α	$(\alpha^2, \alpha, \alpha, \alpha^2, \alpha)$	$(0, 1, 1, 0, 1)$	1
α^2	$(\alpha^2, 1, \alpha^2, 0, \alpha^2)$	$(1, \alpha^2, \alpha, 1, \alpha^2)$	α
0	$(\alpha^2, 0, \alpha^2, \alpha^2, \alpha)$	$(\alpha, \alpha, 1, \alpha^2, \alpha)$	0

Since $\lambda = 0$ is the only value for which $\mathcal{Z}(\mathbf{y}_\lambda) = \lambda$ holds, we get the correct plaintext by

$$\mathbf{z}_0 = \mathcal{T}^{-1}(\mathbf{y}_0) = (\alpha, 0, 1, 0, \alpha^2).$$

As mentioned above, the public key of the internally perturbed Matsumoto-Imai scheme is no longer bijective. Therefore, it might happen that one ciphertext corresponds to several possible plaintexts. In other words, the decryption process might not lead to a unique plaintext.

In order to study this phenomenon in the setting of our toy example, we encrypted all possible $q^n = 1024$ plaintexts with the public key shown above. By doing so, we got 704 different ciphertexts. Of these 704 ciphertexts, 439 decrypted to a unique plaintext, for 215 we found 2 possible plaintexts. 45 of the ciphertexts corresponded to 3 different plaintexts, and 5 decrypted to 4 possible plaintexts.

3.4 Signature Schemes Based on MI

While, in the last section, we looked at variants of the Matsumoto-Imai cryptosystem used for encryption, this section gives an overview of Matsumoto-Imai variants for digital signatures. The main design goal of these constructions is to prevent the linearization equations attack of Patarin (see Sect. 3.2). However, as we will see, creating a secure MI variant for digital signatures slows down the signature generation process significantly.

In this section, we first introduce the minus modification, leading to schemes such as MI-Minus or SFLASH. After that, we describe the differential attack of Dubois et al. on SFLASH and present Ding's projection technique to prevent this attack.

3.4.1 The Minus Variation and SFLASH

A very simple idea to prevent Patarin's linearization equations attack is provided by the minus modification. The idea of this variation is to remove a small number a of equations from the public key. The resulting public key \mathcal{P} is therefore a multivariate quadratic map from \mathbb{F}^n to \mathbb{F}^{n-a} . Therefore, \mathcal{P} is no longer injective as in the case of the basic Matsumoto-Imai cryptosystem, which restricts the resulting scheme to digital signatures.

The MI-Minus cryptosystem (also known as SFLASH) can be described as follows.

Just as in the case of the standard Matsumoto-Imai scheme, we have a finite field \mathbb{F} with q elements and a degree n extension field \mathbb{E} . We define an isomorphism $\phi : \mathbb{F}^n \rightarrow \mathbb{E}$ between the vector space \mathbb{F}^n and the extension field \mathbb{E} . Additionally, we choose for SFLASH a small integer a (number of minus equations).

Key Generation The private key of SFLASH is nearly the same as the private key of the standard Matsumoto-Imai cryptosystem. We have

- *central map*: The central map of SFLASH is just a MI central map: for a parameter θ with $\gcd(q^\theta + 1, q^n - 1) = 1$, we define a univariate map

$$\mathcal{F} : \mathbb{E} \rightarrow \mathbb{E}, \mathcal{F}(Y) = Y^{q^\theta + 1}.$$

We use the extended Euclidean algorithm to compute the signing exponent h with

$$h(1 + q^\theta) = 1 \bmod q^{n-1}.$$

- *affine transformations*: The affine map $\mathcal{T} : \mathbb{F}^n \rightarrow \mathbb{F}^n$ is, just as in the case of MI, a randomly chosen invertible affine transformation over the vector space \mathbb{F}^n . For the map \mathcal{S} , the minus transformation comes into play. So, \mathcal{S} is no longer

an invertible affine transformation, but a randomly chosen map of the form $\mathcal{S} : \mathbb{F}^n \rightarrow \mathbb{F}^{n-a}$ of maximal rank.

Private Key The private key consists of the two maps \mathcal{S} and \mathcal{T} , the signing exponent h and possibly the isomorphism $\phi : \mathbb{F}^n \rightarrow \mathbb{E}$.⁴

Public Key The public key \mathcal{P} is the composed map $\mathcal{S} \circ \phi^{-1} \circ \mathcal{F} \circ \phi \circ \mathcal{T} : \mathbb{F}^n \rightarrow \mathbb{F}^{n-a}$. So, the public key of SFLASH consists of $(n - a)$ multivariate quadratic equations in n variables.

Signature Generation To generate an SFLASH signature for a document $d \in \{0, 1\}^*$, the owner of the private key uses a hash function $\mathcal{H} : \{0, 1\}^* \rightarrow \mathbb{F}^{n-a}$ to compute the hash value $\mathbf{w} = \mathcal{H}(d)$ and performs the following three steps.

1. Find a pre-image $\mathbf{x} \in \mathbb{F}^n$ of the hash value $\mathbf{w} \in \mathbb{F}^{n-a}$ under the affine transformation \mathcal{S} and lift it to the extension field \mathbb{E} , i.e. compute $X = \phi(\mathbf{x})$.
2. Compute $Y = X^h$.
3. Move Y down to the vector space \mathbb{F}^n , i.e. compute $\mathbf{y} = \phi^{-1}(Y)$ and compute the signature $\mathbf{z} \in \mathbb{F}^n$ by

$$\mathbf{z} = \mathcal{T}^{-1}(\mathbf{y}).$$

Signature Verification To check the authenticity of a signature $\mathbf{z} \in \mathbb{F}^n$, the verifier computes $\mathbf{w} = \mathcal{H}(d) \in \mathbb{F}^{n-a}$ and $\mathbf{w}' = \mathcal{P}(\mathbf{z}) \in \mathbb{F}^{n-a}$. If $\mathbf{w}' = \mathbf{w}$ holds, the signature is accepted, otherwise it is rejected.

SFLASH appeared to be a very promising candidate for a post-quantum digital signature scheme and was selected as a standard by the NESSIE project (New European Schemes for Signatures, Identification and Encryption). However, it was eventually broken by a differential attack of Dubois et al. (see Sect. 3.4.3).

3.4.2 Toy Example

In the following, we illustrate the workflow of the SFLASH signature scheme using a toy example. We choose $\mathbb{F} = GF(4)$ as the underlying field and $(n, a) = (5, 2)$. To generate the extension field $\mathbb{E} = GF(4^5)$, we use the irreducible polynomial $f(X) = X^5 + X + \alpha$. Furthermore, we use $\theta = 2$ as the Matsumoto-Imai parameter for the verification. The corresponding signing exponent is $h = 662$.

The private key of the scheme consists of two affine maps $\mathcal{S} : \mathbb{F}^5 \rightarrow \mathbb{F}^3$,

⁴Similar to the case of the MI scheme, the isomorphism ϕ can be a public parameter or part of the private key.

$$\mathcal{S}(x_1, \dots, x_5) = \begin{pmatrix} 1 & \alpha^2 & 0 & \alpha & 1 \\ 1 & 1 & 0 & 1 & \alpha^2 \\ \alpha & 0 & \alpha & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} + \begin{pmatrix} 0 \\ \alpha \\ 1 \end{pmatrix}$$

and $\mathcal{T} : \mathbb{F}^5 \rightarrow \mathbb{F}^5$,

$$\mathcal{T}(x_1, \dots, x_5) = \begin{pmatrix} \alpha^2 & \alpha^2 & \alpha^2 & \alpha & 1 \\ 1 & 1 & \alpha & 0 & \alpha^2 \\ 0 & 0 & \alpha^2 & 1 & \alpha \\ 0 & \alpha^2 & \alpha^2 & \alpha & \alpha \\ 0 & 1 & 1 & \alpha^2 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} + \begin{pmatrix} \alpha \\ 1 \\ \alpha^2 \\ 1 \\ 0 \end{pmatrix}.$$

In order to compute the public key, we first lift the map \mathcal{T} to the extension field, obtaining

$$\begin{aligned} A = & (\alpha^2 x_1 + \alpha^2 x_2 + \alpha^2 x_3 + \alpha x_4 + x_5 + \alpha) X^4 \\ & + (x_1 + x_2 + \alpha x_3 + \alpha^2 x_5 + 1) X^3 \\ & + (\alpha^2 x_3 + x_4 + \alpha x_5 + \alpha^2) X^2 \\ & + (\alpha^2 x_2 + \alpha^2 x_3 + \alpha x_4 + \alpha x_5 + 1) X \\ & + x_2 + x_3 + \alpha^2 x_4 + x_5 \end{aligned}$$

and compute $B = A^{q^\theta+1} = A^{17} \bmod f(X)$. We obtain

$$\begin{aligned} B = & (x_1^2 + x_1 x_2 + \alpha^2 x_1 x_3 + x_1 x_4 + \alpha^2 x_1 + \alpha x_2^2 + \alpha x_2 x_3 + x_2 x_5 + x_2 \\ & + \alpha^2 x_3 x_4 + x_3 x_5 + x_3 + \alpha^2 x_4^2 + x_4 x_5 + x_4 + x_5^2 + \alpha^2 x_5) X^4 \\ & + (\alpha^2 x_1^2 + x_1 x_2 + \alpha x_1 x_3 + x_1 x_4 + \alpha^2 x_1 x_5 + x_2 x_3 + x_2 x_4 + x_2 + x_3^2 \\ & + \alpha^2 x_3 x_4 + \alpha x_3 x_5 + \alpha^2 x_3 + \alpha x_4 x_5 + \alpha x_5) X^3 \\ & + (\alpha^2 x_1^2 + \alpha^2 x_1 x_2 + x_1 x_3 + \alpha x_1 x_4 + x_1 x_5 + \alpha x_1 + \alpha^2 x_2 x_4 + \alpha x_2 x_5 \\ & + x_2 + \alpha^2 x_3^2 + \alpha x_3 x_4 + \alpha x_3 x_5 + \alpha x_4^2 + x_4 x_5 + \alpha x_5^2 + \alpha x_5 + \alpha^2) X^2 \\ & + (\alpha^2 x_1^2 + \alpha x_1 x_2 + \alpha x_1 x_3 + \alpha^2 x_1 x_5 + x_2^2 + \alpha x_2 x_3 + \alpha^2 x_2 x_4 + x_2 x_5 \\ & + \alpha x_2 + x_3^2 + \alpha^2 x_3 x_5 + \alpha^2 x_4^2 + x_4 x_5 + \alpha^2 x_5 + \alpha) X \\ & + x_1^2 + \alpha x_1 x_3 + \alpha^2 x_1 x_5 + \alpha^2 x_1 + x_2^2 + \alpha x_2 x_4 + x_2 x_5 + \alpha^2 x_2 + \alpha^2 x_3^2 \\ & + x_3 x_5 + x_3 + \alpha x_4^2 + x_4 x_5 + \alpha x_4 + x_5^2 + \alpha^2 x_5. \end{aligned}$$

Finally, we move B down to the vector space \mathbb{F}^5 and apply the affine transformation S to get the public key $\mathcal{P} = (p^{(1)}, p^{(2)}, p^{(3)}) : \mathbb{F}^5 \rightarrow \mathbb{F}^3$,

$$\begin{aligned} p^{(1)} &= \alpha^2 x_1^2 + \alpha x_1 x_2 + \alpha^2 x_1 x_3 + \alpha^2 x_1 x_4 + x_2 x_3 + \alpha x_2 x_4 + \alpha^2 x_2 x_5 + x_2 \\ &\quad + \alpha x_3^2 + \alpha x_3 x_4 + x_3 x_5 + x_3 + \alpha^2 x_4^2 + \alpha^2 x_4 + x_5 + 1, \\ p^{(2)} &= \alpha x_1^2 + \alpha x_1 x_4 + \alpha^2 x_1 x_5 + x_1 + x_2^2 + \alpha x_2 x_3 + \alpha^2 x_2 x_5 + \alpha^2 x_2 \\ &\quad + \alpha^2 x_3^2 + x_3 x_4 + \alpha^2 x_3 x_5 + x_3 + \alpha^2 x_4^2 + x_4 x_5 + x_4 + \alpha x_5^2, \\ p^{(3)} &= \alpha x_1^2 + \alpha x_1 x_3 + \alpha x_1 x_4 + \alpha^2 x_1 x_5 + x_1 + \alpha x_2 x_3 + \alpha x_2 x_4 + \alpha x_2 \\ &\quad + \alpha^2 x_3 + \alpha^2 x_4^2 + x_4 x_5 + \alpha x_4 + x_5. \end{aligned}$$

In order to generate a signature for the message (or hash value) $\mathbf{w} = (1, 0, 0) \in \mathbb{F}^3$, we first compute a pre-image $\mathbf{x} \in \mathbb{F}^5$ of \mathbf{w} under the affine map S , e.g. $\mathbf{x} = (\alpha^2, \alpha^2, \alpha^2, 0, \alpha^2)$ and lift it to the extension field. We get

$$B = \alpha^2 X^4 + \alpha X^2 + \alpha^2 X + \alpha^2.$$

Next, we invert the Matsumoto-Imai central map by computing $A = B^h = B^{662} \bmod f(X)$, obtaining

$$A = X^4 + \alpha.$$

By moving A down to the vector space \mathbb{F}^5 and inverting the affine map \mathcal{T} , we get the signature

$$\mathbf{z} = (\alpha^2, \alpha^2, \alpha, \alpha^2, \alpha^2) \in \mathbb{F}^5.$$

To check the authenticity of the signature $\mathbf{z} \in \mathbb{F}^5$, we just compute

$$\mathbf{w}' = \mathcal{P}(\mathbf{z}) = (1, 0, 0).$$

Since the result is equal to the message \mathbf{w} , the signature is accepted.

3.4.3 Differential Attack on SFLASH

In [5, 6] Dubois et al. proposed two attacks against the SFLASH signature scheme. Both attacks use symmetries in the differential of the SFLASH public key to recover the missing equations. After that one can use Patarin's linearization equations attack (see Sect. 3.2) to forge signatures.

The first attack [6] uses so called skew symmetric maps and works in the case of $\gcd(\theta, n) > 1$. The second attack [5] uses the multiplicative symmetry of the differential and works for all values of n and θ .

Recall from Chap. 2 that the differential of a multivariate quadratic system \mathcal{P} is given as

$$\mathcal{DP}(\mathbf{x}, \mathbf{y}) = \mathcal{P}(\mathbf{x} + \mathbf{y}) - \mathcal{P}(\mathbf{x}) - \mathcal{P}(\mathbf{y}) + \mathcal{P}(\mathbf{0}).$$

The differential of a multivariate quadratic system is homogeneous quadratic, bilinear and symmetric in \mathbf{x} and \mathbf{y} .

3.4.3.1 Skew Symmetric Maps

The first attack [6] looks for so called skew symmetric maps of the differential \mathcal{DP} of the public key of SFLASH. These are linear maps $M : \mathbb{F}^n \rightarrow \mathbb{F}^n$ such that

$$\mathcal{DP}(M(\mathbf{x}), \mathbf{y}) + \mathcal{DP}(\mathbf{x}, M(\mathbf{y})) = 0.$$

For a generic system \mathcal{P} , this equation is only fulfilled for trivial matrices M , namely $M = a \cdot 1_n$ where $a \in \mathbb{F}$ and 1_n being the $n \times n$ identity matrix. However, for the case of SFLASH, there are more solutions. To see this, we look at the MI central map $\mathcal{F}(X) = X^{q^\theta+1}$. The differential \mathcal{DF} has the form

$$\mathcal{DF}(X, Y) = X^{q^\theta} Y + XY^{q^\theta}. \quad (3.10)$$

We see that, for $\bar{F} = \phi^{-1} \circ \mathcal{F} \circ \phi$, the equation

$$\mathcal{D}\bar{F}(M(\mathbf{x}), \mathbf{y}) + \mathcal{D}\bar{F}(\mathbf{x}, M(\mathbf{y})) = 0$$

holds for every matrix M_ζ which can be viewed as a multiplication by $\zeta \in \mathbb{E}$ with

$$\zeta^{q^\theta} + \zeta = 0. \quad (3.11)$$

In [6] it was shown that these are the only solutions for M . Obviously, the matrices M_ζ form a linear space T . To determine the dimension of T , we look at the number of possible values for ζ in (3.11). ζ is a $(q^\theta - 1)$ -th root of unity and the number of these elements is given by $\gcd(q^\theta - 1, q^n - 1) = q^{\gcd(\theta, n)} - 1$. The dimension of the linear space T is therefore $d = \gcd(\theta, n)$. In the following we assume that $d > 1$ holds.

It can be shown that, even after composing \mathcal{F} with the affine transformations \mathcal{S} and \mathcal{T} and after removing some of the public equations, the skew symmetry of the differential still holds. In other words, there exist non trivial matrices N_ζ such that

$$\mathcal{DP}(N_\zeta(\mathbf{x}), \mathbf{y}) + \mathcal{DP}(\mathbf{x}, N_\zeta(\mathbf{y})) = 0 \quad (3.12)$$

holds. By substituting a large number of pairs (\mathbf{x}, \mathbf{y}) into (3.12), it is possible to recover the linear space spanned by the matrices N_ζ .

Now, Dubois et al. showed that, besides of \mathcal{P} , also $\mathcal{P} \circ N_\zeta$ forms a valid SFLASH public key (for the same private key). By adding a of the polynomials of $\mathcal{P} \circ N_\zeta$ to \mathcal{P} , it is therefore possible to extend the SFLASH public key \mathcal{P} to a full Matsumoto-Imai public key. We can then use Patarin's linearization equations attack (see Sect. 3.2) to forge signatures.

3.4.3.2 The Multiplicative Symmetry

In the second attack [5], Dubois et al. looked at the so called multiplicative symmetry of the differential. In particular, they looked at elements $\zeta \in \mathbb{E}$ for which

$$\mathcal{DF}(\zeta X, Y) + \mathcal{DF}(X, \zeta Y) = (\zeta^{q^\theta} + \zeta)\mathcal{DF}(X, Y)$$

holds. For the special form of the differential of the Matsumoto-Imai central map (3.10), it is easy to see that this equation holds for all $\zeta \in \mathbb{E}$.

It can be shown that this symmetry still holds after composing \mathcal{F} with linear maps \mathcal{S} and \mathcal{T} and removing some of the public equations. In particular, we have for the differential of the SFLASH public key \mathcal{P}_Π

$$\mathcal{DP}_\Pi(N_\zeta(\mathbf{x}), \mathbf{y}) + \mathcal{DP}_\Pi(\mathbf{x}, N_\zeta(\mathbf{y})) = \underbrace{\mathcal{S}_\Pi \circ M_{L(\zeta)} \circ \mathcal{S}^{-1}}_{\Delta(L(\zeta))} \mathcal{DP}(\mathbf{x}, \mathbf{y}) \quad (3.13)$$

for some unknown matrices $N_\zeta = \mathcal{T}^{-1} \circ M_\zeta \circ \mathcal{T}$ and $M_{L(\zeta)}$ being the matrix representing the multiplication by $(\zeta^{q^\theta} + \zeta)$. Note that the right hand side of this equation is a linear combination of the components of \mathcal{DP} , which are partially known to the attacker. The goal of the attack is now to identify matrices N_ζ for which the right hand side of (3.13) is a linear combination of the known first $(n - a)$ components of \mathcal{DP} .

For this, Dubois et al. defined the vector spaces V , generated by the n components of \mathcal{DP} , and V_Π which is generated by the $(n - a)$ known components of \mathcal{DP}_Π . It is clear that V_Π is an $(n - a)$ dimensional subspace of the n dimensional space V . We now consider the equation

$$S_M(\mathbf{x}, \mathbf{y}) = \mathcal{DP}_\Pi(M(\mathbf{x}), \mathbf{y}) + \mathcal{DP}(\mathbf{x}, M(\mathbf{y})),$$

where S_M is a tuple of $(n - a)$ symmetric bilinear forms. For most choices of M (those, who do not correspond to a multiplication with some $\zeta \in \mathbb{E}$), the equation will not have a solution. However, if M is one of the matrices N_ζ , the components of S_M are elements of the space V . It is very unlikely that they are elements of V_Π ,

but, for some choices of ζ , at least the first k components of S_{N_ζ} will be elements of V_Π .

It can be shown that, for $k \geq 3$, these matrices N_ζ form a linear space of dimension $n - ka$. By testing a large number of matrices M and checking if the first 3 coordinates of the corresponding matrices S_M are contained in V_Π , we can therefore find non trivial matrices N_ζ with the desired property.

The last step of the attack consists of recovering the missing a polynomials to get a full Matsumoto-Imai public key. For this, we compose the SFLASH public key \mathcal{P}_Π with one of the previously found transformations N_ζ . We get

$$\mathcal{P}'_\Pi(\mathbf{x}) = \mathcal{P}_\Pi \circ N_\zeta(\mathbf{x}) = S_\Pi \circ M_{\zeta^{q^\theta+1}} \circ \mathcal{F} \circ \mathcal{T}(\mathbf{x}).$$

So, \mathcal{P}_Π and \mathcal{P}'_Π contain together $2 \cdot (n - a)$ linear combinations of the quadratic polynomials $\mathcal{F} \circ \mathcal{T}$. With high probability, we can construct from this a full rank Matsumoto-Imai public key. By applying Patarin's linearization equations attack (see Sect. 3.2) on this key, we can therefore forge signatures.

3.4.4 Preventing the Differential Attack and PFLASH

In [4] Ding et al. proposed a technique to prevent the differential attack against SFLASH: Projection. The idea of the PFLASH (Projected FLASH) signature scheme is to project the SFLASH public key to a subspace. In particular, one fixes a small number (say s) of the public key variables to 0, which makes the public key \mathcal{P} to be a multivariate quadratic map from \mathbb{F}^{n-s} to \mathbb{F}^{n-a} . The scheme can be described as follows.

Just as in the case of the standard Matsumoto-Imai scheme we have a finite field \mathbb{F} with q elements, a degree n extension field \mathbb{E} of \mathbb{F} and an isomorphism $\phi : \mathbb{F}^n \rightarrow \mathbb{E}$ between the vector space \mathbb{F}^n and the extension field \mathbb{E} . Additionally we use two integers a and s denoting the number of minus equations and the number of projected variables respectively.

Key Generation The components of a PFLASH private key are chosen as follows.

- *central map*: central MI map $\mathcal{F} : \mathbb{E} \rightarrow \mathbb{E} : Y \mapsto Y^{q^\theta+1}$ for an MI exponent θ with $\gcd(q^{n-1}, q^{\theta+1}) = 1$.
- *signing exponent*: Use the extended Euclidean algorithm to compute $h \in \{2, \dots, n-1\}$ such that $h(1 + q^\theta) = 1 \pmod{(q^n - 1)}$.
- *affine transformations*: $\mathcal{S} : \mathbb{F}^n \rightarrow \mathbb{F}^{n-a}$, $\mathcal{T} : \mathbb{F}^n \rightarrow \mathbb{F}^n$.

Private Key The private key consists of the two maps \mathcal{S} and \mathcal{T} , the signing exponent h and possibly the isomorphism ϕ .

Public Key The public key is the composed map $\mathcal{P} = \Pi_{n-s} \circ \mathcal{S} \circ \phi^{-1} \circ \mathcal{F} \circ \phi \circ \mathcal{T} : \mathbb{F}^{n-s} \rightarrow \mathbb{F}^{n-a}$, where Π_{n-s} is the projection to the first $n - s$ components.

Signature Generation In order to generate a signature for a document $d \in \{0, 1\}^*$, the signer uses a hash function $\mathcal{H} : \{0, 1\}^* \rightarrow \mathbb{F}^{n-a}$ to compute the hash value $\mathbf{w} = \mathcal{H}(d) \in \mathbb{F}^{n-a}$ and performs the following steps

1. Compute a pre-image $\mathbf{x} \in \mathbb{F}^n$ of \mathbf{w} under the affine transformation \mathcal{S} and lift it to the extension field \mathbb{E} , i.e. compute $X = \phi(\mathbf{x})$.
2. Compute $Y = X^h$.
3. Move Y down to the vector space \mathbb{F}^n , obtaining \mathbf{y} , and compute $\mathbf{z} = \mathcal{T}^{-1}(\mathbf{y}) \in \mathbb{F}^n$. If the last s coordinates of $\mathbf{z} = (z_1, \dots, z_n)$ are zero, output the signature $\sigma = (z_1, \dots, z_{n-s}) \in \mathbb{F}^{n-s}$. Otherwise, one has to choose another pre-image of \mathbf{w} under \mathcal{S} and try again.

Signature Verification To check the authenticity of a signature $\sigma \in \mathbb{F}^{n-s}$, the verifier computes $\mathbf{w} = \mathcal{H}(d) \in \mathbb{F}^{n-a}$ and $\mathbf{w}' = \mathcal{P}(\sigma)$. If $\mathbf{w}' = \mathbf{w}$ holds, the signature is accepted, otherwise rejected.

Step 3 of the signature generation process contains a guessing step: If the last s components of the SFLASH signature are not equal to zero, we have to compute a new signature. This means that we have to perform this step about q^s times, which leads to a slow down of the signature generation process by a factor of q^s .

In [4] it is shown by computer experiments that, even for small values of s , the projection efficiently breaks both the skew-symmetric and the multiplicative symmetry of the differential of the Matsumoto-Imai public key. Therefore, the differential attacks described in Sect. 3.4.3 can not be used against PFLASH.

3.4.5 Toy Example

We continue the SFLASH toy example from Sect. 3.4.1. We had $\mathbb{F} = \text{GF}(4)$, $\theta = 2$, $h = 662$ and $(n, a) = (5, 2)$. The extension field \mathbb{E} was $\text{GF}(4^5)$ using the irreducible polynomial $g(X) = X^5 + X + \alpha$. Furthermore, we set the projection parameter s to 1. The affine maps \mathcal{S} and \mathcal{T} were given as

$$\mathcal{S}(x_1, \dots, x_5) = \begin{pmatrix} 1 & \alpha^2 & 0 & \alpha & 1 \\ 1 & 1 & 0 & 1 & \alpha^2 \\ \alpha & 0 & \alpha & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} + \begin{pmatrix} 0 \\ \alpha \\ 1 \end{pmatrix}$$

and

$$\mathcal{T}(x_1, \dots, x_5) = \begin{pmatrix} \alpha^2 & \alpha^2 & \alpha^2 & \alpha & 1 \\ 1 & 1 & \alpha & 0 & \alpha^2 \\ 0 & 0 & \alpha^2 & 1 & \alpha \\ 0 & \alpha^2 & \alpha^2 & \alpha & \alpha \\ 0 & 1 & 1 & \alpha^2 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} + \begin{pmatrix} \alpha \\ 1 \\ \alpha^2 \\ 1 \\ 0 \end{pmatrix}.$$

This lead to the SFLASH public key $\bar{\mathcal{P}} = (\bar{p}^{(1)}, \bar{p}^{(2)}, \bar{p}^{(3)})$ with

$$\begin{aligned} \bar{p}^{(1)} &= \alpha^2 x_1^2 + \alpha x_1 x_2 + \alpha^2 x_1 x_3 + \alpha^2 x_1 x_4 + x_2 x_3 + \alpha x_2 x_4 + \alpha^2 x_2 x_5 + x_2 \\ &\quad + \alpha x_3^2 + \alpha x_3 x_4 + x_3 x_5 + x_3 + \alpha^2 x_4^2 + \alpha^2 x_4 + x_5 + 1, \\ \bar{p}^{(2)} &= \alpha x_1^2 + \alpha x_1 x_4 + \alpha^2 x_1 x_5 + x_1 + x_2^2 + \alpha x_2 x_3 + \alpha^2 x_2 x_5 + \alpha^2 x_2 \\ &\quad + \alpha^2 x_3^2 + x_3 x_4 + \alpha^2 x_3 x_5 + x_3 + \alpha^2 x_4^2 + x_4 x_5 + x_4 + \alpha x_5^2, \\ \bar{p}^{(3)} &= \alpha x_1^2 + \alpha x_1 x_3 + \alpha x_1 x_4 + \alpha^2 x_1 x_5 + x_1 + \alpha x_2 x_3 + \alpha x_2 x_4 + \alpha x_2 \\ &\quad + \alpha^2 x_3 + \alpha^2 x_4^2 + x_4 x_5 + \alpha x_4 + x_5. \end{aligned}$$

To get the corresponding PFLASH public key, we set the last variable x_5 to 0. Therefore we get the public key $\mathcal{P} = (p^{(1)}, p^{(2)}, p^{(3)})$ with

$$\begin{aligned} p^{(1)} &= \alpha^2 x_1^2 + \alpha x_1 x_2 + \alpha^2 x_1 x_3 + \alpha^2 x_1 x_4 + x_2 x_3 + \alpha x_2 x_4 + x_2 \\ &\quad + \alpha x_3^2 + \alpha x_3 x_4 + \alpha^2 x_4^2 + \alpha^2 x_4 + 1, \\ p^{(2)} &= \alpha x_1^2 + \alpha x_1 x_4 + x_1 + x_2^2 + \alpha x_2 x_3 + \alpha^2 x_2 + \alpha^2 x_3^2 + x_3 x_4 \\ &\quad + x_3 + \alpha^2 x_4^2 + x_4, \\ p^{(3)} &= \alpha x_1^2 + \alpha x_1 x_3 + \alpha x_1 x_4 + x_1 + \alpha x_2 x_3 + \alpha x_2 x_4 + \alpha x_2 \\ &\quad + \alpha^2 x_3 + \alpha^2 x_4^2 + \alpha x_4. \end{aligned}$$

We want to compute a PFLASH signature for the message/hash value $\mathbf{w} = (0, \alpha^2, \alpha^2) \in \mathbb{F}^3$.

For this, we compute a pre-image \mathbf{x} of \mathbf{w} under the affine map \mathcal{S} . We get

$$\mathbf{x}_1 = (0, \alpha, \alpha^2, 1, \alpha^2) \in \mathbb{F}^5.$$

Lifting \mathbf{x}_1 to the extension field \mathbb{E} and inverting the central map \mathcal{F} yields

$$Y_1 = X^4 + \alpha^2 X^3 + \alpha X^2 + 1.$$

Moving Y_1 down to the vector space \mathbb{F}^5 and inverting the second affine map \mathcal{T} yields

$$\mathbf{z}_1 = (1, \alpha, 0, \alpha, \alpha) \in \mathbb{F}^5.$$

Since the last coordinate of \mathbf{z}_1 is different from 0, we have to choose another pre-image of \mathbf{w} and try again.

This time we get

$$\mathbf{x}_2 = \mathcal{S}^{-1}(\mathbf{w}) = (0, \alpha, \alpha, 0, 1).$$

Lifting \mathbf{x}_2 to the extension field and inverting \mathcal{F} yields

$$Y_2 = \alpha^2 X^3 + \alpha^2 X^2 + \alpha^2 X + \alpha^2.$$

Moving Y_2 down to the vector space \mathbb{F}^5 and inverting \mathcal{T} yields

$$\mathbf{z}_2 = (0, 0, 1, \alpha^2, 0) \in \mathbb{F}^5.$$

This time, the last coordinate of \mathbf{z}_2 is zero. So, we can discard it and obtain the PFLASH signature

$$\sigma = (0, 0, 1, \alpha^2) \in \mathbb{F}^4.$$

During verification, we find

$$\mathcal{P}(\sigma) = (0, \alpha^2, \alpha^2) = \mathbf{w}$$

and therefore accept the signature.

References

1. N. Courtois, L. Goubin, J. Patarin, Sflash: primitive specification, 2002. <https://www.cosic.esat.kuleuven.be/nessie>
2. J. Ding, A new variant of the matsumoto-imai cryptosystem through perturbation, in *PKC 2004*. Lecture Notes in Computer Science, vol. 2947 (Springer, Heidelberg, 2004), pp. 305–318
3. J. Ding, J.E. Gower, Inoculating multivariate schemes against differential attacks, in *PKC 2006*. Lecture Notes in Computer Science, vol. 3958 (Springer, Heidelberg, 2006), pp. 290–301
4. J. Ding, B.-Y. Yang, V. Dubois, C.-M. Cheng, O. Chen, Breaking the symmetry: a way to resist the new differential attack, 2007. <http://eprint.iacr.org/2007/366>
5. V. Dubois, P. Fouque, A. Shamir, J. Stern, Practical cryptanalysis of SFLASH, in *CRYPTO 2007*. Lecture Notes in Computer Science, vol. 4622 (Springer, Berlin, 2007), pp. 1–12
6. V. Dubois, P. Fouque, J. Stern, Cryptanalysis of SFLASH with slightly modified parameters, in *EUROCRYPT 2007*. Lecture Notes in Computer Science, vol. 4515 (Springer, Heidelberg, 2007), pp. 264–275
7. P.A. Fouque, L. Granboulan, J. Stern, Differential cryptanalysis for multivariate schemes, in *EUROCRYPT 2005*. Lecture Notes in Computer Science, vol. 3494, pp. 249–265 (Springer, Heidelberg, 2005)

8. T. Matsumoto, H. Imai, Public quadratic polynomial-tuples for efficient signature verification and message-encryption, in *EUROCRYPT 1988*. Lecture Notes in Computer Science, vol. 330 (Springer, Heidelberg, 1988), pp. 419–553
9. J. Patarin, Cryptanalysis of the Matsumoto and Imai public key scheme of Eurocrypt 88, in *CRYPTO 1995*. Lecture Notes in Computer Science, vol. 963 (Springer, Berlin, 1995), pp. 248–261
10. B. Preneel, The NESSIE Project: Towards New Cryptographic Algorithms, 2000. www.cryptoneessie.org

Chapter 4

Hidden Field Equations



Abstract This chapter deals with the Hidden Field Equations (HFE) cryptosystem and its variants. We start by an introduction of the basic HFE cryptosystem and study its security against direct and rank attacks. Furthermore, we give an overview of the various HFE variants for encryption and digital signatures. In the area of encryption schemes we study here the IPHFE+ and the ZHFE schemes, while, in the area of signature schemes, we analyze the HFEv- signature scheme and its extension Gui, which produces the shortest signatures of all currently existing schemes.

After the Matsumoto–Imai cryptosystem had been broken by his linearization equations attack, Patarin had the idea of the Hidden Field Equations (HFE) cryptosystem [10]. The basic idea of HFE is to add additional terms to the Matsumoto–Imai central map while ensuring that

1. the resulting public key stays quadratic and
2. the central map is still efficiently invertible.

This idea uses the Frobenius automorphism, which ensures that, over a finite field of characteristic q , every polynomial $f(X) = X^{q^i}$ ($i \in \mathbb{N}_0$) is linear. Therefore, a univariate polynomial map $\mathcal{F} : \mathbb{E} \rightarrow \mathbb{E}$ over a degree n extension field \mathbb{E} of \mathbb{F} of the form

$$\mathcal{F}(X) = \sum_{i,j} X^{q^i + q^j}$$

leads to a quadratic map $\tilde{\mathcal{F}} = \phi^{-1} \circ \mathcal{F} \circ \phi$ over the base field \mathbb{F} . By adding these terms to the MI central map, Patarin not only defended the scheme against the Linearization Equations attacks, but also increased its security against direct and rank attacks. However, as we will show in this chapter, the basic HFE scheme is still vulnerable against these attacks.

Therefore, the HFE scheme is nowadays mainly used as a basis for constructing more advanced schemes. Especially in the area of digital signature schemes, the HFEv- scheme and its extensions Gui [11] and GeMSS [3] are very promising.

This chapter is organized as follows: In Sect. 4.1 we introduce the basic HFE cryptosystem and illustrate its workflow using a toy example, while Sect. 4.2 deals with the main attacks against HFE and the security of the scheme. Section 4.3 describes encryption schemes based on HFE and, in Sect. 4.4, we present signature schemes based on HFE.

4.1 The Basic HFE Cryptosystem

In this section we describe the basic HFE cryptosystem as proposed by Patarin in [10]. As the Matsumoto–Imai scheme of the previous chapter, the scheme belongs to the BigField family of multivariate schemes, which means that it uses a degree n extension field \mathbb{E} of \mathbb{F} as well as an isomorphism $\phi : \mathbb{F}^n \rightarrow \mathbb{E}$. In its basic form, the HFE cryptosystem can be used both as an encryption and signature scheme.

The central map of the HFE cryptosystem is a univariate polynomial map $\mathcal{F} : \mathbb{E} \rightarrow \mathbb{E}$ of the form

$$\mathcal{F}(X) = \sum_{i,j=0}^{q^i+q^j \leq D} \alpha_{ij} X^{q^i+q^j} + \sum_{i=0}^{q^i \leq D} \beta_i X^{q^i} + \gamma \quad (4.1)$$

with coefficients α_{ij} , β_i and γ randomly chosen from \mathbb{E} . Due to the special form of \mathcal{F} , the map $\tilde{\mathcal{F}} = \phi^{-1} \circ \mathcal{F} \circ \phi$ is a quadratic map over the vector space \mathbb{F}^n . In order to hide the structure of \mathcal{F} in the public key, $\tilde{\mathcal{F}}$ is composed with two affine maps \mathcal{S} and $\mathcal{T} : \mathbb{F}^n \rightarrow \mathbb{F}^n$. Therefore, the public key \mathcal{P} of the scheme is given as

$$\mathcal{P} = \mathcal{S} \circ \tilde{\mathcal{F}} \circ \mathcal{T}$$

and is a quadratic map from \mathbb{F}^n to \mathbb{F}^n .

Remark 4.1 The parameter D in (4.1) is introduced to ensure the efficient inversion of the map \mathcal{F} . During the decryption or signature generation process (see below), we have to invert \mathcal{F} , which corresponds to the solution of a univariate polynomial of degree D . The complexity of this step highly depends on D , due to which D can not be chosen too large.

The *private key* of the scheme consists of the three maps \mathcal{S} , \mathcal{F} and \mathcal{T} (and possibly the isomorphism ϕ).

The *public key* is the multivariate quadratic map $\mathcal{P} : \mathbb{F}^n \rightarrow \mathbb{F}^n$.

The scheme can be used both as an encryption and a digital signature scheme.

4.1.1 HFE as an Encryption Scheme

Encryption To encrypt a message $\mathbf{z} \in \mathbb{F}^n$, one simply evaluates the public key \mathcal{P} to get the ciphertext

$$\mathbf{w} = \mathcal{P}(\mathbf{z}) \in \mathbb{F}^n.$$

Decryption To decrypt a ciphertext $\mathbf{w} \in \mathbb{F}^n$, one has to perform the following three steps.

1. Invert the first affine map \mathcal{S} , i.e. compute

$$\mathbf{x} = \mathcal{S}^{-1}(\mathbf{w}) \in \mathbb{F}^n$$

and lift the result to the extension field \mathbb{E} , i.e. compute

$$X = \phi(\mathbf{x}).$$

2. Find all the solutions Y_1, \dots, Y_k of $\mathcal{F}(Y) = X$, i.e. compute the set

$$\mathcal{Y} = \{Y \in \mathbb{E} : \mathcal{F}(Y) = X\}.$$

For this step, one uses e.g. Berlekamp's algorithm or the Cantor–Zassenhaus algorithm (see Sect. 8.2).

3. For each $i \in \{1, \dots, k\}$, send Y_i down to the vector space \mathbb{F}^n , i.e. compute

$$\mathbf{y}_i = \phi^{-1}(Y_i)$$

and compute the plaintext candidate $\mathbf{z}_i \in \mathbb{F}^n$ by $\mathbf{z}_i = \mathcal{T}^{-1}(\mathbf{y}_i)$.

Since the HFE central map is not bijective, the equation $\mathcal{F}(Y) = X$ in step 2 of the decryption process may have several solutions (this case is also shown in the toy example below). However, for each ciphertext, there is at least one solution, which corresponds to the correct plaintext. To distinguish between correct and false plaintext candidates, one can use several techniques (e.g. hash functions or redundancy in the plaintext).

4.1.2 HFE as a Signature Scheme

Since the HFE central map is not bijective, not every hash value $\mathbf{w} \in \mathbb{F}^n$ necessarily has a signature. To overcome this problem, one can use for example a counter $r = 0, 1, \dots$ during the signature generation process. Instead of generating an HFE signature \mathbf{z} for the hash value $\mathbf{w} = \mathcal{H}(d)$, one generates an HFE signature for the hash value $\mathbf{w} = \mathcal{H}(d||r)$. If \mathbf{w} does not lead to a signature, one increases the counter r and tries again. The final signature σ sent to the verifier has the form $\sigma = (\mathbf{z}, r)$.

Signature Generation To generate a signature for a document d , one starts with $r = 0$, computes the hash value $\mathbf{w} = \mathcal{H}(d||r)$ and performs the following three steps.

1. Invert the first affine map \mathcal{S} , i.e. compute

$$\mathbf{x} = \mathcal{S}^{-1}(\mathbf{w}) \in \mathbb{F}^n$$

and lift the result to the extension field \mathbb{E} , i.e. compute

$$X = \phi(\mathbf{x}).$$

2. Find a solution $Y \in \mathbb{E}$ of the univariate polynomial equation $\mathcal{F}(Y) = X$ for example via Berlekamp's algorithm or the Cantor–Zassenhaus algorithm (see Sect. 8.2). If the equation does not have a solution, increment the counter r , compute the new hash value $\mathbf{w} = \mathcal{H}(d||r)$ and start again with step 1.
3. Move the result Y down to the vector space \mathbb{F}^n , i.e. compute

$$\mathbf{y} = \phi^{-1}(Y)$$

and compute the HFE signature $\mathbf{z} \in \mathbb{F}^n$ by $\mathbf{z} = \mathcal{T}^{-1}(\mathbf{y})$. Send (\mathbf{z}, r) to the verifier.

Signature Verification To check the authenticity of a signature (\mathbf{z}, r) , the verifier computes the hash value $\mathbf{w} = \mathcal{H}(d||r)$ and $\mathbf{w}' = \mathcal{P}(\mathbf{z})$. If $\mathbf{w}' = \mathbf{w}$ holds, the signature is accepted, otherwise rejected.

4.1.3 Key Sizes and Efficiency

The public key size of HFE is

$$\text{size}_{\text{pk HFE}} = n \frac{(n+1)(n+2)}{2}$$

\mathbb{F} -elements. The size of the private key is

$$\text{size}_{\text{sk HFE}} = \underbrace{(n+1)^2}_{\mathcal{S}} + \underbrace{(n+1)^2}_{\mathcal{T}} + \underbrace{kn}_{\mathcal{F}}$$

\mathbb{F} -elements. The number of coefficients of the HFE polynomial is k , which is bounded from above by

$$k \leq \frac{\lceil \log_q(D) \rceil \cdot (\lceil \log_q(D) \rceil + 1)}{2} + \log_q(D) + 1.$$

The most costly step during the decryption/signature generation process of HFE is the inversion of the univariate HFE polynomial, which is performed by Berlekamp's or the Cantor–Zassenhaus algorithm (see Sect. 8.2). The complexity of both algorithms is cubic in the parameter D .

4.1.4 Toy Example

For our toy example we use $\mathbb{F} = GF(4)$ as the underlying field and choose the HFE parameters $(n, D) = (3, 17)$. The irreducible polynomial used to generate the extension field $\mathbb{E} = \mathbb{F}_{4^3}$ is $f(X) = X^3 + \alpha$.

We choose the affine maps \mathcal{S} and $\mathcal{T} : \mathbb{F}^3 \rightarrow \mathbb{F}^3$ as

$$\mathcal{S}(x_1, \dots, x_3) = \begin{pmatrix} \alpha & \alpha^2 & \alpha^2 \\ 0 & \alpha^2 & 1 \\ \alpha & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} \alpha \\ \alpha \\ \alpha \end{pmatrix}$$

and

$$\mathcal{T}(x_1, x_2, x_3) = \begin{pmatrix} \alpha^2 & 1 & 1 \\ 0 & \alpha^2 & \alpha \\ 0 & \alpha & \alpha \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} \alpha^2 \\ 0 \\ 1 \end{pmatrix}.$$

The HFE central map $\mathcal{F} : \mathbb{E} \rightarrow \mathbb{E}$ is given as

$$\mathcal{F}(\hat{X}) = \beta_{17}\hat{X}^{4^2+4^0} + \beta_8\hat{X}^{4^1+4^1} + \beta_5\hat{X}^{4^1+4^0} + \beta_2\hat{X}^{4^0+4^0} + \gamma_{16}\hat{X}^{4^2} + \gamma_4\hat{X}^{4^1} + \gamma_1\hat{X}^{4^0} + \delta$$

with

$$\begin{aligned} \beta_{17} &= X^2 + \alpha X + 1, \\ \beta_8 &= \alpha^2 X^2 + \alpha^2 X, \\ \beta_5 &= \alpha^2 X^2 + \alpha, \\ \beta_2 &= \alpha^2 X^2 + \alpha X + \alpha, \\ \gamma_{16} &= \alpha X + \alpha, \\ \gamma_4 &= X^2 + 1, \\ \gamma_1 &= \alpha X^2 + X + \alpha^2, \\ \delta &= \alpha^2 X^2 + \alpha X + 1. \end{aligned}$$

Here, the coefficients of the HFE central map are represented by univariate polynomials of degree ≤ 2 in $\mathbb{F}[X]$.

In order to compute the public key of our HFE instance, we first lift the affine transformation $\mathcal{T}(x_1, \dots, x_3)$ to a univariate polynomial $\tilde{T}(X)$ over the extension field \mathbb{E} . We obtain

$$\tilde{T}(X) = (\alpha x_2 + \alpha x_3 + 1)X^2 + (\alpha^2 x_2 + \alpha x_3)X + \alpha^2 x_1 + x_2 + x_3 + \alpha^2.$$

Next, we compute $B(X) = \mathcal{F}(\tilde{T}(X))$, obtaining

$$\begin{aligned} B(X) &= (\alpha^2 x_1^2 + x_1 x_2 + \alpha^2 x_1 x_3 + \alpha x_1 + \alpha x_2 x_3 + x_2 + 1)X^2 \\ &\quad + (x_1^2 + \alpha x_1 x_2 + \alpha x_2 x_3 + \alpha^2 x_2 + \alpha^2 x_3^2 + \alpha x_3 + \alpha^2)X \\ &\quad + \alpha x_1^2 + \alpha x_1 x_2 + \alpha x_1 x_3 + x_1 + \alpha^2 x_2^2 + \alpha x_2 x_3 + \alpha x_2 + \alpha^2 x_3^2 + \alpha^2 x_3. \end{aligned}$$

Finally, we move $B(X)$ down to the vector space \mathbb{F}^3 and apply the affine transformation \mathcal{S} to obtain the public key $\mathcal{P} = (p^{(1)}, p^{(2)}, p^{(3)})$ as

$$\begin{aligned} p^{(1)} &= \alpha x_1^2 + x_1 x_2 + x_1 x_3 + \alpha^2 x_1 + x_2^2 + \alpha^2 x_2 x_3 + \alpha x_2 + \alpha^2 x_3^2 + \alpha^2, \\ p^{(2)} &= \alpha^2 x_1 x_3 + \alpha x_1 + \alpha^2 x_2 x_3 + \alpha^2 x_2 + \alpha x_3^2 + x_3 + 1, \\ p^{(3)} &= x_1^2 + x_2^2 + \alpha^2 x_2 x_3 + x_2 + \alpha x_3^2 + \alpha^2 x_3. \end{aligned}$$

We want to encrypt the message $\mathbf{z} = (\alpha^2, \alpha^2, \alpha) \in \mathbb{F}^3$. We find

$$\mathbf{w} = \mathcal{P}(\mathbf{z}) = (\alpha^2, 1, 0).$$

In order to decrypt the ciphertext $\mathbf{w} = (\alpha^2, 1, 0)$, we first have to invert the affine map \mathcal{S} . We find

$$\mathcal{S}^{-1} = \begin{pmatrix} \alpha & 0 & 1 \\ \alpha & \alpha^2 & \alpha \\ 1 & \alpha^2 & 1 \end{pmatrix}$$

and

$$\mathbf{x} = \mathcal{S}^{-1}(\mathbf{w}) = (0, \alpha^2, 1).$$

We lift \mathbf{x} to the extension field \mathbb{E} , obtaining

$$\hat{X} = X^2 + \alpha^2 X$$

and invert the central map \mathcal{F} by finding the solutions of $\mathcal{F}(\hat{Y}) - \hat{X} = 0$ using Berlekamp's algorithm. By doing so, we get the two solutions

$$\hat{Y}_1 = \alpha^2 X^2 + X \quad \text{and} \quad \hat{Y}_2 = X^2 + \alpha^2.$$

By moving \hat{Y}_1 and \hat{Y}_2 down to the vector space \mathbb{F}^3 and inverting the affine map \mathcal{T} , we obtain the two plaintext candidates

$$\mathbf{z}_1 = (\alpha^2, \alpha^2, \alpha) \quad \text{and} \quad \mathbf{z}_2 = (0, 0, 0).$$

Note that \mathbf{z}_1 is the encrypted plaintext. For example by using redundancy in the plaintexts, we can exclude \mathbf{z}_2 from the list of possible plaintext candidates.

4.2 Attacks on HFE

The most important attacks against the HFE cryptosystem are

- the direct attack and
- Rank Attacks of the Kipnis–Shamir type [8]

4.2.1 The Direct Attack on HFE

As already mentioned in Sect. 2.4, a direct attack considers the public equation $\mathcal{P}(\mathbf{z}) = \mathbf{w}$ as an instance of the MQ Problem. The resulting multivariate quadratic system is then solved using the XL-Algorithm or a Gröbner basis method (see Chap. 8). Experiments [6, 9] have shown that, in the case of HFE, the resulting multivariate quadratic systems can be solved significantly faster than random systems. The reason for this is the low degree of regularity of these systems. For HFE, the degree of regularity of the multivariate quadratic system is bounded from above by

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

where $r = \lfloor \log_q(D-1) \rfloor + 1$. A proof of this formula can be found in Sect. 8.6.

4.2.2 Rank Attacks of the Kipnis–Shamir Type

We define

$$r = \log_q \lfloor D-1 \rfloor + 1.$$

The basic idea of the Kipnis–Shamir attack against HFE is to consider the public key of a multivariate BigField Scheme such as HFE as a univariate polynomial map over the extension field \mathbb{E} . Before we come to the description of the attack itself, we start by introducing the notion of the (min)-Q-Rank of a multivariate quadratic map, which is a central notion for the cryptanalysis of multivariate BigField schemes.

4.2.2.1 The Notion of Q-Rank

Via the isomorphism ϕ , we can lift every multivariate quadratic map $\mathcal{G} : \mathbb{F}^n \rightarrow \mathbb{F}^n$ to a univariate polynomial map $\phi \circ \mathcal{G} \circ \phi^{-1}$ over the extension field \mathbb{E} of the form

$$\phi \circ \mathcal{G} \circ \phi^{-1} = \sum_{i,j=0}^{n-1} \alpha_{ij} X^{q^i + q^j}.$$

Via the identification $X_i := X^{q^i}$, $\phi \circ \mathcal{G} \circ \phi^{-1}$ becomes a quadratic form in the multivariate polynomial ring $\mathbb{E}[X_0, \dots, X_{n-1}]$.

With this notion, we can define the Q-Rank of the multivariate quadratic map \mathcal{G} as follows.

Definition 4.2 Let \mathbb{E} be a degree n extension field of \mathbb{F} . The **Q-rank** of a quadratic map $\mathcal{G}(\mathbf{x}) : \mathbb{F}^n \rightarrow \mathbb{F}^n$ is the rank of the quadratic form $\phi \circ \mathcal{G} \circ \phi^{-1}$ in $\mathbb{E}[X_0, \dots, X_{n-1}]$ via the identification $X_i = \phi(\mathbf{x})^{q^i}$.

The Q-rank of a multivariate quadratic map \mathcal{G} is therefore the minimum number of variables required to express the quadratic form $\phi \circ \mathcal{G} \circ \phi^{-1}$ in the multivariate polynomial ring $\mathbb{E}[X_0, \dots, X_{n-1}]$.

The Q-rank of a multivariate quadratic map \mathcal{G} is invariant under one-sided isomorphisms $\mathcal{G} \mapsto \mathcal{G} \circ \mathcal{T}$, but is not invariant under isomorphisms of polynomials in general, i.e. transformations of the form $\mathcal{G} \rightarrow \mathcal{S} \circ \mathcal{G} \circ \mathcal{T}$, where \mathcal{S} and \mathcal{T} are invertible affine transformations. Therefore, when studying the security of multivariate public key cryptosystems, another quantity which is invariant under isomorphisms of polynomials is more important.

Definition 4.3 Let \mathbb{E} be a degree n extension field of \mathbb{F} . The **min-Q-rank** of a quadratic map $\mathcal{G} : \mathbb{F}^n \rightarrow \mathbb{F}^m$ over \mathbb{E} is

$$\text{min-Q-rank}(\mathcal{G}) = \min_S \max_{\mathcal{T}} \{\text{Q-rank}(\mathcal{S} \circ \mathcal{G} \circ \mathcal{T})\},$$

where $\mathcal{S} : \mathbb{F}^m \rightarrow \mathbb{F}^r$ and $\mathcal{T} : \mathbb{F}^s \rightarrow \mathbb{F}^n$ are full rank linear transformations. As above, “Q-rank” computes the rank of its input as a quadratic form over $\mathbb{E}[X_0, \dots, X_{n-1}]$ via the identification $X_i = X^{q^i}$.¹

It is easy to see that the min-Q-rank of a multivariate quadratic map is invariant under isomorphisms of polynomials. Therefore, for a multivariate public key cryptosystem, the min-Q-rank of the central map \mathcal{F} (or $\phi^{-1} \circ \mathcal{F} \circ \phi$) and the min-Q-rank of the public key \mathcal{P} are equal. Therefore, the notion of min-Q-rank plays an important role in the cryptanalysis of multivariate public key cryptosystems of the BigField family.

4.2.2.2 The Case of HFE

Let α be a primitive element of the degree n extension field \mathbb{E} of \mathbb{F} . We define the isomorphism $\phi : \mathbb{F}^n \rightarrow \mathbb{E}$ as $X = \phi(x_1, \dots, x_n) = \sum_{i=1}^n x_i \alpha^{i-1}$. Furthermore, we define a map $\varphi : \mathbb{E} \rightarrow \mathbb{E}^n$ over the extension field \mathbb{E} as $\varphi(a) = (a, a^q, \dots, a^{q^{n-1}})$ and denote the image of φ by \mathbb{A} .

Define a map $\mathcal{M}_n : \mathbb{F}^n \rightarrow \mathbb{A}$ by $\mathcal{M}_n = \varphi \circ \phi$. We can explicitly represent this map by the matrix

$$\mathbf{M}_n = \begin{pmatrix} 1 & 1 & \dots & 1 \\ \alpha & \alpha^q & \dots & \alpha^{q^{n-1}} \\ \alpha^2 & \alpha^{2q} & \dots & \alpha^{2q^{n-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha^{n-1} & \alpha^{(n-1)q} & \dots & \alpha^{(n-1)q^{n-1}} \end{pmatrix} \in \mathbb{E}^{n \times n},$$

acting via right multiplication. We can thus pass between the two interesting representations of elements of \mathbb{E} of the form $(x_1, \dots, x_n) \in \mathbb{F}^n$ and $(X, X^q, \dots, X^{q^{n-1}}) \in \mathbb{A}$ simply by right multiplication by \mathbf{M}_n or \mathbf{M}_n^{-1} .

With the help of this matrix \mathbf{M}_n , we can write the public key of HFE in matrix form. Let \mathbf{F}^{*i} be the matrix representation of the quadratic form over \mathbb{A} corresponding to the map $x \mapsto X^{q^i}$.

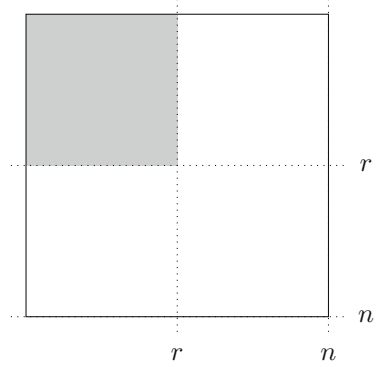
Let $(F^{(1)}, \dots, F^{(n)})$ denote the n -dimensional vector of $n \times n$ symmetric matrices associated to the private key. We find

$$\mathbf{M}_n(F^{(1)}, F^{(2)}, \dots, F^{(n)}) = (\mathbf{M}_n \mathbf{F}^{*0} \mathbf{M}_n^\top, \mathbf{M}_n \mathbf{F}^{*1} \mathbf{M}_n^\top, \dots, \mathbf{M}_n \mathbf{F}^{*(n-1)} \mathbf{M}_n^\top).$$

Here, $\mathbf{M}_n(F^{(1)}, F^{(2)}, \dots, F^{(n)})$ denotes the function of applying each coordinate of the vector $(F^{(1)}, F^{(2)}, \dots, F^{(n)})$, followed by the application of the linear map \mathbf{M}_n .

¹The definition holds also for non square transformations \mathcal{S} and \mathcal{T} as they are used in variants of MI and HFE. Therefore we speak here of full rank linear transformations.

Fig. 4.1 Structure of the matrix \mathbf{F}^{*0} for HFE



The next observation is that, due to the special form of the HFE central map, the matrix \mathbf{F}^{*0} has the form shown in Fig. 4.1.

As Fig. 4.1 shows, the rank of the matrix \mathbf{F}^{*0} is less than or equal to $r = \log_q \lfloor D - 1 \rfloor + 1$. Therefore, we can recover the matrix \mathbf{F}^{*0} by solving an instance of the MinRank Problem.

Definition 4.4 (MinRank Problem) Given k $n \times n$ matrices M_1, \dots, M_k , find a linear combination

$$M = \sum_{i=1}^k \lambda_i M_i$$

of minimal rank r .

There exist two techniques to solve the MinRank Problem for HFE: the Kipnis–Shamir Modeling and the Minors Modeling.

4.2.2.3 Kipnis–Shamir Modeling

Let M_1, \dots, M_k be k $n \times n$ matrices. We want to find a linear combination $M = \sum_{i=1}^k \lambda_i M_i$ of rank $\leq r$. In the Kipnis–Shamir Modeling [8], we do this by computing the right kernel of the (unknown) matrix M . Since M has rank r , there exist $n - r$ linear independent vectors $\mathbf{v}_1, \dots, \mathbf{v}_{n-r}$ such that $M \cdot \mathbf{v}_i = 0 \forall i = 1, \dots, n - r$. If we define V as the $n \times (n - r)$ matrix whose columns are the vectors $\mathbf{v}_1, \dots, \mathbf{v}_{n-r}$, we can write this as a matrix equation of the form

$$M \cdot V = 0_{n \times (n-r)}. \quad (4.2)$$

We can further assume that V is given in a systematic form, i.e.

$$V = \begin{pmatrix} I_{n-r} \\ V'_{r \times (n-r)} \end{pmatrix}.$$

Therefore, Eq. (4.2) yields $n(n-r)$ quadratic equations in the unknowns $\lambda_1, \dots, \lambda_k$ and the $r(n-r)$ entries of the matrix V' . The Kipnis–Shamir modeling solves the MinRank Problem by turning it into an overdetermined system of bilinear equations.

4.2.2.4 Minors Modeling

In [1], Courtois et al. proposed another technique to solve the MinRank Problem for HFE called Minors modeling. This technique has the benefit that it can be extended more easily to other HFE variants such as HFEv- (see Sect. 4.4). The basic observation is that, for a matrix of rank r , all $r+1$ minors (the determinants of $(r+1) \times (r+1)$ submatrices) must be 0.

We consider a general linear combination of the matrices M_1, \dots, M_k and its $r+1$ minors (which are given as polynomial equations of degree $r+1$ in the k unknowns $\lambda_1, \dots, \lambda_k$). We therefore have to solve a system of $\binom{n}{r+1}^2/2$ equations of degree $r+1$ in k variables.²

Courtois et al. now observed that it is sufficient to compute a Gröbner basis of the system over the base field \mathbb{F} , while computing the variety over the extension field \mathbb{E} . Due to this observation, they could speed up the attack drastically. The complexity of solving the HFE MinRank Problem ($k = n$) by this technique can be estimated by $O(n^{\omega})$, where $2 < \omega \leq 3$ is the linear algebra constant of solving a linear system.

Remark 4.5 Additionally to the Kipnis–Shamir and the Minors modeling described above, there exists a third technique to solve the MinRank Problem. Following this strategy, one chooses random vectors $\mathbf{v} \in \mathbb{F}^n$ and checks if \mathbf{v} is contained in the kernel of the linear combination M . Since M has rank r , this is fulfilled with probability q^{r-n} (where q is the cardinality of the underlying field \mathbb{F}). This technique is mainly used when applying the MinRank method to SingleField schemes such as Rainbow (see Sect. 5.4) and SimpleMatrix (see Sect. 7). We describe the technique in more detail in Sect. 5.5.

4.2.3 Summary of the Security of HFE

As we have seen above, the security of an HFE instance is mainly determined by the choice of the parameter D (or more precisely $r = \lfloor \log_q(D-1) \rfloor + 1$). In order to increase the security of HFE against direct and rank attacks, a higher value of r is desirable.

²Since we deal with symmetric matrices, each two minors produce the same equation.

However, the complexity of the decryption and signature generation process is cubic in D , which forbids a too large value of D . The best way to balance security and efficiency of the scheme is to choose the parameter q as small as possible. But even with $q = 2$ it is very difficult to create an HFE scheme which is both secure and efficient. Therefore, HFE is mainly used today as a building block for more advanced schemes. Some of these are described in the next two sections.

4.3 Encryption Schemes Based on HFE

After the basic HFE cryptosystem was shown to be insecure, several HFE variants for encryption have been proposed. In this section we introduce two of them: the IPHFE+ and the ZHFE encryption scheme.

4.3.1 The IPHFE+ Encryption Scheme

In order to transform the basic HFE cryptosystem into a more secure encryption scheme, IPHFE+ uses the same techniques as the PMI+ encryption scheme, namely internal perturbation and the plus modification. The scheme can be described as follows.

Let \mathbb{F}_q be a finite field of q elements and \mathbb{E} be a degree n extension field of \mathbb{F} . Let $\phi : \mathbb{F}^n \rightarrow \mathbb{E}$ be an isomorphism between the vector space \mathbb{F}^n and the extension field \mathbb{E} . Furthermore, we choose three integers D, r and s .

Key Generation Let

$$\mathcal{F} : \mathbb{E} \rightarrow \mathbb{E}, \mathcal{F}(X) = \sum_{i,j=0}^{q^i+q^j \leq D} \alpha_{ij} X^{q^i+q^j} + \sum_{i=0}^{q^i \leq D} \beta_i X^{q^i} + \gamma$$

be a HFE central map. As in the case of PMI (see Sect. 3.3) we choose randomly a linear map $\mathcal{Z} : \mathbb{F}^n \rightarrow \mathbb{F}^r$ and a quadratic map $\tilde{\mathcal{F}} : \mathbb{F}^r \rightarrow \mathbb{F}^n$. We compute the set

$$P = \{(\lambda, \mu) | \tilde{\mathcal{F}}(\lambda) = \mu, \lambda \in \mathbb{F}^r\}.$$

The central map $\tilde{\mathcal{F}} : \mathbb{F}^n \rightarrow \mathbb{F}^n$ of IPHFE+ is given by

$$\tilde{\mathcal{F}} = \phi^{-1} \circ \mathcal{F} \circ \phi + \tilde{\mathcal{F}} \circ \mathcal{Z}.$$

Furthermore, we choose randomly a quadratic map $\mathcal{G} : \mathbb{F}^n \rightarrow \mathbb{F}^s$.

To hide the structure of the central map in the public key, we compose it with two invertible linear maps $\mathcal{S} : \mathbb{F}^{n+s} \rightarrow \mathbb{F}^{n+s}$ and $\mathcal{T} : \mathbb{F}^n \rightarrow \mathbb{F}^n$, so that we get

Public Key $\mathcal{P} : \mathbb{F}^n \rightarrow \mathbb{F}^{n+s} : \mathcal{P}(x_1, \dots, x_n) = \mathcal{S} \circ (\tilde{\mathcal{F}} || \mathcal{G}) \circ \mathcal{T}(x_1, \dots, x_n)$.

Private Key linear maps \mathcal{S} and \mathcal{T} , central map \mathcal{F} , linear map \mathcal{Z} , set P .

Encryption To encrypt a plaintext $\mathbf{z} \in \mathbb{F}^n$, the sender of a message computes the ciphertext $\mathbf{w} \in \mathbb{F}^{n+s}$ by $\mathbf{w} = \mathcal{P}(\mathbf{z})$.

Decryption In order to decrypt the ciphertext $\mathbf{w} \in \mathbb{F}^{n+s}$ the owner of the private key performs the following three steps.

1. Compute $\mathbf{x}' = \mathcal{S}^{-1}(\mathbf{w})$ and set $\mathbf{x} = (x'_1, \dots, x'_n)$.
2. Perform for each pair $(\lambda, \mu) \in P$ the following steps
 - (a) Compute $X_\lambda = (\mathbf{x} + \mu) \in \mathbb{E}$.
 - (b) Use Berlekamp's algorithm to find $Y_\lambda \in \mathbb{E}$ such that $\mathcal{F}(Y_\lambda) = X_\mu$.
 - (c) Compute $\mathbf{y}_\lambda = \phi^{-1}(Y_\lambda) \in \mathbb{F}^n$.
3. If $\mathcal{Z}(\mathbf{y})_\lambda = \lambda$ holds, $\mathbf{z}_\lambda = \mathcal{T}^{-1}(\mathbf{y}_\lambda) \in \mathbb{F}^n$ is a valid plaintext candidate. Otherwise, we discard the pair (λ, μ) and continue.

By this strategy, we might get several plaintext candidates $\mathbf{z}_\mu \in \mathbb{F}^n$. In this case, we can use the plus polynomials to distinguish between correct and false plaintext candidates.

4.3.2 Security and Efficiency

The security of IPHFE+ is based on the following observation: the internal perturbation increases the number of terms of the form $X^{q^i+q^j}$ in the central map \mathcal{F} and therefore the rank of the matrix $\mathbf{F}^{\star 0}$. In fact, it was shown by computer experiments [5] that the rank of the matrix $\mathbf{F}^{\star 0}$ is, for IPHFE+, close to $r + 1 + \log_q D$.

Therefore, the internal perturbation prevents MinRank attacks of the Kipnis–Shamir type and also increases the complexity of direct attacks. Similar to the case of PMI+ (see Sect. 3.3), the plus modification prevents differential attacks to remove the internal perturbation from the scheme.

Currently, no efficient attack against the IPHFE+ cryptosystem is known and the scheme is believed to be secure. However, since the decryption process contains a guessing step, the scheme is not very efficient and hardly used in practice.

4.3.3 The ZHFE Encryption Scheme

The main weakness of the HFE cryptosystem is the low min-Q-rank of the central map, which is caused by the upper bound on the degree of the univariate HFE polynomial. To address this fact, Porras et al. proposed in [12] the ZHFE encryption

scheme. The scheme uses two HFE polynomials \mathcal{F}_1 and \mathcal{F}_2 of high degree, which are connected by a low degree secret polynomial Ψ .

Therefore, the central map \mathcal{F} of the scheme has a high min-Q-rank, while we can use the low min-Q-rank map Ψ during the decryption process. However, it was discovered in [2] that the ZHFE scheme can still be attacked by some kind of rank attack (see next section). The ZHFE scheme can be described as follows.

Key Generation We consider two HFE polynomials $\mathcal{F}^{(1)}$ and $\mathcal{F}^{(2)}$ of high degree, i.e.

$$\begin{aligned}\mathcal{F}^{(1)}(X) &= \sum_{i \geq j \geq 0}^{n-1} \alpha_{ij}^{(1)} X^{q^i + q^j} + \sum_{i \geq 0}^{n-1} \beta_i^{(1)} X^{q^i} + \gamma^{(1)} \text{ and} \\ \mathcal{F}^{(2)}(X) &= \sum_{i \geq j \geq 0}^{n-1} \alpha_{ij}^{(2)} X^{q^i + q^j} + \sum_{i \geq 0}^{n-1} \beta_i^{(2)} X^{q^i} + \gamma^{(2)}.\end{aligned}\quad (4.3)$$

The coefficients $\alpha_{ij}^{(k)}, \beta_i^{(k)}$ and $\gamma^{(k)}$ ($k = 1, 2$) are so far undetermined elements of the extension field \mathbb{E} . Furthermore we define a polynomial Ψ by

$$\begin{aligned}\Psi(X) &= X \left(\sum_{i=0}^{n-1} \left(a_i (\mathcal{F}^{(1)}(X))^{q^i} + b_i (\mathcal{F}^{(2)}(X))^{q^i} \right) \right) \\ &\quad + X^q \left(\sum_{i=0}^{n-1} \left(c_i (\mathcal{F}^{(1)}(X))^{q^i} + d_i (\mathcal{F}^{(2)}(X))^{q^i} \right) \right)\end{aligned}\quad (4.4)$$

with randomly chosen coefficients a_i, b_i, c_i and $d_i \in \mathbb{E}$.

The coefficients $\alpha_{ij}^{(k)}, \beta_i^{(k)}$ and $\gamma^{(k)}$ of $\mathcal{F}^{(1)}$ and $\mathcal{F}^{(2)}$ are determined in such a way that all terms of degree $> D_0$ in Ψ vanish (for a small integer D_0). This implies the solution of a large linear system. In [7] it is shown how this can be done in an efficient way.

The *central map* $\mathcal{F} : \mathbb{E} \rightarrow \mathbb{E}^2$ of the scheme is defined as the concatenation of the polynomials $\mathcal{F}^{(1)}$ and $\mathcal{F}^{(2)}$, i.e. $\mathcal{F} = (\mathcal{F}^{(1)} || \mathcal{F}^{(2)})$.

To hide the structure of \mathcal{F} in the public key, we combine it with two invertible linear (or affine) maps $\mathcal{S} : \mathbb{F}^{2n} \rightarrow \mathbb{F}^{2n}$ and $\mathcal{T} : \mathbb{F}^n \rightarrow \mathbb{F}^n$.

Public Key The public key of ZHFE is the multivariate quadratic map

$$\mathcal{P} = \mathcal{S} \circ (\phi^{-1} \times \phi^{-1}) \circ (\mathcal{F}^{(1)}, \mathcal{F}^{(2)}) \circ \phi \circ \mathcal{T} : \mathbb{F}^n \rightarrow \mathbb{F}^{2n}.$$

Private Key The private key of ZHFE consists of the two affine maps $\mathcal{S} : \mathbb{F}^{2n} \rightarrow \mathbb{F}^{2n}$ and $\mathcal{T} : \mathbb{F}^n \rightarrow \mathbb{F}^n$, the two maps $\mathcal{F}^{(1)}, \mathcal{F}^{(2)} : \mathbb{E} \rightarrow \mathbb{E}$ and the coefficients a_i, b_i, c_i and d_i of the map Ψ .

Encryption To encrypt a plaintext $\mathbf{z} \in \mathbb{F}^n$, one simply computes $\mathbf{w} = \mathcal{P}(\mathbf{z}) \in \mathbb{F}^{2n}$.

Decryption To decrypt a ciphertext $\mathbf{w} \in \mathbb{F}^{2n}$, the owner of the private key performs the following four steps.

1. Compute $\mathbf{x} = \mathcal{S}^{-1}(\mathbf{w}) \in \mathbb{F}^n$ and set $X_1 = \phi(x_1, \dots, x_n)$,
 $X_2 = \phi(x_{n+1}, \dots, x_{2n}) \in \mathbb{E}$.
2. Define the three maps $\tilde{\mathcal{F}}^{(1)} = \mathcal{F}_1 - X_1$, $\tilde{\mathcal{F}}^{(2)} = \mathcal{F}_2 - X_2$ and

$$\begin{aligned} \tilde{\Psi} = & X \left(\sum_{i=0}^{n-1} \left(a_i (\tilde{\mathcal{F}}^{(1)}(X))^{q^i} + b_i (\tilde{\mathcal{F}}^{(2)}(X))^{q^i} \right) \right) \\ & + X^q \left(\sum_{i=0}^{n-1} \left(c_i (\tilde{\mathcal{F}}^{(1)}(X))^{q^i} + d_i (\tilde{\mathcal{F}}^{(2)}(X))^{q^i} \right) \right). \end{aligned}$$

Obviously, the degree of $\tilde{\Psi}$ is bounded by D_0 .

3. Use Berlekamp's algorithm to find a solution $Y \in \mathbb{E}$ of the equation $\tilde{\Psi}(Y) = 0$ and set $\mathbf{y} = \phi^{-1}(Y) \in \mathbb{F}^n$.
4. Compute the plaintext $\mathbf{z} \in \mathbb{F}^n$ by $\mathbf{z} = \mathcal{T}^{-1}(\mathbf{y})$.

4.3.4 Key Sizes and Efficiency

The public key size of ZHFE is given as

$$\text{size}_{\text{pk ZHFE}} = 2n \frac{(n+1)(n+2)}{2}$$

\mathbb{F} -elements, the private key size is

$$\text{size}_{\text{sk ZHFE}} = \underbrace{2n(2n+1)}_{\mathcal{S}} + \underbrace{n(n+1)}_{\mathcal{T}} + 2kn$$

\mathbb{F} -elements. Here, k is the number of coefficients in the map $\mathcal{F}^{(1)}$, which is bounded above by n^2 .

The most costly step in the decryption process is the inversion of the map $\tilde{\Psi}$, which implies the solution of a univariate polynomial of degree D_0 .

4.3.5 Cryptanalysis of ZHFE

ZHFE was constructed with the idea in mind that an injective multivariate map from \mathbb{F}^n to \mathbb{F}^{2n} may have more freedom in its structure than a bijective map from \mathbb{F}^n to \mathbb{F}^n , and therefore can be more secure. However, the ZHFE construction could not really achieve this goal.

The ZHFE encryption scheme was shown to be insecure by a practical key recovery attack of Cabarcas et al. [2] based on the discoveries of a hidden low rank property of ZHFE. Despite the two central maps of ZHFE having high degree, the existence of the low rank combination Ψ makes ZHFE vulnerable to the Kipnis–Shamir(KS) rank attack.

The attack is based on the following observation.

Theorem 4.6 *Let $\mathcal{P} = \mathcal{S} \circ (\phi^{-1} \times \phi^{-1}) \circ (\mathcal{F}^{(1)}, \mathcal{F}^{(2)}) \circ \phi \circ \mathcal{T}$ be a ZHFE public key. Then, with high probability, there exists a ZHFE private key $\Pi = (\tilde{\mathcal{S}}, \tilde{\mathcal{F}}^{(1)}, \tilde{\mathcal{F}}^{(2)}, \tilde{\mathcal{T}})$ with $\tilde{\mathcal{S}} \circ (\phi^{-1} \times \phi^{-1}) \circ (\tilde{\mathcal{F}}^{(1)}, \tilde{\mathcal{F}}^{(2)}) \circ \phi \circ \tilde{\mathcal{T}} = \mathcal{P}$ for which the matrices $\tilde{\mathcal{F}}^{(1)}$ and $\tilde{\mathcal{F}}^{(2)}$ associated to the maps $\tilde{\mathcal{F}}^{(1)}$ and $\tilde{\mathcal{F}}^{(2)}$ are of low rank $\leq r + 1$ ($r = \log_q \lfloor D - 1 \rfloor + 1$).*

Proof See [2, Sect. 3.1]. □

In [2] it is shown how to find this equivalent private key by solving a MinRank Problem with target rank $r + 1$ in the $2n$ matrices associated to the public polynomials.

The complexity of the attack is $O(n^{(r+2)\omega})$, where $2 < \omega \leq 3$ is the exponent in the complexity of solving a linear system. For constant r , the running time of the attack is therefore polynomial in n .

4.4 Signature Schemes Based on HFE

The most popular signature scheme on the basis of HFE is HFEv-, which is obtained from the basic HFE scheme by using the minus and the Vinegar modifications.

Definition 4.7 The **Vinegar modification** perturbs the central map \mathcal{F} of a multivariate public key scheme by making the coefficients of the linear and constant terms of \mathcal{F} dependent on a set of external (vinegar) variables.

In contrast to the internal perturbation (see Sect. 3.3) the Vinegar modification introduces additional variables into the multivariate quadratic system. The resulting system thus has more variables than equations, which means that the Vinegar modification can only be used for signature schemes. In Sect. 4.4.4 we will see how the Vinegar modification affects known attacks against HFE.

Additional to providing a modification method for multivariate BigField schemes, the idea of the Vinegar modification can also be used to obtain a completely new signature scheme. This so called Oil and Vinegar signature scheme is discussed in Chap. 5.

Table 4.1 Modifications of BigField schemes

Encryption schemes	Plus +	Add randomly chosen quadratic equations to the public key	+ Prevents differential attacks (PMI+, IPHFE+)
	Internal perturbation IP	Add perturbation to the central map using internal variables	+ Prevents Linearization attacks (PMI) + Increases rank of the central map (IPHFE) – slows down decryption
Signature schemes	Minus –	Remove equations from the public key	+ prevents linearization equations attacks (MI-) + increases the rank of the central map (HFE-)
	Projection p	Project public key to a subspace	+ prevents differential attacks (MI-) – slows down signature generation
	Vinegar v	Add perturbation to the central map using external variables	+ increases the rank of the central map (HFEv-)

Table 4.1 gives an overview of the various modifications on multivariate BigField schemes.

4.4.1 The HFEv- Signature Scheme

Similar to the HFE cryptosystem, the HFEv- scheme uses a degree n extension field \mathbb{E} of \mathbb{F} and an isomorphism $\phi : \mathbb{F}^n \rightarrow \mathbb{E}$. As in HFE, we have an upper bound D on the degree of the univariate HFE polynomial in use. However, HFEv- uses two additional parameters:

- a : the number of minus equations and
- v : the number of vinegar variables.

Key Generation The central map $\mathcal{F} : \mathbb{E} \times \mathbb{F}^v \rightarrow \mathbb{E}$ of the HFEv- scheme has the form

$$\mathcal{F}(X, x_1, \dots, x_v) = \sum_{i,j \geq 0}^{q^i + q^j \leq D} \alpha_{i,j} X^{q^i + q^j} + \sum_{i \geq 0}^{q^i \leq D} \beta_i(x_1, \dots, x_v) X^{q^i} + \gamma(x_1, \dots, x_v).$$

Here, the coefficients $\alpha_{i,j}$ are randomly chosen elements from the field \mathbb{E} , while $\beta_i : \mathbb{F}^v \rightarrow \mathbb{E}$ and $\gamma : \mathbb{F}^v \rightarrow \mathbb{E}$ are linear and quadratic maps respectively. Due to the structure of the map \mathcal{F} the map

$$\bar{\mathcal{F}} : \phi^{-1} \circ \mathcal{F} \circ (\phi \times id_v)$$

is a quadratic map from \mathbb{F}^{n+v} to \mathbb{F}^n . Here, id_v is the identity map on \mathbb{F}^v .

To hide the structure of the central map \mathcal{F} in the public key, one composes $\bar{\mathcal{F}}$ with two linear or affine maps $\mathcal{S} : \mathbb{F}^n \rightarrow \mathbb{F}^{n-a}$ (minus modification) and $\mathcal{T} : \mathbb{F}^{n+v} \rightarrow \mathbb{F}^{n+v}$.

Public Key The public key of HFEv- is the multivariate quadratic map

$$\mathcal{P} = \mathcal{S} \circ \phi^{-1} \circ \mathcal{F} \circ (\phi \times id_v) \circ \mathcal{T} : \mathbb{F}^{n+v} \rightarrow \mathbb{F}^{n-a}.$$

Private Key The private key of HFEv- consists of the three maps \mathcal{S} , \mathcal{F} and \mathcal{T} (and possibly the isomorphism ϕ).

Since the public key of HFEv- contains more variables than equations, the scheme can only be used as a signature scheme.

Signature Generation To generate an HFEv- signature for a document $d \in \{0, 1\}^*$, one uses a hash function $\mathcal{H} : \{0, 1\}^* \rightarrow \mathbb{F}^{n-a}$ to compute the hash value $\mathbf{w} = \mathcal{H}(d) \in \mathbb{F}^{n-a}$ and performs the following four steps.

1. Find a pre-image $\mathbf{x} \in \mathbb{F}^n$ of the hash value \mathbf{w} under the affine map \mathcal{S} and lift it to the extension field \mathbb{E} . Denote the result by X .
2. Choose random values for the vinegar variables x_1, \dots, x_v and substitute them into the central map \mathcal{F} to obtain the parametrized map $\mathcal{F}_V : \mathbb{E} \rightarrow \mathbb{E}$.
3. Find a root of the univariate polynomial equation $\mathcal{F}_V(\hat{Y}) = X$ by e.g. Berlekamp's algorithm (see Sect. 8.2). Denote the root by $Y \in \mathbb{E}$. If there is no solution, go back to step 2.
4. Compute $\mathbf{y}' = \phi^{-1}(Y) \in \mathbb{F}^n$, append the vinegar variables x_1, \dots, x_v to get $\mathbf{y} = (\mathbf{y}' || x_1 || \dots || x_v) \in \mathbb{F}^{n+v}$ and compute the signature $\mathbf{z} \in \mathbb{F}^{n+v}$ as

$$\mathbf{z} = \mathcal{T}^{-1}(\mathbf{y}).$$

Signature Verification To check the authenticity of a signature $\mathbf{z} \in \mathbb{F}^{n+v}$, the verifier computes the hash value $\mathbf{w} = \mathcal{H}(d) \in \mathbb{F}^{n-a}$ and evaluates the public key at \mathbf{z} to obtain

$$\mathbf{w}' = \mathcal{P}(\mathbf{z}) \in \mathbb{F}^{n-a}.$$

If $\mathbf{w}' = \mathbf{w}$ holds, the signature \mathbf{z} is accepted, otherwise rejected.

4.4.2 Key Sizes and Efficiency

The public key size of HFEv- is

$$\text{size}_{\text{pk HFEv-}} = (n - a) \frac{(n + v + 1)(n + v + 2)}{2}$$

\mathbb{F} -elements. The size of the private key is

$$\text{size}_{\text{sk HFEv-}} = \underbrace{(n - a)n}_{\mathcal{S}} + \underbrace{n(n + v)}_{\mathcal{T}} + \underbrace{kn}_{\mathcal{F}}$$

\mathbb{F} -elements. Here, k is the number of coefficients in the HFEv- polynomial with the upper bound

$$k \leq \frac{\lceil \log_q(D) \rceil (\lceil \log_q(D) \rceil + 1)}{2} + \lceil \log_q(D) \rceil (v + 1) + \frac{(v + 1)(v + 2)}{2}.$$

Each of these coefficients is stored using n \mathbb{F} -elements. As in the case of the plain HFE scheme, the most costly step in the signature generation of the HFEv- scheme is the inversion of the univariate HFEv- polynomial, whose complexity is cubic in the parameter D .

4.4.3 Toy Example

For our toy example of the HFEv- signature scheme, we use the field $\text{GF}(4)$ with 4 elements and the HFEv- parameters $(n, D, a, v) = (4, 17, 1, 1)$. The irreducible polynomial used to generate the extension field $\mathbb{E} = \mathbb{F}_{4^4}$ is $f(X) = X^4 + X^2 + \alpha X + 1$.

We choose the affine maps $\mathcal{S} : \mathbb{F}^4 \rightarrow \mathbb{F}^3$ and $\mathcal{T} : \mathbb{F}^5 \rightarrow \mathbb{F}^5$ as

$$\mathcal{S}(x_1, \dots, x_4) = \begin{pmatrix} \alpha^2 & 1 & 1 & \alpha \\ \alpha & 1 & 0 & 0 \\ \alpha^2 & 1 & \alpha^2 & \alpha \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ \alpha \end{pmatrix}$$

and

$$\mathcal{T}(x_1, \dots, x_5) = \begin{pmatrix} 1 & \alpha & \alpha & 0 & \alpha^2 \\ 1 & 1 & 1 & 1 & \alpha \\ \alpha & \alpha & 1 & 0 & 0 \\ 0 & \alpha & \alpha^2 & 1 & \alpha \\ 0 & \alpha & \alpha^2 & \alpha & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} + \begin{pmatrix} \alpha \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

The central HFEv- map $\mathcal{F} : \mathbb{E} \times \mathbb{F} \rightarrow \mathbb{E}$ is given as

$$\begin{aligned} \mathcal{F}(\hat{X}, x_5) &= \beta_{17}\hat{X}^{4^2+4^0} + \beta_8\hat{X}^{4^1+4^1} + \beta_5\hat{X}^{4^1+4^0} + \beta_2\hat{X}^{4^0+4^0} \\ &\quad + \gamma_{16}(x_5)\hat{X}^{4^2} + \gamma_4(x_5)\hat{X}^{4^1} + \gamma_1(x_5)\hat{X}^{4^0} + \delta(x_5) \end{aligned}$$

with

$$\begin{aligned} \beta_{17} &= X^3 + \alpha X^2 + \alpha^2 X + \alpha^2, \\ \beta_8 &= \alpha X^3 + \alpha^2 X^2 + \alpha^2 X + \alpha^2, \\ \beta_5 &= X^3 + \alpha^2 X^2 + \alpha X + \alpha, \\ \beta_2 &= \alpha X^2 + X, \\ \gamma_{16}(x_5) &= \alpha^2 X^3 + (\alpha x_5 + \alpha^2) X^2 + (\alpha x_5 + \alpha) X + \alpha x_5, \\ \gamma_4(x_5) &= (\alpha^2 x_5 + \alpha^2) X^3 + (\alpha x_5 + \alpha^2) X^2 + (\alpha^2 x_5 + 1) X + x_5, \\ \gamma_1(x_5) &= (x_5 + \alpha^2) X^3 + \alpha x_5 X + \alpha x_5 + 1, \\ \delta(x_5) &= \alpha^2 X^3 + (\alpha^2 x_5^2 + 1) X^2 + (\alpha^2 x_5 + 1) X + \alpha x_5 + \alpha^2. \end{aligned}$$

Here, the elements of the extension field \mathbb{E} are represented as univariate polynomials of degree ≤ 3 in $\mathbb{F}[X]$.

In order to compute the public key \mathcal{P} , we first lift the affine map $\mathcal{T}(x_1, \dots, x_5)$ to a univariate polynomial $\tilde{\mathcal{T}}(X)$ over the extension field \mathbb{E} , obtaining

$$\begin{aligned} \tilde{\mathcal{T}}(X) &= (\alpha x_2 + \alpha^2 x_3 + \alpha x_4 + x_5 + 1) X^4 \\ &\quad + (\alpha x_2 + \alpha^2 x_3 + x_4 + \alpha x_5) X^3 + (\alpha x_1 + \alpha x_2 + x_3) X^2 \\ &\quad + (x_1 + x_2 + x_3 + x_4 + \alpha x_5 + 1) X + x_1 + \alpha x_2 + \alpha x_3 + \alpha^2 x_5 + \alpha. \end{aligned}$$

We substitute $\tilde{\mathcal{T}}$ into the HFE central map to obtain $B(X) = \mathcal{F}(\tilde{\mathcal{T}}(X))$, getting

$$\begin{aligned} B(X) &= (\alpha^2 x_1 x_3 + \alpha x_1 x_4 + x_1 x_5 + \alpha^2 x_1 + \alpha^2 x_2^2 + \alpha x_2 x_3 + \alpha x_3^2 + x_3 x_4 + \alpha x_3 x_5 \\ &\quad + \alpha^2 x_4^2 + \alpha x_4 x_5 + \alpha^2 x_5 + \alpha) X^3 \\ &\quad + (x_1^2 + x_1 x_2 + \alpha x_1 x_3 + \alpha x_1 x_4 + x_1 x_5 + \alpha x_1 + \alpha^2 x_2^2 + \alpha^2 x_2 x_3 + x_2 x_4 \\ &\quad + \alpha^2 x_2 x_5 + \alpha^2 x_2 + \alpha^2 x_3^2 + \alpha^2 x_3 x_4 + \alpha x_3 x_5 + \alpha x_3 + x_4^2 + x_4 + x_5^2 + \alpha^2) X^2 \end{aligned}$$

$$\begin{aligned}
& + (\alpha x_1^2 + \alpha x_1 x_2 + \alpha^2 x_1 x_4 + \alpha x_1 x_5 + \alpha^2 x_1 + \alpha x_2^2 + x_2 x_4 + \alpha x_2 x_5 + \alpha^2 x_3^2 \\
& + \alpha x_3 x_4 + x_3 + \alpha x_4^2 + x_4 x_5 + \alpha x_4 + x_5^2)X \\
& + x_1 x_2 + \alpha^2 x_1 x_3 + \alpha x_1 x_4 + x_1 x_5 + \alpha x_2^2 + \alpha^2 x_2 x_3 + \alpha^2 x_2 x_4 + \alpha x_2 x_5 \\
& + \alpha x_2 + \alpha^2 x_3 x_4 + \alpha^2 x_3 x_5 + \alpha^2 x_3 + x_4^2 + x_4 x_5 + \alpha^2 x_5^2 + \alpha^2 x_5 + \alpha.
\end{aligned}$$

Finally, we move $B(X)$ down to the vector space \mathbb{F}^4 and apply the affine transformation \mathcal{S} to obtain the public key $\mathcal{P} = (p^{(1)}, p^{(2)}, p^{(3)})$ as

$$\begin{aligned}
p^{(1)} &= \alpha^2 x_1^2 + x_1 x_3 + \alpha^2 x_1 x_4 + \alpha x_1 x_5 + x_2^2 + \alpha x_2 x_3 + \alpha x_2 x_4 + \alpha x_2 + \alpha^2 x_3^2 \\
&+ x_3 x_4 + \alpha^2 x_3 x_5 + x_3 + x_4^2 + x_4 x_5 + \alpha^2 x_4 + \alpha x_5^2 + \alpha^2 x_5, \\
p^{(2)} &= \alpha x_1^2 + x_1 x_3 + \alpha^2 x_1 + x_2^2 + x_2 x_3 + x_2 x_5 + \alpha^2 x_2 + \alpha^2 x_3^2 + \alpha^2 x_3 x_4 \\
&+ x_3 x_5 + \alpha^2 x_4 x_5 + \alpha x_4 + x_5 + \alpha, \\
p^{(3)} &= x_1^2 + \alpha x_1 x_2 + \alpha x_1 x_3 + \alpha^2 x_1 + \alpha^2 x_2 x_3 + x_2 x_5 + \alpha^2 x_2 + \alpha x_3^2 + \alpha x_3 \\
&+ \alpha^2 x_4^2 + x_4 x_5 + x_4 + \alpha^2 x_5 + \alpha.
\end{aligned}$$

We want to generate a signature for the hash value $\mathbf{w} = (0, 0, \alpha^2) \in \mathbb{F}^3$. A pre-image of \mathbf{w} under the affine map \mathcal{S} is given by

$$\mathbf{x} = (\alpha^2, 0, 0, \alpha) \in \mathbb{F}^4.$$

Next, we lift \mathbf{x} to the extension field \mathbb{E} , obtaining

$$\hat{X} = \alpha X^3 + \alpha^2.$$

In order to invert the central map, we choose the value of the vinegar variable x_5 randomly, e.g. $x_5 = \alpha$.

We substitute $x_5 = \alpha$ into the coefficients γ_i and δ of the central map, obtaining

$$\begin{aligned}
\gamma_{16} &= \alpha^2 X^3 + X + \alpha^2, \\
\gamma_4 &= \alpha X^3 + \alpha, \\
\gamma_1 &= X^3 + \alpha^2 X + \alpha, \\
\delta &= \alpha^2 X^3 + \alpha^2 X^2,
\end{aligned}$$

and try to solve the equation $\mathcal{F}_\alpha(Y) = \hat{X}$ using Berlekamp's algorithm.

Since we do not find a solution, we have to choose another value for x_5 and try again.

We choose $x_5 = 0$ and evaluate the coefficients γ_i and δ to

$$\begin{aligned}
\gamma_{16} &= \alpha^2 X^3 + \alpha^2 X^2 + \alpha X, \\
\gamma_4 &= \alpha^2 X^3 + \alpha^2 X^2 + X, \\
\gamma_1 &= \alpha^2 X^3 + 1, \\
\delta &= \alpha^2 X^3 + X^2 + X + \alpha^2.
\end{aligned}$$

Solving the equation $\mathcal{F}_0(Y) = \hat{X}$ yields

$$Y = \alpha X^3 + X^2 + \alpha X.$$

Moving Y down to the vector space \mathbb{F}^4 , appending the vinegar variable $x_5 = 0$ and inverting the affine map \mathcal{T} yields the signature

$$\mathbf{z} = (\alpha, \alpha^2, \alpha^2, 1, 0) \in \mathbb{F}^5.$$

To check the authenticity of the signature $\mathbf{z} = (\alpha, \alpha^2, \alpha^2, 1, 0)$, we just evaluate the public key \mathcal{P} at \mathbf{z} . We get

$$\mathbf{w} = \mathcal{P}(\mathbf{z}) = (0, 0, \alpha^2).$$

Since the result is equal to the hash value of the message, we accept the signature.

4.4.4 Security of HFEv-

Similar to the case of the standard HFE cryptosystem, the most important attacks against the HFEv- signature scheme are

- direct attacks and
- rank attacks of the Kipnis–Shamir type [8].

4.4.4.1 Direct Attacks

Similar to the case of HFE, direct attacks against HFEv- are more efficient than against random system. The reason for this is again the lower degree of regularity, which, in the case of HFEv-, is upper bounded by

$$d_{\text{reg}}(\text{HFEv-}) = \begin{cases} \frac{(q-1)(r+a+v-1)}{2} + 2 & \text{for } q \text{ even and } r + a \text{ odd} \\ \frac{(q-1)(r+a+v)}{2} + 2 & \text{otherwise,} \end{cases} \quad (4.5)$$

where $r = \lfloor \log_q(D-1) \rfloor + 1$. A derivation of this formula can be found in Sect. 8.6.

4.4.4.2 The Kipnis–Shamir Attack on HFEv-

In this section we use the same notation as in Sect. 4.2.2. The Kipnis–Shamir attack against HFEv- works mainly the same as the attack against HFE. We only have two important differences:

- due to the use of the vinegar variables, the matrices \mathbf{F}^{*i} ($i = 0, \dots, n-1$) are no longer $n \times n$, but $(n+v) \times (n+v)$ matrices. In the matrix \mathbf{F}^{*i} , the top left $n \times n$ submatrix corresponds to the matrix \mathbf{F}^{*i} of the HFE case, the bottom right $v \times v$ contains the coefficients of the quadratic map γ and the remaining parts correspond to the linear maps β_i .
- due to the use of the minus modification, the quadratic terms are no longer restricted to the top left $r \times r$ corner. Instead of this, we have a copies of this $r \times r$ matrix which are moved to the bottom right direction.

Therefore, the matrix \mathbf{F}^{*0} corresponding to the central map \mathcal{F} of HFEv- has the form shown in Fig. 4.2. The rank of the matrix \mathbf{F}^{*0} is given by

$$\text{Rank}(\mathbf{F}^{*0}) = r + a + v.$$

The complexity of the Kipnis–Shamir attack (using Minors modelling) is

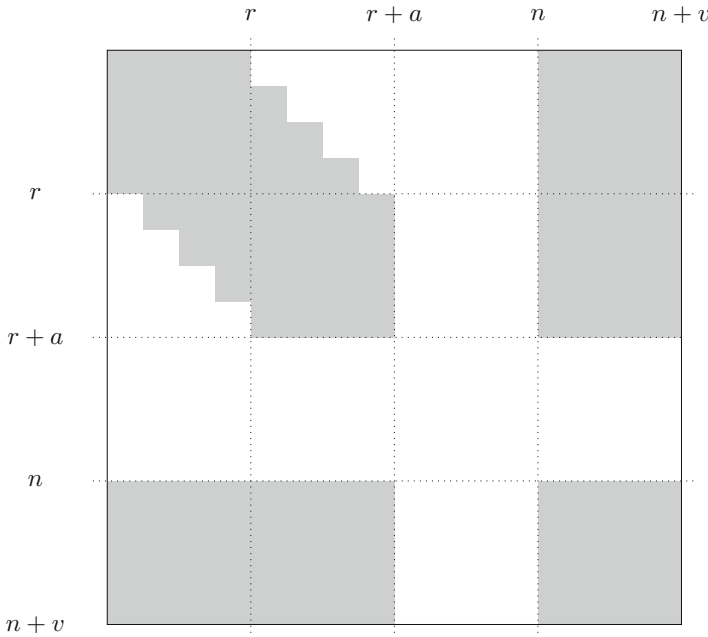


Fig. 4.2 Structure of the matrix \mathbf{F}^{*0} for HFEv-

$$\text{Complexity}_{\text{KSattack}} = \binom{n}{r+a+v}^\omega$$

with $2 < \omega \leq 3$ being the linear algebra constant.

4.4.5 The Gui Signature Scheme

When choosing parameters for HFEv-, one has to find a balance between security and efficiency of the scheme

- Security: the HFEv- central map should contain not too few quadratic terms; that is, $r = \lfloor \log_q(D-1) \rfloor + 1$ should not be too small.
- Efficiency: the inversion of the central map should be reasonably fast; that is, the degree D of the univariate polynomial \mathcal{F}_V must not be too large.

To find a balance between security and efficiency, the HFEv- signature scheme is therefore mainly used over the field $\text{GF}(2)$.

However, this brings up another problem: To reach a security of k bits against collision attacks, we need a hash value of length at least $2k$ bit. This means that the number of equations $n - a$ of a plain HFEv- scheme must be at least

$$n - a \geq \frac{2k}{\log_2(q)}.$$

For $q = 2$, this would lead to a very high number of equations in the system and therefore to a very large public key.

In order to deal with this problem, Petzoldt et al. introduced in [11] a specially designed signature generation process for HFEv-. Their Gui scheme generates ℓ HFEv- signatures for different hash values of the message d , and combines them to a single Gui signature σ of length $|\sigma| = (n - a) + \ell(a + v)$ bit. Analogously, the signature verification algorithm comprises ℓ evaluations of the public key \mathcal{P} .

The Gui signature scheme can be described as follows: Just as in the case of the HFEv- signature scheme, we have a finite field \mathbb{F} of q elements and a degree n extension field \mathbb{E} of \mathbb{F} . We have an isomorphism $\phi : \mathbb{F}^n \rightarrow \mathbb{E}$ between the vector space \mathbb{F}^n and the extension field \mathbb{E} . As in the case of HFEv- we use two integers a and v denoting the number of minus equations and vinegar variables. The only difference to HFEv- is the introduction of a new parameter $\ell \in \mathbb{N}$ (repetition factor).

Key Generation The key generation process of Gui works exactly as the key generation process of HFEv-. The only difference is that we choose an additional parameter ℓ (repetition factor), which is appended both at the private and public key. Therefore, we get

Public Key HFEv- public key \mathcal{P} , repetition factor ℓ .

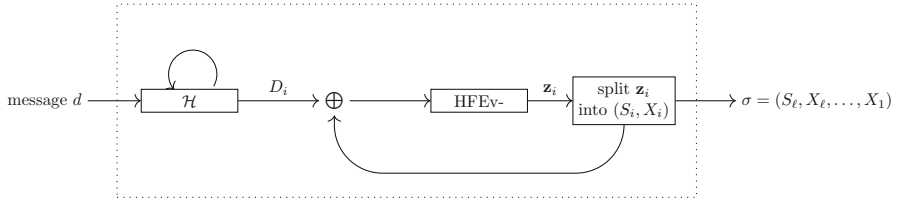


Fig. 4.3 Signature generation process of Gui

Private Key HFEv- private key $(\mathcal{S}, \mathcal{F}, \mathcal{T})$, repetition factor ℓ .

Signature Generation To generate a Gui signature for a document $d \in \{0, 1\}^*$, the signer uses a hash function $\mathcal{H} : \{0, 1\}^* \rightarrow \mathbb{F}^{n-a}$ to compute ℓ different hash values $D_1, \dots, D_\ell \in \mathbb{F}^{n-a}$ of the document d . He then sets $S_0 = \mathbf{0} \in \mathbb{F}^{n-a}$ and performs for $i = 1, \dots, \ell$ the following two steps

1. Generate an HFEv- signature $\sigma_i \in \mathbb{F}^{n+v}$ for the hash value $(D_i \oplus S_{i-1}) \in \mathbb{F}^{n-a}$.
2. Divide σ_i into two parts, i.e. $\sigma_i = (S_i, X_i)$ with $S_i \in \mathbb{F}^{n-a}$.

The final Gui signature is given by

$$\sigma = (S_\ell || X_\ell || \dots || X_1) \in \mathbb{F}^{(n-a)+\ell(a+v)}.$$

The process of generating a Gui signature is illustrated by Algorithm 4.1 and Fig. 4.3.

Signature Verification To check the authenticity of a signature $\sigma \in \mathbb{F}^{(n-a)+\ell(a+v)}$, we parse σ into $S_\ell, X_\ell, \dots, X_1$ and compute D_1, \dots, D_ℓ as shown above. For $i = \ell - 1$ to 0 we compute recursively $S_i = \mathcal{P}(S_{i+1} || X_{i+1}) - D_{i+1}$. The signature is accepted, if and only if $S_0 = \mathbf{0} \in \mathbb{F}^{n-a}$ holds (see Algorithm 4.2).

Algorithm 4.1 Signature generation process of Gui

Input: Gui private key $(\mathcal{S}, \mathcal{F}, \mathcal{T})$ message $d \in \{0, 1\}$, repetition factor ℓ

Output: signature $\sigma \in \mathbb{F}^{(n-a)+\ell(a+v)}$

- 1: $D_1 = \mathcal{H}(d)$
 - 2: **for** $i = 2$ to ℓ **do**
 - 3: $D_i = \mathcal{H}(D_{i-1})$
 - 4: **end for**
 - 5: $S_0 \leftarrow \mathbf{0} \in \mathbb{F}^{n-a}$
 - 6: **for** $i = 1$ to ℓ **do**
 - 7: $(S_i, X_i) \leftarrow \text{HFEv}^{-1}(D_i + S_{i-1})$
 - 8: **end for**
 - 9: $\sigma \leftarrow (S_\ell || X_\ell || \dots || X_1)$
 - 10: **return** σ
-

Algorithm 4.2 Signature verification process of Gui

Input: : Gui public key \mathcal{P} , message d , repetition factor ℓ , signature $\sigma \in \mathbb{F}^{(n-a)+\ell \cdot (a+v)}$

Output: : TRUE or FALSE

```

1:  $(S_\ell, X_\ell, \dots, X_1) \leftarrow \sigma$ 
2:  $D_1 \leftarrow \mathcal{H}(d)$ 
3: for  $i = 2$  to  $\ell$  do
4:    $D_i \leftarrow \mathcal{H}(D_{i-1})$ 
5: end for
6: for  $i = \ell - 1$  to 0 do do
7:    $S_i \leftarrow \mathcal{P}(S_{i+1} || X_{i+1}) - D_{i+1}$ 
8: end for
9: if  $S_0 = 0$  then
10:   return TRUE
11: else
12:   return FALSE
13: end if
```

4.4.6 Security

The security of Gui can be reduced to the security of the underlying HFEv- scheme. Therefore, the parameters (n, D, a, v) of Gui have to be chosen in a way such that direct and rank attacks against the underlying HFEv- scheme are infeasible.

For the choice of the repetition factor ℓ , we have to take the following theorem into consideration:

Theorem 4.8 *Let $G : \mathbb{F}^n \rightarrow \mathbb{F}^m$. A signature scheme, which generates a signature by combining k inversions of G (for k hash values of a message d) can be broken in $q^{\frac{k}{k+1}m}$ steps.*

Proof See [4], Theorem 3.2.1 □

In the case of Gui, this yields

Corollary 4.9 *The repetition factor ℓ of Gui has to be chosen in a way such that*

$$\frac{\ell}{\ell + 1} \geq \frac{\lambda}{n - a},$$

where λ is the required security level (in bit) of the scheme.

4.4.7 Key Sizes and Efficiency

The key sizes of Gui are exactly those of the HFEv- signature scheme (see Sect. 4.4.1).

Table 4.2 Parameters and key sizes of Gui

Security category	Parameters (n, D, a, v, ℓ)	Public key size (kB)	Private key size (kB)	Signature size (bit)
I	(184, 33, 16, 16, 3)	416.3	19.1	264
II	(312, 129, 24, 20, 2)	1955.1	59.3	376
III	(448, 513, 32, 28, 2)	5789.2	155.9	536

Since, during the signature generation process of Gui, we have to invert the HFEv- polynomial k times, the signature generation is k times slower than that of HFEv- and much slower than that of UOV and Rainbow (see Chap. 5). On the other hand, Gui provides the shortest signatures of all existing signature schemes (see Table 4.2).

References

1. L. Bettale, J. Faugère, L. Perret, Cryptanalysis of HFE, multi-HFE and variants for odd and even characteristic. *Des. Codes Cryptography* **69**(1), 1–52 (2013)
2. D. Cabarcas, D. Smith-Tone, J.A. Verbel, Key recovery attack for ZHFE, in *The Eighth International Conference on Post-Quantum Cryptography (PQCrypto 2017)*. Lecture Notes in Computer Science, vol. 10346 (Springer, Berlin, 2017), pp. 289–308
3. A. Casanova, J. Faugère, G. Macario-Rat, J. Patarin, L. Perret, J. Ryckeghem, GeMSS: a great multivariate short signature. <https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-2-Submissions>
4. N. Courtois, Generic attacks and the security of quartz, in *International Workshop on Public Key Cryptography (PKC 2003)*. Lecture Notes in Computer Science, vol. 2567 (Springer, Berlin, 2003), pp. 351–364
5. J. Ding, D. Schmidt, Cryptanalysis of HFEv and internal perturbation of HFE, in *International Workshop on Public Key Cryptography (PKC 2005)*. Lecture Notes in Computer Science, vol. 3386 (Springer, Berlin, 2005), pp. 288–301
6. J. Faugère, A. Joux, Algebraic cryptanalysis of hidden field equation (HFE) cryptosystems using gröbner bases, in *Annual International Cryptology Conference (CRYPTO 2003)*. Lecture Notes in Computer Science, vol. 2729 (Springer, Berlin, 2003), pp. 44–60
7. Y. Ikematsu, D.H. Duong, A. Petzoldt, T. Takagi, An efficient key generation of ZHFE public key cryptosystem. *IEICE Trans.* **101-A**(1), 29–38 (2018)
8. A. Kipnis, A. Shamir, Cryptanalysis of the HFE public key cryptosystem by relinearization, in *Annual International Cryptology Conference (CRYPTO 1999)*. Lecture Notes in Computer Science, vol. 1666 (Springer, Berlin, 1999), pp. 19–30
9. M.S.E. Mohamed, J. Ding, J.A. Buchmann, Towards algebraic cryptanalysis of HFE challenge 2, in *International Conference on Information Security and Assurance (ISA 2011)*. Communications in Computer and Information Science, vol. 200 (Springer, Berlin, 2011), pp. 123–131

10. J. Patarin, Hidden field equations (HFE) and isomorphisms of polynomials (IP): two new families of asymmetric algorithms, in *International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 1996)*. Lecture Notes in Computer Science, vol. 1070 (Springer, Berlin, 1996), pp. 33–48
11. A. Petzoldt, M.-S. Chen, B.-Y. Yang, T. Chengdong, J. Ding, Design principles for HFEv-based signature schemes, in *International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2015 - Part I)*. Lecture Notes in Computer Science, vol. 9452 (Springer, Berlin, 2015), pp. 1–24
12. J. Porras, J. Baena, J. Ding, ZHFE, a new multivariate public key encryption scheme, in *6th International Workshop Post-quantum Cryptography (PQCrypto 2014)*. Lecture Notes in Computer Science, vol. 8772 (Springer, Berlin, 2014), pp. 229–245

Chapter 5

Oil and Vinegar



Abstract This chapter deals with multivariate signature schemes following the concept of Oil and Vinegar. After introducing the basic (balanced) Oil and Vinegar (OV) signature scheme, we describe its cryptanalysis by the invariant subspace attack of Kipnis and Shamir and how this attack is prevented in the unbalanced Oil and Vinegar scheme (UOV). We then introduce the signature scheme Rainbow, which can be seen as multi-layer version of UOV and discuss its security and efficiency. Finally, we present techniques to reduce the public key size of UOV and Rainbow.

In this chapter we discuss multivariate signature schemes of the so called Oil and Vinegar type. In contrast to the multivariate schemes presented in the previous chapters, the schemes discussed here are SingleField schemes, which means that all the computations are performed over a relatively small finite field, enabling efficient implementations of the schemes. Furthermore, the public key of all the schemes discussed in this chapter contains more variables than equations, which restricts the schemes to being used as signature schemes.¹

The basic idea of Oil and Vinegar was inspired by Patarin's Linearization Equations attack against the Matsumoto–Imai cryptosystem (see Sect. 3.2). This was the first time in the history of cryptography that an attack method was transformed into a cryptographic scheme. After Patarin's first attempt [11] called Balanced Oil and Vinegar (OV) was defeated by a linear algebra attack of Kipnis and Shamir [10], the Unbalanced Oil and Vinegar scheme (UOV) was proposed [9].

Later, in order to reduce key sizes and increase the efficiency of the scheme, Ding and Schmidt [5] proposed the Rainbow signature scheme, which can be seen as multi-layer version of UOV. Recently, Petzoldt proposed a technique to generate structured UOV and Rainbow public keys which helps to reduce the key sizes of these schemes significantly [13, 14]. Another attempt in this direction is the LUOV signature scheme of Beullens et al. [2]. An algorithm speeding up the key generation of Rainbow significantly was proposed by Petzoldt in [12].

¹ An encryption scheme of the SingleField type is presented in Chap. 7.

This chapter is organized as follows: In the first section of this chapter (Sect. 5.1), we present the basic Oil and Vinegar signature scheme as proposed by Patarin and show its workflow using a toy example. In Sect. 5.2 we discuss the attack of Kipnis and Shamir against the balanced Oil and Vinegar scheme and how this attack is prevented by the unbalanced Oil and Vinegar (UOV) signature scheme. Section 5.3 considers the security of this scheme and proposes concrete parameter sets. Section 5.4 presents the Rainbow signature scheme both in theory and using a toy example. In Sect. 5.5 we discuss the security of Rainbow, describe the known attacks against the scheme and derive from this practical parameter sets. Section 5.6, presents techniques to reduce the public key size of UOV and Rainbow. Finally, in the last section of this chapter (Sect. 5.7), we describe a newly developed technique to speed up the key generation process of Rainbow.

5.1 The Oil and Vinegar Signature Scheme

In this section we present the basic Oil and Vinegar signature scheme as proposed by Patarin in [11].

Let $\mathbb{F} = \mathbb{F}_q$ be a finite field with q elements and o, v be integers. In the original paper of Patarin o was chosen to be equal to v (balanced Oil and Vinegar), but we do not require this here. The number of equations in the scheme is equal to o , the number of variables is given by $n = o + v$. Furthermore, we define the index sets $V = \{1, \dots, v\}$ and $O = \{v + 1, \dots, n\}$. We denote the variables x_i ($i \in V$) as vinegar variables, the variables x_{v+1}, \dots, x_n as Oil variables.

Key Generation In order to create a key pair for the Oil and Vinegar signature scheme, Alice chooses an affine map $\mathcal{T} : \mathbb{F}^n \rightarrow \mathbb{F}^n$ with randomly chosen coefficients and an OV central map $\mathcal{F} = (f^{(1)}, \dots, f^{(o)}) : \mathbb{F}^n \rightarrow \mathbb{F}^o$. The polynomials $f^{(1)}, \dots, f^{(o)}$ are of the form

$$f^{(i)} = \sum_{j,k \in V} \alpha_{j,k}^{(i)} x_j x_k + \sum_{j \in V, k \in O} \beta_{j,k}^{(i)} x_j x_k + \sum_{j \in V \cup O} \gamma_j^{(i)} x_j + \delta^{(i)} \quad (i = 1, \dots, o)$$

with coefficients $\alpha_{j,k}^{(i)}, \beta_{j,k}^{(i)}, \gamma_j^{(i)}$ and $\delta^{(i)}$ randomly chosen from the field \mathbb{F} . These polynomials are denoted as Oil and Vinegar polynomials. The name is derived from the fact that, in the polynomials $f^{(1)}, \dots, f^{(o)}$, the variables x_1, \dots, x_n are not fully mixed, just like oil and vinegar in a salad dressing.

Private Key The private key of the Oil and Vinegar signature scheme consists of the two maps $\mathcal{F} : \mathbb{F}^n \rightarrow \mathbb{F}^o$ and $\mathcal{T} : \mathbb{F}^n \rightarrow \mathbb{F}^n$.

Public Key The public key \mathcal{P} of the Oil and Vinegar signature scheme is the composed map $\mathcal{P} = \mathcal{F} \circ \mathcal{T}$ and consists of o quadratic polynomials in n variables. Note that, in contrast to the standard construction of multivariate cryptography (see Chap. 2), we do not use a second affine map \mathcal{S} in the construction of the public key of

the Oil and Vinegar scheme. The reason for this is, that any linear combination of the Oil and Vinegar polynomials $f^{(1)}, \dots, f^{(o)}$ is again an Oil and Vinegar polynomial of the same structure. Therefore, the use of \mathcal{S} does not increase the security of the scheme and we can omit it.

5.1.1 Properties of the Central Map

As already explained in Chap. 2, the homogeneous quadratic part of the polynomials $f^{(1)}, \dots, f^{(o)}$ can be written as quadratic forms, i.e. $\hat{f}^{(i)}(\mathbf{x}) = \mathbf{x}^T \cdot F^{(i)} \cdot \mathbf{x}$ ($i = 1, \dots, o$). Here, $\hat{f}^{(i)}$ denotes the homogeneous quadratic part of the polynomial $f^{(i)}$. Due to the special structure of the Oil and Vinegar polynomials $f^{(i)}$, the matrices $F^{(i)} \in \mathbb{F}^{n \times n}$ have the form

$$F^{(i)} = \begin{pmatrix} \star_{v \times v} & \star_{v \times o} \\ \star_{o \times v} & 0_{o \times o} \end{pmatrix}. \quad (5.1)$$

A key pair of the Oil and Vinegar signature scheme can be described as follows.

Inverting the Central Map The special structure of the Oil and Vinegar polynomials enables us to invert the central map \mathcal{F} efficiently. Remember that \mathcal{F} is an underdetermined map with o equations in $o + v$ variables. Therefore, even after fixing v of the variables, the resulting determined system will be solvable with high probability. Due to the special structure of the polynomials $f^{(i)}$ we can, by fixing the vinegar variables x_1, \dots, x_v , transform the quadratic map \mathcal{F} into a system of o linear equations in the o Oil variables x_{v+1}, \dots, x_n . This system can easily be solved by Gaussian elimination.

Note that this process to invert the central map \mathcal{F} is very similar to the second step of the Linearization Equations attack (see Sect. 3.2), where the given ciphertext is substituted into the linearization equations to obtain a linear system in the plaintext variables.

With the knowledge of how to invert the central map, we can proceed in the description of the scheme.

Signature Generation To generate a signature $\mathbf{z} \in \mathbb{F}^n$ for a document d , one uses a hash function $\mathcal{H} : \{0, 1\}^* \rightarrow \mathbb{F}^o$ to compute the hash value $\mathbf{w} = \mathcal{H}(d) \in \mathbb{F}^o$ and performs the following 2 steps.

1. Find a pre-image $\mathbf{y} \in \mathbb{F}^n$ of \mathbf{w} under the central map \mathcal{F} .
 - Choose random values for the vinegar variables y_1, \dots, y_v and substitute them into the polynomials $f^{(1)}, \dots, f^{(o)}$.

- Solve the resulting linear system of o equations in the o Oil variables y_{v+1}, \dots, y_n by Gaussian elimination. If the system does not have a solution, choose other values for the vinegar variables x_1, \dots, x_v and try again.²

2. Compute the signature $\mathbf{z} \in \mathbb{F}^n$ by $\mathbf{z} = \mathcal{T}^{-1}(\mathbf{y})$.

Signature Verification To check, if $\mathbf{z} \in \mathbb{F}^n$ is indeed a valid signature for the document d , one uses the hash function \mathcal{H} to compute $\mathbf{w} = \mathcal{H}(d) \in \mathbb{F}^o$ and computes $\mathbf{w}' = \mathcal{P}(\mathbf{z}) \in \mathbb{F}^o$. If $\mathbf{w}' = \mathbf{w}$ holds, the signature \mathbf{z} is accepted, otherwise rejected.

5.1.2 Key Sizes and Efficiency

The size of the UOV public key is

$$\text{size}_{\text{pk UOV}} = o \frac{(n+1)(n+2)}{2}$$

field elements, the size of the private key

$$\text{size}_{\text{sk UOV}} = \underbrace{n(n+1)}_{\text{map } \mathcal{T}} + o \underbrace{\left(\frac{v(v+1)}{2} + ov + n + 1 \right)}_{\text{map } \mathcal{F}}$$

field elements. In contrast to the HFEv- scheme (see Sect. 4.4), the signature generation process of UOV only requires the solution of a linear system, which can be efficiently done by Gaussian elimination. Therefore, the UOV signature scheme can be implemented much easier and much more efficient than HFEv-.

5.1.3 Toy Example

In the following we present the general workflow of the Oil and Vinegar signature scheme using a toy example. We choose $\mathbb{F} = GF(4)$ as the base field and $(o, v) = (3, 3)$, leading to a public key of 3 quadratic equations in 6 variables.

The private key of our scheme consists of the affine map $\mathcal{T} : \mathbb{F}^6 \rightarrow \mathbb{F}^6$,

²The probability that the linear system of o equations in o variables does not have a solution is about $1/q$, where q is the size of the underlying field. Therefore, for reasonably large q (e.g. $q \in \{31, 256\}$), we usually find a solution at the first try.

$$\mathcal{T}(x_1, \dots, x_6) = \begin{pmatrix} \alpha & \alpha & \alpha & 0 & 1 & \alpha^2 \\ \alpha^2 & 1 & 1 & \alpha & 1 & \alpha \\ \alpha^2 & 0 & 0 & \alpha & \alpha^2 & \alpha \\ \alpha & \alpha & 0 & \alpha & \alpha^2 & \alpha \\ 1 & 0 & 1 & \alpha^2 & 0 & 0 \\ \alpha^2 & \alpha & 1 & \alpha^2 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ \alpha^2 \\ \alpha^2 \\ \alpha \\ 0 \end{pmatrix}$$

and the OV central map $\mathcal{F} : \mathbb{F}^6 \rightarrow \mathbb{F}^3$ given by the polynomials

$$\begin{aligned} f^{(1)} &= x_1^2 + x_1x_2 + \alpha^2x_1x_3 + \alpha x_1x_4 + x_1x_5 + \alpha^2x_1x_6 + x_2^2 + x_2x_3 + \alpha x_2x_4 \\ &\quad + x_2x_6 + x_2 + \alpha^2x_3x_4 + \alpha x_3x_6 + x_3 + \alpha^2x_4 + \alpha^2x_5 + \alpha^2, \\ f^{(2)} &= x_1^2 + x_1x_3 + \alpha x_1x_4 + x_1x_5 + \alpha x_1x_6 + x_1 + \alpha^2x_2^2 + \alpha^2x_2x_3 + x_2x_4 \\ &\quad + x_2x_5 + \alpha x_2x_6 + \alpha x_3^2 + \alpha x_3x_4 + x_3x_6 + x_3 + \alpha^2x_4 + \alpha x_5 + \alpha x_6 + \alpha^2, \\ f^{(3)} &= \alpha x_1^2 + \alpha^2x_1x_2 + \alpha^2x_1x_4 + \alpha x_1x_5 + \alpha x_1x_6 + \alpha x_1 + \alpha^2x_2x_5 \\ &\quad + \alpha x_2x_6 + \alpha x_2 + x_3^2 + \alpha x_3x_5 + \alpha^2x_3 + x_4 + x_5 + \alpha^2x_6 + \alpha^2 \end{aligned}$$

We compute the public key $\mathcal{P} = (p^{(1)}, p^{(2)}, p^{(3)}) : \mathbb{F}^6 \rightarrow \mathbb{F}^3$ by $\mathcal{P} = \mathcal{F} \circ \mathcal{T}$, obtaining

$$\begin{aligned} p^{(1)} &= x_1^2 + \alpha x_1x_2 + \alpha x_1x_3 + \alpha x_1x_4 + \alpha x_1x_5 + \alpha x_1x_6 + x_1 + \alpha x_2^2 \\ &\quad + x_2x_4 + \alpha x_2x_5 + \alpha x_2x_6 + \alpha x_2 + \alpha x_3^2 + x_3x_4 + \alpha x_3x_5 \\ &\quad + \alpha x_3x_6 + \alpha x_4x_5 + \alpha x_4 + \alpha x_5^2 + \alpha^2x_5 + x_6^2 + x_6 + \alpha, \\ p^{(2)} &= x_1^2 + \alpha x_1x_4 + \alpha x_1x_5 + \alpha x_1x_6 + \alpha x_1 + x_2^2 + \alpha^2x_2x_3 + x_2x_5 \\ &\quad + x_2x_6 + \alpha x_3^2 + \alpha^2x_3x_4 + x_3 + x_4^2 + \alpha^2x_4x_5 + \alpha^2x_4 + \alpha x_5^2 \\ &\quad + x_5x_6 + \alpha x_5 + \alpha x_6^2 + 1, \\ p^{(3)} &= \alpha^2x_1^2 + \alpha x_1x_2 + x_1x_3 + x_1x_4 + \alpha^2x_1x_5 + \alpha^2x_1 + x_2x_3 + \alpha x_2x_4 \\ &\quad + x_2x_5 + \alpha x_2x_6 + x_3^2 + \alpha x_3x_4 + \alpha^2x_3x_6 + \alpha x_3 + \alpha^2x_4x_5 \\ &\quad + \alpha^2x_4x_6 + x_4 + x_5^2 + x_5x_6 + \alpha^2x_5 + x_6 + 1. \end{aligned}$$

In order to generate a OV signature for the message $\mathbf{w} = (1, \alpha, \alpha)$, we first need to compute $\mathbf{y} = \mathcal{F}^{-1}(\mathbf{w})$. To do this, we choose random values for the vinegar variables x_1, x_2, x_3 , e.g. $(x_1, x_2, x_3) = (\alpha, 1, 1)$ and substitute them into the polynomials $f^{(1)}, f^{(2)}$ and $f^{(3)}$. By doing so, we obtain a linear system in the Oil variables x_4, x_5 and x_6 of the form

$$\begin{aligned}
\tilde{f}^{(1)} &= x_4 + x_5 + \alpha x_6 = \alpha \\
\tilde{f}^{(2)} &= \alpha^2 x_4 + x_5 + \alpha x_6 = 1 \\
\tilde{f}^{(3)} &= \alpha^2 x_5 + \alpha x_6 = \alpha.
\end{aligned}$$

This system has the solution $(x_4, x_5, x_6) = (\alpha, 1, \alpha^2)$. Attaching the vinegar variables yields

$$\mathbf{y} = \mathcal{F}^{-1}(\mathbf{w}) = (\alpha, 1, 1, \alpha, 1, \alpha^2).$$

Finally, we compute

$$\mathbf{z} = \mathcal{T}^{-1}(\mathbf{y}) = (0, \alpha, 0, 1, \alpha^2, 1)$$

to obtain a signature $\mathbf{z} \in \mathbb{F}^6$ for the message \mathbf{w} .

In order to check if \mathbf{z} is indeed a valid signature for the message \mathbf{w} , we compute

$$\mathbf{w}' = \mathcal{P}(\mathbf{z}) = (1, \alpha, \alpha).$$

Since $\mathbf{w}' = \mathbf{w}$ holds, the signature is accepted.

5.2 The Kipnis–Shamir Attack on Balanced Oil and Vinegar and UOV

In [10], Kipnis and Shamir proposed a powerful attack against the balanced Oil and Vinegar signature scheme ($n = 2v$), which finds an equivalent private key in polynomial time. This key can then be used to generate signatures for arbitrary messages. To simplify our description, we assume that the components of the UOV central map \mathcal{F} are homogeneous quadratic polynomials and that the transformation \mathcal{T} is linear. Note that, in this case, the UOV public key $\mathcal{P} = \mathcal{F} \circ \mathcal{T}$ will be a homogeneous quadratic map, too.

Let $f(\mathbf{x})$ be a central polynomial. Using the above simplification, we can write $f(\mathbf{x})$ as a quadratic form $f(\mathbf{x}) = \mathbf{x}^T \cdot F \cdot \mathbf{x}$ with an $n \times n$ matrix F of the form

$$F = \begin{pmatrix} F_1 & F_2 \\ F_3 & 0_{v \times v} \end{pmatrix} \quad (5.2)$$

with all F_1, F_2, F_3 and $0_{v \times v}$ being $v \times v$ matrices with entries in \mathbb{F} .

The matrix P representing the quadratic form of the corresponding public polynomial $p(\mathbf{x})$ is given as

$$P = T^T \cdot F \cdot T,$$

where T is the matrix representing the linear transformation \mathcal{T} .

For the description of the attack we need the following definition.

Definition 5.1 We define the Oil subspace of \mathbb{F}^n as

$$\mathcal{O} = \{\mathbf{x} = (x_1, \dots, x_n)^T \in \mathbb{F}^n : x_1 = \dots = x_v = 0\}.$$

The Vinegar subspace is the set

$$\mathcal{V} = \{\mathbf{x} = (x_1, \dots, x_n)^T \in \mathbb{F}^n : x_{v+1} = \dots = x_n = 0\}.$$

Note that we have $n = 2v$.

Then we have

Lemma 5.2

1. For any $\mathbf{u}_1, \mathbf{u}_2 \in \mathcal{O}$ we have

$$\mathbf{u}_1^T \cdot F \cdot \mathbf{u}_2 = 0.$$

2. For any $\mathbf{v}_1, \mathbf{v}_2 \in \mathcal{T}^{-1}(\mathcal{O})$ we have

$$\mathbf{v}_1^T \cdot P \cdot \mathbf{v}_2 = 0.$$

Proof

1. Since $\mathbf{u}_1, \mathbf{u}_2 \in \mathcal{O}$, we can write $\mathbf{u}_1 = (0, \mathbf{u}'_1)^T$ and $\mathbf{u}_2 = (0, \mathbf{u}'_2)^T$. Therefore we get

$$\begin{aligned} \mathbf{u}_1^T \cdot F \cdot \mathbf{u}_2 &= (0, \mathbf{u}'_1) \cdot \begin{pmatrix} F_1 & F_2 \\ F_3 & 0_v \end{pmatrix} \cdot \begin{pmatrix} 0 \\ \mathbf{u}'_2 \end{pmatrix} \\ &= (0, \mathbf{u}'_1) \cdot \begin{pmatrix} F_2 \cdot \mathbf{u}'_2 \\ 0 \end{pmatrix} = 0. \end{aligned}$$

2. Let $\mathbf{v}'_1, \mathbf{v}'_2 \in \mathcal{O}$ such that $\mathbf{v}_1 = \mathcal{T}^{-1}(\mathbf{v}'_1)$ and $\mathbf{v}_2 = \mathcal{T}^{-1}(\mathbf{v}'_2)$. Thus we get

$$\begin{aligned} \mathbf{v}_1^T \cdot P \cdot \mathbf{v}_2 &= (T^{-1} \cdot \mathbf{v}'_1)^T \cdot P \cdot (T^{-1} \cdot \mathbf{v}'_2) \\ &= \mathbf{v}'_1{}^T \cdot (T^T)^{-1} \cdot T^T \cdot F \cdot T \cdot T^{-1} \cdot \mathbf{v}_2 \\ &= \mathbf{v}'_1{}^T \cdot F \cdot \mathbf{v}'_2 = 0. \end{aligned}$$

The last “=” holds because of 1. □

The goal of the attack of Kipnis and Shamir is to find the pre-image of the Oil subspace under the map \mathcal{T} .

Let $E : \mathbb{F}^n \rightarrow \mathbb{F}^n$ be a linear transformation of the form (5.2). Then we have

Lemma 5.3

1. $E(\mathcal{O}) \subset \mathcal{V}$.
2. If E is invertible, we have $E(\mathcal{O}) = \mathcal{V}$ and $E^{-1}(\mathcal{V}) = \mathcal{O}$.

Proof

1. Let $\mathbf{o} = (0, \mathbf{o}') \in \mathcal{O}$. Then we have

$$\begin{pmatrix} E_1 & E_2 \\ E_3 & 0_{v \times v} \end{pmatrix} \cdot \begin{pmatrix} 0 \\ \mathbf{o}' \end{pmatrix} = \begin{pmatrix} E_2 \cdot \mathbf{o}' \\ 0 \end{pmatrix} \in \mathcal{V}.$$

2. If E is invertible, the image space of $E(\mathcal{O})$ has dimension $\dim(\mathcal{O}) = v$, and therefore we have $E(\mathcal{O}) = \mathcal{V}$ and $E^{-1}(\mathcal{V}) = \mathcal{O}$. \square

For the following we denote the matrix associated with the i -th component of the central map by $F^{(i)}$. Note that $F^{(i)}$ is of the form of (5.2). Analogously, we set $P^{(i)}$ to be the matrix associated to the i -th component of the public key. Note that we have $P^{(i)} = T^T \cdot F^{(i)} \cdot T$ for every $i \in \{1, \dots, o\}$.

Let H_1 and H_2 be linear combinations of the matrices $F^{(i)}$. Note that H_1 and H_2 are of the form of (5.2). In addition to this assume that the matrix H_1 is invertible. We obtain

Corollary 5.4 *The oil subspace \mathcal{O} is a common invariant subspace of all matrices $H = H_1^{-1} \cdot H_2$.*

Proof This follows directly from Lemma 5.3. \square

Let W_1 and W_2 be linear combinations of the matrices $P^{(i)}$ ($i = 1, \dots, o$) and assume that W_1 is invertible. Note that W_1 and W_2 can be written as

$$W_1 = T^T \cdot \hat{F}_1 \cdot T \quad \text{and} \quad W_2 = T^T \cdot \hat{F}_2 \cdot T$$

for some matrices \hat{F}_1 and \hat{F}_2 of the form (5.2). We find

Theorem 5.5 *The space $\mathcal{T}^{-1}(\mathcal{O})$ is a common invariant subspace of all the matrices $W = W_1^{-1} \cdot W_2$.*

Proof

$$\begin{aligned} W_1^{-1} \cdot W_2(\mathcal{T}^{-1}(\mathcal{O})) &= (T^T \cdot \hat{F}_1 \cdot T)^{-1} \cdot T^T \cdot \hat{F}_2 \cdot T \cdot \mathcal{T}^{-1}(\mathcal{O}) \\ &= T^{-1} \cdot \hat{F}_1^{-1} \cdot (T^T)^{-1} \cdot T^T \cdot \hat{F}_2 \cdot T \cdot \mathcal{T}^{-1}(\mathcal{O}) \\ &= T^{-1} \cdot \hat{F}_1^{-1} \cdot \hat{F}_2(\mathcal{O}) \end{aligned}$$

$$= \mathcal{T}^{-1}(\mathcal{O}).$$

Here, the last “=” holds due to the fact that \mathcal{O} is an invariant subspace of $\hat{F}_1^{-1} \cdot \hat{F}_2$ (Corollary 5.4). \square

The problem of finding this subspace can be solved by standard linear algebra techniques. After having found $\mathcal{T}^{-1}(\mathcal{O})$, we know the relevant part of the transformation \mathcal{T} , which then can be used to compute an equivalent private key $(\tilde{\mathcal{F}}, \tilde{\mathcal{T}})$ which again can be used to generate signatures for arbitrary messages.

We present now two probabilistic polynomial time algorithms for finding the space $\mathcal{T}^{-1}(\mathcal{O})$ (one for fields of odd characteristic the other for even characteristic). The algorithms take a random linear combination W_2 of the matrices $P^{(i)}$ associated to the public key polynomials and multiply it by an invertible matrix $W_1 = \sum_{i=1}^o \lambda_i P^{(i)}$ to obtain a matrix W of the form $W = W_1^{-1} \cdot W_2$.

The algorithms then compute the so called minimal invariant subspaces (an invariant subspace which contains no non-trivial invariant subspaces) of this matrix. These subspaces correspond to the irreducible factors of the characteristic polynomial of W , and can be found in probabilistic polynomial time using standard linear algebra techniques. Each minimal invariant subspace of W may or may not be a subspace of $\mathcal{T}^{-1}(\mathcal{O})$.

However, by Lemma 5.2 (2.), we can distinguish between “correct” and “false” subspaces. We continue this process until having found o linear independent basis vectors of $\mathcal{T}^{-1}(\mathcal{O})$.

5.2.1 The Case of q Odd

In the case of an underlying field of odd characteristic we can write the homogeneous quadratic part of the public polynomials $p^{(1)}(\mathbf{x}), \dots, p^{(o)}(\mathbf{x})$ as quadratic forms

$$\mathbf{x}^T \cdot \bar{Q}^{(1)} \cdot \mathbf{x}, \dots, \mathbf{x}^T \cdot \bar{Q}^{(o)} \cdot \mathbf{x}$$

with symmetric matrices $\bar{Q}^{(i)}$ ($i = 1, \dots, o$) (see Sect. 2.1). Hereby, the entries $q_{jk}^{(i)}$ of the matrix $\bar{Q}^{(i)}$ are given as

$$q_{jk}^{(i)} = \begin{cases} \text{MonomialCoefficient}(p^{(i)}, x_j^2) & j = k, \\ \text{MonomialCoefficient}(p^{(i)}, x_j x_k) / 2 & j \neq k. \end{cases}$$

We define $\Omega = \text{span}(\bar{Q}^{(1)}, \dots, \bar{Q}^{(o)})$. Let W_1 and W_2 be elements of Ω (W_1 must be invertible) and set $W = W_1^{-1} \cdot W_2$. As Theorem 5.5 states, the desired space $\mathcal{T}^{-1}(\mathcal{O})$ will be an invariant subspace of W .

In the following, we compute the minimal invariant subspaces of the matrix W (i.e. the invariant subspaces not containing a non-trivial invariant subspace). Each of these minimal invariant subspaces might or might not be a subspace of $\mathcal{T}^{-1}(\mathcal{O})$. This can be checked using the test provided in part 2 of Lemma 5.2.

Algorithm 5.1 Attack on balanced oil and vinegar (q odd)

Input: OV public key $\mathcal{P} = (p^{(1)}, \dots, p^{(o)})$

Output: equivalent private key $(\tilde{\mathcal{F}}, \tilde{\mathcal{T}})$

- 1: Compute the symmetric matrices $\tilde{Q}^{(i)}$ associated to the components of the public key. Denote the span of these matrices by Ω .
 - 2: Set $\mathcal{B} = \emptyset$.
 - 3: **repeat**
 - 4: Choose randomly two matrices W_1 and $W_2 \in \Omega$ and test if W_1 is invertible. If so, set $W = W_1^{-1} \cdot W_2$.
 - 5: Compute the characteristic polynomial $C(\lambda)$ of the matrix W and check if it has only quadratic factors. If so, set $C_1(\lambda) = \sqrt{C(\lambda)}$ and compute $C_1(W)$. Otherwise, go back to step 4.
 - 6: Find a basis $\mathbf{v}_1, \dots, \mathbf{v}_k$ of the image space of $C_1(W)$.
 - 7: For $i = 1, \dots, k$, use Lemma 5.2 (2.) to check whether the vector \mathbf{v}_i lies in the desired space $\mathcal{T}^{-1}(\mathcal{O})$. If yes, add \mathbf{v}_i to \mathcal{B} .
 - 8: **until** $\dim(\mathcal{B}) = o$.
 - 9: Extend \mathcal{B} to a basis of the whole space \mathbb{F}^n and insert the basis vectors into the columns of the matrix \tilde{T}^{-1} .
 - 10: Compute, for $i = 1, \dots, o$, $\tilde{F}^{(i)} = (\tilde{T}^{-1})^T \circ P^{(i)} \circ \tilde{T}^{-1}$.
 - 11: Set $\tilde{\mathcal{F}} = (\tilde{F}^{(1)}, \tilde{F}^{(2)}, \dots, \tilde{F}^{(o)})$.
 - 12: **return** $(\tilde{\mathcal{F}}, \tilde{\mathcal{T}})$
-

In order to find the minimal invariant subspaces of the matrix W , we compute the characteristic polynomial $C(\lambda)$ of this matrix. If $C(\lambda)$ contains only quadratic factors, we set $C(\lambda) = C_1(\lambda)^2$.

The desired basis vectors of the invariant subspaces are now located in the image space of $C_1(W)$. If $C_1(W) = \{0\}$. We can not find such a basis vector and we have to choose other linear combinations W_1 and W_2 . Otherwise, let $\mathbf{v}_1, \dots, \mathbf{v}_k$ be a basis of the image space of $C_1(W)$. Check, for $i = 1, \dots, k$, if the vector \mathbf{v}_i is an element $\mathcal{T}^{-1}(\mathcal{O})$. If not, discard it. Denote the remaining basis vectors by $\mathbf{v}_1, \dots, \mathbf{v}_\ell$. If $\ell = o$ holds, we have found a basis of $\mathcal{T}^{-1}(\mathcal{O})$ and we are ready.

Otherwise, we have to use other linear combinations W_1 and W_2 to find additional vectors $\mathbf{v}_{\ell+1}, \dots, \mathbf{v}_o$ to extend $\mathbf{v}_1, \dots, \mathbf{v}_\ell$ to a basis of $\mathcal{T}^{-1}(\mathcal{O})$. In the last step of the attack, we extend the basis $\mathbf{v}_1, \dots, \mathbf{v}_o$ to a basis of the whole space \mathbb{F}^n . These basis vectors build the columns of an equivalent matrix \tilde{T}^{-1} . Finally, we compute the matrices $\tilde{F}^{(i)}$ representing the components of the equivalent central map $\tilde{\mathcal{F}}$ by

$$\tilde{F}^{(i)} = (\tilde{T}^{-1})^T \cdot P^{(i)} \cdot \tilde{T}^{-1} \quad (i = 1, \dots, o).$$

The map $\tilde{\mathcal{F}}$ and the matrix $\tilde{\mathcal{T}}$ can now be used to generate signatures for arbitrary messages. Algorithm 5.1 shows the single steps of the attack in a compact form.

5.2.2 Toy Example

In our toy example we choose $q = 7$ and $o = v = 3$. The public key of our balanced OV instance has the form $\mathcal{P} = (p^{(1)}, p^{(2)}, p^{(3)})$ with

$$\begin{aligned} p^{(1)}(x_1, \dots, x_6) = & x_1x_2 + 6x_1x_4 + x_1x_5 + 2x_1x_6 + 5x_2x_3 + 3x_2x_4 + 3x_2x_5 \\ & + 5x_2x_6 + 5x_3^2 + 2x_3x_4 + 4x_3x_5 + 3x_3x_6 + 5x_4^2 + 4x_4x_5 \\ & + 4x_4x_6 + 4x_5^2 + 2x_6^2, \end{aligned}$$

$$\begin{aligned} p^{(2)}(x_1, \dots, x_6) = & 5x_1^2 + 3x_1x_2 + 3x_1x_3 + 5x_1x_4 + x_1x_6 + 6x_2^2 + 5x_2x_3 + 5x_2x_4 \\ & + 3x_2x_5 + x_2x_6 + 6x_3^2 + 3x_3x_4 + 2x_3x_5 + x_3x_6 + 3x_4x_5 \\ & + 3x_4x_6 + x_5x_6 + 3x_6^2, \end{aligned}$$

$$\begin{aligned} p^{(3)}(x_1, \dots, x_6) = & 6x_1^2 + 2x_1x_2 + 2x_1x_3 + 5x_1x_5 + 5x_2^2 + 3x_2x_3 + 6x_2x_4 + 4x_2x_5 \\ & + x_2x_6 + 6x_3^2 + 2x_3x_4 + 4x_4^2 + 3x_4x_5 + 4x_4x_6 + 2x_5^2 + 3x_6^2. \end{aligned}$$

Therefore, the symmetric matrices $\bar{Q}_1, \bar{Q}_2, \bar{Q}_3$ representing the public polynomials are given by

$$\begin{aligned} \bar{Q}^{(1)} &= \begin{pmatrix} 0 & 4 & 0 & 3 & 4 & 1 \\ 4 & 0 & 6 & 5 & 5 & 6 \\ 0 & 6 & 5 & 1 & 2 & 5 \\ 3 & 5 & 1 & 5 & 2 & 2 \\ 4 & 5 & 2 & 2 & 4 & 0 \\ 1 & 6 & 5 & 2 & 0 & 2 \end{pmatrix}, \\ \bar{Q}^{(2)} &= \begin{pmatrix} 5 & 5 & 5 & 6 & 0 & 3 \\ 5 & 6 & 6 & 6 & 5 & 4 \\ 5 & 6 & 6 & 5 & 1 & 4 \\ 6 & 6 & 5 & 0 & 5 & 5 \\ 0 & 5 & 1 & 5 & 0 & 4 \\ 3 & 4 & 4 & 5 & 4 & 3 \end{pmatrix}, \\ \bar{Q}^{(3)} &= \begin{pmatrix} 6 & 1 & 1 & 0 & 6 & 0 \\ 1 & 5 & 5 & 3 & 2 & 4 \\ 1 & 5 & 6 & 1 & 0 & 0 \\ 0 & 3 & 1 & 4 & 5 & 5 \\ 6 & 2 & 0 & 5 & 2 & 0 \\ 0 & 4 & 0 & 5 & 0 & 3 \end{pmatrix}. \end{aligned}$$

We define the matrices W_1 and W_2 by

$$\begin{aligned} W_1 &= 4\bar{Q}^{(1)} + 3\bar{Q}^{(2)} + 4\bar{Q}^{(3)} \\ W_2 &= 3\bar{Q}^{(1)} + 4\bar{Q}^{(2)} + \bar{Q}^{(3)} \end{aligned}$$

and obtain

$$W = W_1^{-1} \cdot W_2 = \begin{pmatrix} 2 & 6 & 5 & 5 & 1 & 0 \\ 6 & 0 & 3 & 6 & 6 & 2 \\ 3 & 0 & 2 & 4 & 4 & 0 \\ 3 & 6 & 5 & 3 & 5 & 3 \\ 3 & 0 & 3 & 0 & 6 & 0 \\ 4 & 3 & 6 & 6 & 4 & 5 \end{pmatrix}.$$

The characteristic polynomial of the matrix W is given by

$$C(X) = X^6 + 3X^5 + 4X^4 + X^3 + 5X^2 + 2 = (X^3 + 5X^2 + 4)^2.$$

Therefore we get $C_1(X) = \sqrt{C(X)} = X^3 + 5X^2 + 4$ and

$$C_1(W) = \begin{pmatrix} 1 & 4 & 2 & 4 & 2 & 2 \\ 2 & 5 & 5 & 3 & 0 & 1 \\ 2 & 0 & 1 & 1 & 4 & 1 \\ 6 & 2 & 2 & 3 & 5 & 2 \\ 4 & 2 & 3 & 1 & 3 & 5 \\ 2 & 3 & 2 & 5 & 3 & 1 \end{pmatrix}.$$

The kernel of this matrix has dimension $o = 3$. A basis of the kernel is given by

$$\begin{aligned} \mathbf{v}_1 &= (1, 0, 0, 4, 2, 0)^T, \\ \mathbf{v}_2 &= (0, 1, 0, 0, 3, 2)^T, \\ \mathbf{v}_3 &= (0, 0, 1, 1, 5, 6)^T. \end{aligned}$$

By using Lemma 5.2, we find that all the vectors \mathbf{v}_1 , \mathbf{v}_2 and \mathbf{v}_3 lie in the desired space $\mathcal{T}^{-1}(\mathcal{O})$.

We extend this basis to a basis of $\text{GF}(7)^6$ and obtain an equivalent linear transformation \tilde{T}^{-1} by

$$\tilde{T}^{-1} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 4 \\ 0 & 1 & 0 & 5 & 3 & 2 \\ 1 & 0 & 0 & 6 & 2 & 0 \end{pmatrix}.$$

Finally, we compute the equivalent central map $\tilde{\mathcal{F}}$ by

$$\begin{aligned} \tilde{F}^{(1)} &= (\tilde{T}^{-1})^T \cdot P^{(1)} \cdot \tilde{T}^{-1} = \begin{pmatrix} 2 & 0 & 2 & 5 & 3 & 2 \\ 0 & 4 & 2 & 3 & 3 & 6 \\ 2 & 2 & 5 & 0 & 1 & 6 \\ 5 & 3 & 0 & 0 & 0 & 0 \\ 3 & 3 & 1 & 0 & 0 & 0 \\ 2 & 6 & 6 & 0 & 0 & 0 \end{pmatrix}, \\ \tilde{F}^{(2)} &= (\tilde{T}^{-1})^T \cdot P^{(2)} \cdot \tilde{T}^{-1} = \begin{pmatrix} 3 & 4 & 5 & 5 & 1 & 3 \\ 4 & 0 & 5 & 2 & 6 & 6 \\ 5 & 5 & 0 & 4 & 3 & 2 \\ 5 & 2 & 4 & 0 & 0 & 0 \\ 1 & 6 & 3 & 0 & 0 & 0 \\ 3 & 6 & 2 & 0 & 0 & 0 \end{pmatrix}, \\ \tilde{F}^{(3)} &= (\tilde{T}^{-1})^T \cdot P^{(3)} \cdot \tilde{T}^{-1} = \begin{pmatrix} 3 & 0 & 5 & 2 & 3 & 6 \\ 0 & 2 & 5 & 1 & 1 & 2 \\ 5 & 5 & 4 & 4 & 0 & 5 \\ 2 & 1 & 4 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 & 0 & 0 \\ 6 & 2 & 5 & 0 & 0 & 0 \end{pmatrix}. \end{aligned}$$

Note that bottom right $o \times o$ submatrices of $\tilde{F}^{(1)}$, $\tilde{F}^{(2)}$ and $\tilde{F}^{(3)}$ contain only zeros. Therefore, we can use these matrices together with \tilde{T} to generate valid signatures for arbitrary messages.

5.2.3 The Case of q Even

For even characteristic we can not write the public polynomials as quadratic forms $\mathbf{x}^T \cdot P^{(i)} \cdot \mathbf{x}$ with symmetric matrices $P^{(i)}$. The reason for this is that, for a symmetric matrix $A = (a_{ij})$ over a field of even characteristic, always $a_{ij} + a_{ji} = 0$ holds, which would correspond to a zero coefficient of $x_i x_j$. We therefore define for each public polynomial $p^{(i)}(x)$ the matrix $P^{(i)} = (p_{jk})^{(i)}$ to be the upper triangular $n \times n$ matrix with

$$p_{jk}^{(i)} = \begin{cases} \text{MonomialCoefficient}(p^{(i)}, x_j x_k) & j \leq k, \\ 0 & \text{otherwise.} \end{cases}$$

and set

$$\bar{Q}^{(i)} = P^{(i)} + (P^{(i)})^T \quad (i = 1, \dots, o).$$

Note that the $\bar{Q}^{(i)}$'s are symmetric matrices with zeros on the main diagonal.

As in the case of odd characteristic, we define two matrices W_1 and W_2 as random linear combinations of the matrices $\bar{Q}^{(i)}$ ($i = 1, \dots, o$). Hereby, we have to ensure that the matrix W_1 is invertible. We compute $W = W_1^{-1} \cdot W_2$ and $C(\lambda)$ to be the characteristic polynomial of the matrix W . As above, let $C_1(\lambda)$ to be the square root of $C(\lambda)$ (provided that $C(\lambda)$ is a real square). However, unlike in the case of an odd q , $C_1(W)$ will be zero in any case. We therefore have to modify the above algorithm.

We factor $C_1(\lambda)$ into irreducible factors and look for a distinctive linear factor $(\lambda - \lambda_1)$ of multiplicity 1. Such a factor should exist with reasonably high probability. The eigenspace of the matrix W according to the eigenvalue λ_1 has dimension exactly two, and one of the eigenvectors must be in the space $\mathcal{T}^{-1}(\mathcal{O})$.

If we denote the two basis vectors of the eigenspace by \mathbf{v}_1 and \mathbf{v}_2 , we therefore know that one of the $q + 1$ vectors $\mathbf{v}_1 + k\mathbf{v}_2$ ($k \in \mathbb{F}$) or \mathbf{v}_2 must be in the desired invariant space. Note that all other vectors in the eigenspace are multiples of these vectors and therefore will span the same image space.

To check which of these $q + 1$ vectors is the correct one, we compute for each vector the space spanned by the vector and its images under linear maps of the form $W_1^{-1} \cdot W_2$. If this space has dimension $> o$, then it can not be the space $\mathcal{T}^{-1}(\mathcal{O})$, which means that we have chosen the wrong vector, so we discard it and try the next one. This process is illustrated by Algorithm 5.2.

Note that the map $\tilde{\mathcal{F}}$ delivered by Algorithm 5.2 is not really a UOV central map. In fact, it might contain oil \times oil terms of the form x_i^2 ($i \in \mathcal{O}$). To cover this fact, we have to adapt the inversion of the central map slightly.

After choosing random values for the vinegar variables and substituting them into the components of the map $\tilde{\mathcal{F}}$, the system (now denoted as $\tilde{\tilde{\mathcal{F}}}$) looks like

$$\begin{aligned} \tilde{\tilde{f}}^{(1)} : \sum_{i \in \mathcal{O}} \alpha_i^{(1)} x_i^2 + \sum_{i \in \mathcal{O}} \beta_i^{(1)} x_i + \gamma^{(1)} &= 0 \\ \tilde{\tilde{f}}^{(2)} : \sum_{i \in \mathcal{O}} \alpha_i^{(2)} x_i^2 + \sum_{i \in \mathcal{O}} \beta_i^{(2)} x_i + \gamma^{(2)} &= 0 \\ &\vdots \\ \tilde{\tilde{f}}^{(o)} : \sum_{i \in \mathcal{O}} \alpha_i^{(o)} x_i^2 + \sum_{i \in \mathcal{O}} \beta_i^{(o)} x_i + \gamma^{(o)} &= 0 \end{aligned}$$

Algorithm 5.2 Attack on balanced oil and vinegar (q even)**Input:** OV public key $\mathcal{P} = (p^{(1)}, \dots, p^{(o)})$ **Output:** Equivalent private key $(\tilde{\mathcal{F}}, \tilde{\mathcal{T}})$

- 1: Define the upper triangular matrices $P^{(i)}$ representing the homogeneous quadratic part of the public key.
- 2: Set, for $i = 1, \dots, o$, $\bar{Q}^{(i)} = P^{(i)} + (P^{(i)})^T$.
- 3: Define two matrices W_1 and W_2 as random linear combinations of the matrices $\bar{Q}^{(i)}$. W_1 must be invertible.
- 4: Compute $W = W_1^{-1} \cdot W_2$.
- 5: Compute the characteristic polynomial $C(\lambda)$ of W . If it is a real square, set $C_1(\lambda) = \sqrt{C(\lambda)}$.
- 6: Factor $C_1(\lambda)$ and if it has a linear factor $(\lambda - \lambda_1)$ of multiplicativity 1, go to the next step. Otherwise, choose other matrices W_1 and W_2 .
- 7: Find a basis $(\mathbf{v}_1, \mathbf{v}_2)$ of the eigenspace of the matrix $C_1(W)$ to the eigenvalue λ_1 . The set of possible eigenvectors is

$$S = \{\mathbf{v}_1 + k\mathbf{v}_2 : k \in \mathbb{F}\} \cup \{\mathbf{v}_2\}.$$

- 8: **for** each $\mathbf{s} \in S$ **do**
- 9: Denote by V_s the linear space spanned by the eigenvector \mathbf{s} . Set $i = 0$.
- 10: **repeat**
- 11: Choose new matrices W_1, W_2 and compute $W = W_1^{-1} \cdot W_2$.
- 12: Compute the image of the space V_s under the action of W .
- 13: Compute a basis of $V_s \cup W(V_s)$. Denote the new space again by V_s .
- 14: **until** $i = 2o - 1$ or $\dim(V_s) > v$.
- 15: **end for**
- 16: If, for some $\mathbf{s} \in S$, $\dim(V_s) = o$ holds, go to the next step; otherwise, choose new matrices W_1 and W_2 and try again.
- 17: Extend the basis of V_s to a basis of \mathbb{F}^n to find an equivalent linear transformation $\tilde{\mathcal{T}}^{-1}$.
- 18: Compute, for $i = 1, \dots, o$, the matrix $\tilde{F}^{(i)}$ representing the i -th component of the equivalent central map by

$$\tilde{F}^{(i)} = (\tilde{\mathcal{T}}^{-1})^T \cdot P^{(i)} \cdot \tilde{\mathcal{T}}^{-1}.$$

Set $\tilde{\mathcal{F}} = (\tilde{F}^{(1)}, \dots, \tilde{F}^{(o)})$.

- 19: **return** $(\tilde{\mathcal{F}}, \tilde{\mathcal{T}})$

To solve this system of equations, we interpret the coefficients $\alpha_i^{(j)}, \beta_i^{(j)}$ and $\gamma^{(j)}$ ($j = 1, \dots, o, i = v + 1, \dots, n$) as well as the variables x_i (which are defined over the field $\mathbb{F}_q = \mathbb{F}_{2^e}$) as degree $e - 1$ polynomials over $\text{GF}(2)$, e.g.

$$x_i = x_{i,0} + x_{i,1}X + x_{i,2}X^2 + \dots + x_{i,e-1}X^{e-1}. \quad (5.3)$$

We compute $x_i^2 = x_i x_i \bmod p$ as a product of polynomials and substitute the polynomial representations of x_i and x_i^2 into the components of $\tilde{\mathcal{F}}$. Here, p is the irreducible polynomial used to generate the field \mathbb{F} as an extension field of $\text{GF}(2)$. By doing so, we get a system of oe linear equations (one for each equation $\tilde{f}^{(i)}$ and every monomial X^j) in the oe variables $x_{i,j}$ ($i \in O, j = 0, \dots, e - 1$), which can

be solved by Gaussian elimination. By solving this system, we get the values of the variables $x_{v+1,0}, x_{v+1,1}, \dots, x_{v+1,e-1}, x_{v+2,0}, \dots, x_{n,e-1}$. In order to translate this solution over $\text{GF}(2)$ into a solution over \mathbb{F} , we compute for every $i = v + 1, \dots, n$

$$x_i = x_{i,0} + x_{i,1}X + x_{i,2}X^2 + \dots + x_{i,e-1}X^{e-1}. \quad (5.4)$$

The solution \mathbf{x} of $\mathcal{F}(\mathbf{x}) = \mathbf{w}$ is given by $\mathbf{x} = (x_1, \dots, x_n)$. Algorithm 5.3 shows this inversion process in algorithmic form.

Algorithm 5.3 Inversion of the map $\tilde{\mathcal{F}}$

Input: set $\mathcal{P} = (p^{(1)}, \dots, p^{(o)})$ of polynomials $\in \mathbb{F}_{2^e}[x_1, \dots, x_n]$ of the form $\tilde{\mathcal{F}}$ (i.e no oil \times oil terms of the form $x_i x_j$ ($i \neq j$))

Output: vector $\mathbf{x} = (x_1, \dots, x_n)$ with $\mathcal{P}(\mathbf{x}) = 0$

- 1: Choose random values in \mathbb{F} for the vinegar variables x_1, \dots, x_v and substitute them into the polynomials $p^{(1)}, \dots, p^{(o)}$.
 - 2: Interpret all coefficients and variables of the resulting quadratic system as degree $e - 1$ polynomials over $\text{GF}(2)$ as shown by (5.3).
 - 3: Solve the resulting linear system of oe equations in oe variables over $\text{GF}(2)$ by Gaussian elimination.
 - 4: Translate the solution over $\text{GF}(2)$ into a solution (x_{v+1}, \dots, x_n) over \mathbb{F} as shown by (5.4).
 - 5: **return** $\mathbf{x} = (x_1, \dots, x_n)$.
-

5.2.4 Toy Example

For our toy example we choose $q = 4$ and $o = v = 3$. The irreducible polynomial used to generate the extension field $\text{GF}(4) = \text{GF}(2^2)$ is $X^2 + X + 1$.

Let the public key of our balanced OV instance be given by $\mathcal{P} = (p^{(1)}, p^{(2)}, p^{(3)})$ with

$$\begin{aligned} p^{(1)}(x_1, \dots, x_6) &= \alpha^2 x_1 x_4 + x_1 x_5 + \alpha^2 x_1 x_6 + x_2^2 + \alpha^2 x_2 x_5 + x_2 x_6 + \alpha^2 x_3 x_4 \\ &\quad + x_3 x_6 + x_4^2 + x_4 x_5 + x_4 x_6 + \alpha^2 x_5^2 + \alpha^2 x_5 x_6 + \alpha x_6^2, \\ p^{(2)}(x_1, \dots, x_6) &= \alpha x_1 x_2 + \alpha^2 x_1 x_5 + \alpha x_1 x_6 + \alpha^2 x_2 x_3 + \alpha x_2 x_4 + x_2 x_5 \\ &\quad + \alpha^2 x_2 x_6 + \alpha^2 x_3^2 + x_4 x_6 + \alpha^2 x_5^2 + \alpha^2 x_5 x_6, \\ p^{(3)}(x_1, \dots, x_6) &= x_1 x_2 + x_1 x_4 + x_2^2 + x_2 x_3 + \alpha x_2 x_5 + x_2 x_6 + \alpha^2 x_3^2 + x_3 x_4 \\ &\quad + x_3 x_6 + x_4^2 + x_4 x_5 + \alpha x_4 x_6 + x_5^2 + \alpha x_5 x_6 + x_6^2. \end{aligned}$$

5.2.4.1 Recovering an Equivalent Private Key

First we have to compute the symmetric matrices $\bar{Q}^{(1)}, \dots, \bar{Q}^{(o)}$ representing the homogeneous quadratic part of the public key. For fields of even characteristic, these matrices are given by

$$\bar{Q}^{(i)} = P^{(i)} + (P^{(i)})^T,$$

where $P^{(i)}$ is the upper triangular matrix containing the coefficients of the i -th component of the public key. We get

$$\bar{Q}^{(1)} = \begin{pmatrix} 0 & 0 & 0 & \alpha^2 & 1 & \alpha^2 \\ 0 & 0 & 0 & 0 & \alpha^2 & 1 \\ 0 & 0 & 0 & \alpha^2 & 0 & 1 \\ \alpha^2 & 0 & \alpha^2 & 0 & 1 & 1 \\ 1 & \alpha^2 & 0 & 1 & 0 & \alpha^2 \\ \alpha^2 & 1 & 1 & 1 & \alpha^2 & 0 \end{pmatrix}, \quad \bar{Q}^{(2)} = \begin{pmatrix} 0 & \alpha & 0 & 0 & \alpha^2 & \alpha \\ \alpha & 0 & \alpha^2 & \alpha & 1 & \alpha^2 \\ 0 & \alpha^2 & 0 & 0 & 0 & 0 \\ 0 & \alpha & 0 & 0 & 0 & 1 \\ \alpha^2 & 1 & 0 & 0 & 0 & \alpha^2 \\ \alpha & \alpha^2 & 0 & 1 & \alpha^2 & 0 \end{pmatrix},$$

$$\bar{Q}^{(3)} = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & \alpha & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & \alpha \\ 0 & \alpha & 0 & 1 & 0 & \alpha \\ 0 & 1 & 1 & \alpha & \alpha & 0 \end{pmatrix}.$$

We get a matrix $W = W_1^{-1} \cdot W_2$ suitable for our purposes by choosing $W_1 = \bar{Q}^{(1)} + \bar{Q}^{(3)}$, $W_2 = \alpha^2 \bar{Q}^{(2)} + \alpha^2 \bar{Q}^{(3)}$. We obtain

$$W = W_1^{-1} \cdot W_2 = \begin{pmatrix} \alpha & \alpha^2 & 0 & \alpha^2 & \alpha^2 & 0 \\ 0 & 1 & \alpha^2 & 1 & 1 & 1 \\ 0 & 0 & \alpha^2 & 1 & \alpha & \alpha \\ 0 & 0 & \alpha & 1 & \alpha^2 & 1 \\ 0 & \alpha^2 & \alpha & \alpha^2 & \alpha^2 & \alpha^2 \\ 0 & 0 & \alpha^2 & 1 & \alpha & \alpha \end{pmatrix}.$$

The characteristic polynomial of the matrix W is

$$C(X) = X^6 + X^4 + X^2 = X^2 - (X + \alpha)^2(X + \alpha^2)^2,$$

Its square root is

$$C_1(X) = \sqrt{C(X)} = X^3 + X^2 + X.$$

The eigenvectors of the Matrix $C_1(W)$ for the eigenvalue 0 are

$$\mathbf{v}_1 = (0, 1, 0, 0, \alpha, 0) \text{ and } \mathbf{v}_2 = (0, 0, 1, 0, 0, 1).$$

We find that \mathbf{v}_2 is the eigenvector suitable for our purposes. A basis of the image space of \mathbf{v}_2 under matrices of the form $W_1^{-1} \cdot W_2$ is given by

$$\mathbf{b}_1 = \mathbf{v}_2 = (0, 0, 1, 0, 0, 1), \mathbf{b}_2 = (0, 1, 0, \alpha, \alpha^2, 0), \mathbf{b}_3 = (1, 0, 0, 0, \alpha^2, 1).$$

By extending this basis to a basis of the whole space \mathbb{F}^n , we get an equivalent linear transformation

$$\tilde{T}^{-1} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & \alpha & 0 \\ 0 & 1 & 0 & 0 & \alpha^2 & \alpha^2 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

Using this matrix \tilde{T}^{-1} , we can compute the matrices $\tilde{F}^{(1)}, \tilde{F}^{(2)}, \tilde{F}^{(3)}$ representing the components of an equivalent central map by

$$\tilde{F}^{(i)} = (\tilde{T}^{-1})^T \cdot P^{(i)} \cdot \tilde{T}^{-1}.$$

We get

$$\tilde{F}^{(1)} = \begin{pmatrix} \alpha & 0 & 0 & \alpha & 0 & \alpha \\ \alpha^2 & \alpha^2 & 0 & \alpha^2 & \alpha & 1 \\ 1 & 1 & 1 & 1 & 1 & \alpha \\ \alpha^2 & 0 & \alpha^2 & \alpha^2 & 1 & \alpha^2 \\ 1 & \alpha^2 & \alpha & 1 & 0 & \alpha^2 \\ \alpha^2 & \alpha^2 & \alpha^2 & \alpha^2 & \alpha^2 & 1 \end{pmatrix}, \quad \tilde{F}^{(2)} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ \alpha^2 & \alpha^2 & 0 & \alpha^2 & \alpha & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & \alpha^2 & 0 & 0 \\ \alpha^2 & \alpha^2 & \alpha & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & \alpha^2 \end{pmatrix},$$

$$\tilde{F}^{(3)} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ \alpha & 1 & 0 & \alpha & \alpha^2 & 1 \\ \alpha & 1 & 1 & \alpha & 1 & 1 \\ 0 & 0 & 1 & \alpha^2 & \alpha & 0 \\ \alpha^2 & \alpha^2 & \alpha & \alpha & 0 & 1 \\ 0 & \alpha^2 & 1 & 0 & 1 & \alpha \end{pmatrix}.$$

The polynomials of the central map are given by

$$\begin{aligned}
\tilde{f}^{(1)}(x_1, \dots, x_6) &= \alpha x_1^2 + \alpha^2 x_1 x_2 + x_1 x_3 + x_1 x_4 + x_1 x_5 + x_1 x_6 + \alpha^2 x_2^2 \\
&\quad + x_2 x_3 + \alpha^2 x_2 x_4 + x_2 x_5 + \alpha x_2 x_6 + x_3^2 + \alpha x_3 x_4 \\
&\quad + \alpha^2 x_3 x_5 + x_3 x_6 + \alpha^2 x_4^2 + x_6^2, \\
\tilde{f}^{(2)}(x_1, \dots, x_6) &= \alpha^2 x_1 x_2 + x_1 x_3 + \alpha^2 x_1 x_5 + \alpha^2 x_2^2 + \alpha^2 x_2 x_4 + x_2 x_5 \\
&\quad + x_3 x_4 + \alpha x_3 x_5 + x_3 x_6 + \alpha^2 x_4^2 + x_5^2 + \alpha^2 x_6^2, \\
\tilde{f}^{(3)}(x_1, \dots, x_6) &= x_1^2 + \alpha x_1 x_2 + \alpha x_1 x_3 + x_1 x_4 + \alpha^2 x_1 x_5 + x_1 x_6 + x_2^2 \\
&\quad + x_2 x_3 + \alpha x_2 x_4 + \alpha x_2 x_6 + x_3^2 + \alpha^2 x_3 x_4 + \alpha^2 x_3 x_5 + \alpha^2 x_4^2 + \alpha x_6^2.
\end{aligned}$$

Note that the polynomials $\tilde{f}^{(1)}, \dots, \tilde{f}^{(3)}$, contain, besides the terms usually appearing in an OV central map, also terms of the form x_i^2 in the Oil variables.

5.2.4.2 Forging a Signature

We want to generate a signature for the message $\mathbf{w} = (1, 1, \alpha) \in \mathbb{F}^3$.

We choose the vinegar variables $(x_1, x_2, x_3) = (1, \alpha^2, 1)$ and substitute them into the polynomials $\tilde{f}^{(1)}, \tilde{f}^{(2)}, \tilde{f}^{(3)}$. We get

$$\begin{aligned}
\tilde{f}^{(1)}(x_4, x_5, x_6) &= \alpha^2 x_4^2 + x_4 + x_5 + x_6^2 + x_6 + \alpha, \\
\tilde{f}^{(2)}(x_4, x_5, x_6) &= \alpha^2 x_4^2 + \alpha^2 x_4 + x_5^2 + \alpha x_5 + \alpha^2 x_6^2 + x_6 + \alpha, \\
\tilde{f}^{(3)}(x_4, x_5, x_6) &= \alpha^2 x_4^2 + \alpha^2 x_4 + \alpha x_6^2 + \alpha.
\end{aligned}$$

We interpret the variables x_4, x_5, x_6 over $\text{GF}(4)$ as degree 1 polynomials over $\text{GF}(2)$, i.e.

$$\begin{aligned}
x_4 &= x_{4,1} + x_{4,2}X, \\
x_5 &= x_{5,1} + x_{5,2}X, \\
x_6 &= x_{6,1} + x_{6,2}X.
\end{aligned}$$

and compute $x_i^2 = (x_{i,1} + x_{i,2}X)^2 \bmod (X^2 + X + 1)$. We get

$$\begin{aligned}
x_4^2 &= x_{4,2}X + x_{4,1} + x_{4,2}, \\
x_5^2 &= x_{5,2}X + x_{5,1} + x_{5,2}, \\
x_6^2 &= x_{6,2}X + x_{6,1} + x_{6,2}.
\end{aligned}$$

Here, we made use of the field equations $x_{i,j}^2 = x_{i,j}$ over $\text{GF}(2)$. Furthermore, we interpret all coefficients in the polynomials $\tilde{f}^{(i)}$ as degree 1 polynomials over $\text{GF}(2)$. Therefore, we get

$$\begin{aligned}\hat{f}^{(1)} &= (x_{4,1} + x_{5,2} + 1)X + x_{5,1} + x_{6,2}, \\ \hat{f}^{(2)} &= (x_{4,2} + x_{5,1} + x_{6,1} + 1)X + x_{4,2} + x_{5,1}, \\ \hat{f}^{(3)} &= (x_{4,2} + x_{6,1} + 1)X + x_{4,2} + x_{6,2}.\end{aligned}$$

We consider the coefficients of X^1 and $X^0 = 1$ separately and therefore get a system $\hat{\mathcal{F}}$ of linear equations over $\text{GF}(2)$.

$$\hat{f}^{(1,1)} = x_{5,1} + x_{6,2}, \quad (5.5)$$

$$\hat{f}^{(1,2)} = x_{4,1} + x_{5,2} + 1, \quad (5.6)$$

$$\hat{f}^{(2,1)} = x_{4,2} + x_{5,1}, \quad (5.7)$$

$$\hat{f}^{(2,2)} = x_{4,2} + x_{5,1} + x_{6,1} + 1, \quad (5.8)$$

$$\hat{f}^{(3,1)} = x_{4,2} + x_{6,2}, \quad (5.9)$$

$$\hat{f}^{(3,2)} = x_{4,2} + x_{6,1} + 1. \quad (5.10)$$

Until here, our computations only depended on the central map $\tilde{\mathcal{F}}$ and the choice of the vinegar variables and not on the message to be signed. So, if we want to forge signatures for multiple messages, we can precompute the system (5.10) once and have to perform only the remaining steps for each single message.

Now, we transform the message $\mathbf{w} = (1, 1, \alpha) \in \mathbb{F}^3$ into a message $\hat{\mathbf{w}} = (1, 0, 1, 0, 0, 1) \in \text{GF}(2)^6$ and solve the linear system $\hat{\mathcal{F}}(\hat{\mathbf{y}}) = \hat{\mathbf{w}}$. We get

$$\hat{\mathbf{y}} = (1, 0, 1, 0, 0, 0),$$

and transform $\hat{\mathbf{y}}$ back into a vector $\mathbf{y}' = (1, 1, 0) \in \mathbb{F}^3$.

We attach the vinegar variables, obtaining $\mathbf{y} = (1, \alpha^2, 1, 1, 1, 0) \in \mathbb{F}^6$ and finally compute $\mathbf{z} = \mathcal{T}^{-1}(\mathbf{y})$ to get the signature

$$\mathbf{z} = (0, 1, 1, \alpha^2, 0, 0) \in \mathbb{F}^6.$$

Using the public key \mathcal{P} , we can easily verify that \mathbf{z} is a valid signature for the message $(1, 1, \alpha)$.

5.2.5 The Unbalanced Oil and Vinegar Signature Scheme (UOV)

After the balanced Oil and Vinegar signature scheme had been broken by the OV attack described above, Kipnis, Patarin and Goubin proposed in [9] a modified scheme called Unbalanced Oil and Vinegar signature scheme (UOV).

The scheme works exactly as the balanced scheme, but chooses $v > o$. We now consider the question how this choice affects the complexity of the above attack.

For $v \gtrsim o$, the attack works essentially the same as described above, only the spaces \mathcal{O} and \mathcal{V} do not have the same dimension any longer. We therefore have to modify Lemma 5.3 as follows.

Let $E : \mathbb{F}^n \rightarrow \mathbb{F}^n$ be a linear transformation of the form

$$E = \begin{pmatrix} E_1 & E_2 \\ E_3 & 0_{o \times o} \end{pmatrix}, \quad (5.11)$$

where E_1 is a $v \times v$ matrix, E_2 is a $v \times o$ matrix and E_3 is an $o \times v$ matrix with entries randomly chosen from \mathbb{F} . Then we have

Lemma 5.6

1. $E(\mathcal{O})$ is an o -dimensional proper subspace of \mathcal{V} .
2. If E is invertible, $E^{-1}(\mathcal{V})$ is a v -dimensional subspace of \mathbb{F}^n , in which \mathcal{O} is a proper subspace.

Proof Analogous to the proof of Lemma 5.3. □

As in the attack on balanced Oil and Vinegar, we look for the space $\mathcal{T}^{-1}(\mathcal{O})$, which we will denote by $\tilde{\mathcal{O}}$. To find this space, we will again use the matrices $P^{(i)}$ corresponding to the components of the public key. Note again that we have

$$P^{(i)} = T^T \cdot F^{(i)} \cdot T,$$

where the matrices $F^{(i)}$ are of the form (5.11). We denote by Ω the span of the matrices $W_1^{-1} \cdot W_2$, where W_1 (invertible) and W_2 are random linear combinations of the matrices $\tilde{Q}^{(i)}$ ($i = 1, \dots, o$). We are looking for a common invariant subspace of the elements of Ω . The following lemma states that such a space exists with high probability.

Lemma 5.7 *Let $J : \mathbb{F}^n \rightarrow \mathbb{F}^n$ be a randomly chosen invertible \mathbb{F} -linear map such that*

1. *There exist two subspaces A, B in \mathbb{F}^n such that the dimension of A is v , and the dimension of B is o and $B \subset A$;*
2. *$J(B) \subset A$.*

Then the probability that J has a nontrivial invariant subspace in B is not less than q^{o-v} .

Proof We restrict to invariant subspaces of dimension 1. Therefore, a vector $\mathbf{v} \in \mathbb{F}^n$ lies in the invariant subspace, if J maps it to a multiple of itself. The probability that a nonzero vector $\mathbf{v} \in B$ is mapped to a nonzero multiple of itself is $\frac{q-1}{q^v-1}$ (since we know that $J(B) \subset A$). To get the expected number of such vectors, we multiply by the number of nonzero vectors in B to get $\frac{(q-1)(q^o-1)}{q^v-1}$. Since, if a vector is mapped to a multiple of itself, then the same is also true for all multiples of this vector, the probability that there exists an invariant subspace of dimension 1 is roughly $\frac{q^o-1}{q^v-1} \approx q^{o-v}$. \square

Theorem 5.8 Let W_1 and W_2 be randomly chosen linear combinations of the matrices $P^{(i)}$ ($i = 1, \dots, o$) and let W_1 be invertible. Then the probability that the matrix $W_1^{-1} \cdot W_2$ has a nontrivial invariant subspace (which is also a subspace of $\mathcal{T}^{-1}(O)$) is roughly q^{o-v} .

Proof This follows directly from the above Lemma. \square

As for the balanced case we can, by computing the minimal invariant subspaces of the matrices $W_1^{-1} \cdot W_2$ and using Lemma 5.6 to check whether they are subspaces of \mathcal{T}^{-1} , recover the essential parts of the UOV linear transformation \mathcal{T} . From this, we can then compute an equivalent UOV private key $(\tilde{\mathcal{F}}, \tilde{\mathcal{T}})$ which can be used to sign messages.

The complexity of the whole process can be estimated by

$$\text{complexity}_{\text{UOV attack}}(q, o, v) = q^{v-o-1} o^4. \quad (5.12)$$

Equation (5.12) seems to indicate that a bigger v provides more security. However, for larger values of v ($v > 2o$), the scheme can be attacked by direct attacks more easily (see next section). When choosing $v \approx \frac{o^2}{2}$, the complexity of a direct attack against the scheme even becomes polynomial (see Sect. 8.7).

Another point we have to consider when choosing the parameters of UOV is the efficiency of the scheme. The public key size of UOV increases quadratically with the number of vinegar variables, while the signature size increases linearly. Therefore, for large values of v , the key sizes are very big. A good compromise between security and efficiency seems to be choosing $v = 2o$. In such a UOV instance, the number of variables is three times as large as the number of equations.

A possibility to get an even more efficient signature scheme on the basis of UOV is provided by the Rainbow signature scheme, which can be seen as a multi-layer version of UOV (see Sect. 5.4).

5.3 Other Attacks on UOV

Besides the UOV attack presented in the last section, there exist two additional, important attacks against the UOV signature scheme. These are

- The direct attack and
- The UOV Reconciliation attack.

5.3.1 The Direct Attack

In a direct attack, the adversary considers the public equation $\mathcal{P}(\mathbf{z}) = \mathbf{w}$ as an instance of the MQ Problem and tries to solve it using an algorithm like XL or a Gröbner basis method such as F_4 or F_5 (see Chap. 8). In the case of UOV, we have to consider that the UOV public key contains usually three times as many variables as equations.

We therefore have to consider a technique which allows to solve a multivariate quadratic system of m equations in vm variables in the same time as a determined system of $m - \lfloor v \rfloor + 1$ equations (see Sect. 8.7.5). In some cases one can get an additional speed up by guessing some variables in the modified system (Hybrid Approach). Though one has to run the algorithm more often (approximately q^k times when guessing k variables over a field with q elements), the overall complexity of the attack might be smaller.

Experiments have shown that the multivariate quadratic systems given by UOV public keys behave very much like random systems. In particular, the degree of regularity of a system derived from UOV is the same as that of a random system of the same size. It is therefore relatively easy to estimate the complexity of a direct attack against the UOV signature scheme.

The UOV Reconciliation Attack

For the simplicity of our description, we restrict the maps \mathcal{T} and \mathcal{F} to be homogeneous in this section. Note that, in this case, the public key will be a homogeneous quadratic map, too. The attack can be extended to the general case in a straightforward way.

We denote the $n \times n$ matrix representing the quadratic form of the i -th component of the central map by $F^{(i)}$ and the matrix representing the linear transformation \mathcal{T} by T . With this notation, we can compute the i -th component of the public key by

$$P^{(i)} = T^T \cdot F^{(i)} \cdot T. \quad (5.13)$$

5.3.1.1 Equivalent Keys

Let \mathcal{P} be a UOV public key. The goal of the UOV Reconciliation attack is to find a UOV private key $(\mathcal{F}, \mathcal{T})$ with $\mathcal{F} \circ \mathcal{T} = \mathcal{P}$. Hereby, the attack looks for private keys of a very special form.

Theorem 5.9 *Let \mathcal{P} be a UOV public key. Then, with overwhelming probability, there exists a UOV private key $(\tilde{\mathcal{F}}, \tilde{\mathcal{T}})$ with $\tilde{\mathcal{F}} \circ \tilde{\mathcal{T}} = \mathcal{P}$, such that $\tilde{\mathcal{T}}$ has the form*

$$\tilde{\mathcal{T}} = \begin{pmatrix} 1_{v \times v} & T'_{o \times v} \\ 0_{v \times o} & 1_{o \times o} \end{pmatrix} \quad (5.14)$$

Note that $\tilde{\mathcal{T}}^{-1}$ has the same form as $\tilde{\mathcal{T}}$.

In order to prove this theorem, we first need a lemma.

Lemma 5.10 *Let F be the $n \times n$ matrix representing the quadratic form of a UOV polynomial, i.e*

$$F = \begin{pmatrix} F_{v \times v}^{(1)} & F_{v \times o}^{(2)} \\ F_{o \times v}^{(3)} & 0_{o \times o} \end{pmatrix},$$

and $\Omega \in \mathbb{F}^{n \times n}$ be a matrix of the form

$$\Omega = \begin{pmatrix} \Omega_{v \times v}^{(1)} & 0_{v \times o} \\ \Omega_{o \times v}^{(3)} & \Omega_{o \times o}^{(4)} \end{pmatrix}.$$

Then, $F \cdot \Omega$ has the form of a UOV central polynomial.

Proof We have

$$\begin{aligned} F \cdot \Omega &= \begin{pmatrix} F^{(1)} & F^{(2)} \\ F^{(3)} & 0 \end{pmatrix} \cdot \begin{pmatrix} \Omega^{(1)} & 0 \\ \Omega^{(3)} & \Omega^{(4)} \end{pmatrix} \\ &= \begin{pmatrix} F^{(1)} \cdot \Omega^{(1)} + F^{(2)} \cdot \Omega^{(3)} & F^{(2)} \cdot \Omega^{(4)} \\ F^{(3)} \cdot \Omega^{(1)} & 0 \end{pmatrix}, \end{aligned}$$

which has the form of a UOV central polynomial. \square

Note that Ω^{-1} has the same form as Ω . Since $\mathcal{F} \circ \Omega \circ \Omega^{-1} \circ \mathcal{T} = \mathcal{P}$, we can see that $(F \circ \Omega, \Omega^{-1} \circ T)$ is another UOV private key for the public key \mathcal{P} (a so called **equivalent key**).

With this, we can now prove Theorem 5.9.

Proof Since \mathcal{P} is a valid UOV public key, we know that there exists a UOV private key $(\mathcal{F}, \mathcal{T})$ such that $\mathcal{F} \circ \mathcal{T} = \mathcal{P}$. In the following we show that there exists an equivalent private key $(\tilde{\mathcal{F}}, \tilde{\mathcal{T}})$ such that $\tilde{\mathcal{T}}$ is of the form shown in the theorem. To do this, we compute Ω in such a way that $\Omega \cdot \tilde{\mathcal{T}} = \mathcal{T}$. Let $\Omega = \begin{pmatrix} \tilde{\Omega}^{(1)} & 0 \\ \tilde{\Omega}^{(3)} & \tilde{\Omega}^{(4)} \end{pmatrix}$ and

$T = \begin{pmatrix} T^{(1)} & T^{(2)} \\ T^{(3)} & T^{(4)} \end{pmatrix}$. If $T^{(1)}$ is invertible,³ we get

- $\tilde{\Omega}^{(1)} = T^{(1)}$
- $\tilde{\Omega}^{(3)} = T^{(3)}$ and
- $\tilde{\Omega}^{(4)} = T^{(4)} - T^{(3)} \cdot (T^{(1)})^{-1} \cdot T^{(2)}$.

The matrix T' is given as $T' = (T^{(1)})^{-1} \cdot T^{(2)}$. □

The matrix $\tilde{\mathcal{T}}$ can be written as a product of matrices $T_{v+1} \cdot \dots \cdot T_n$ with

$$T_i = \begin{pmatrix} 1 & 0 & 0 & t'_{1i} & 0 \\ & \ddots & \vdots & \vdots & \vdots \\ 0 & 1 & 0 & t'_{vi} & 0 \\ 0 & \dots & 0 & 1 & 0 & 0 \\ \vdots & & \vdots & & \ddots & \\ 0 & \dots & 0 & 0 & & 1 \end{pmatrix} \quad (i = v+1, \dots, n), \quad (5.15)$$

i.e. the only non zero elements not on the main diagonal are the first v elements in the i -th column. The matrices T_i contain the same non zero elements as the matrix T' of (5.14). By inversion we get $\tilde{T}^{-1} = T_n^{-1} \cdot \dots \cdot T_{v+1}^{-1}$. Note that each T_i^{-1} has the form (5.15) ($\forall i = v+1, \dots, n$).

5.3.1.2 The Attack

The UOV-Reconciliation attack [6] is based on the following observation: Let $((\mathcal{F}, \mathcal{T}), \mathcal{P})$ be a UOV key pair such that T has the form (5.14). We write T^{-1} as a product of matrices $T_n \cdot \dots \cdot T_{v+1}$ with matrices T_j having the form of (5.15). Note that each of these matrices contains, besides the 1's on the main diagonal, only v non-zero elements. With this notation, (5.13) yields

³If $T^{(1)}$ is not invertible, we can switch rows and columns of T by renumbering the variables until we get an invertible matrix.

$$F^{(k)} = \underbrace{T_{v+1}^T \cdots T_{n-1}^T \cdot \overbrace{T_n^T \cdot P^{(k)} \cdot T_n \cdot T_{n-1} \cdots T_{v+1}}^{P_{n-2}^{(k)}}}_{P_v^{(k)}} \quad (k = 1, \dots, o) \quad (5.16)$$

with matrices $P_j^{(k)}$ of the form, which we claim to be

$$P_j^{(k)} = \begin{pmatrix} \star_{j \times j} & \star_{j \times (n-j)} \\ \star_{(n-j) \times j} & 0_{(n-j) \times (n-j)} \end{pmatrix} \quad (j = v, \dots, n-1, k = 1, \dots, o). \quad (5.17)$$

The matrices $P_v^{(k)}$ ($k = 1, \dots, o$) have the form of a UOV central map.

The goal of the attack is to compute, starting with $P_n^{(k)} = P^{(k)}$, for each ($k = 1, \dots, o$) a sequence of matrices $P_j^{(k)}$ ($j = n-1, \dots, v$) of the form (5.17). At the end of this process, we will get matrices $P_v^{(k)}$ ($k = 1, \dots, o$), which, together with the matrix $T^{-1} = T_n \cdots T_{v+1}$ can be used as an equivalent private key.

To do this, we have to take a closer look at the question, how we get from $P_{j+1}^{(k)}$ to $P_j^{(k)}$ ($j = n-1, \dots, v$). We have

$$\underbrace{\begin{pmatrix} \star & \dots & \star & \dots & \dots & \star \\ \vdots & & \vdots & & & \vdots \\ \star & \dots & \star & \dots & \dots & \star \\ \star & \dots & \star & 0_{j,j} & \dots & 0_{j,n} \\ \vdots & & \vdots & \vdots & 0 & \dots & 0 \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ \star & \dots & \star & 0_{n,j} & 0 & \dots & 0 \end{pmatrix}}_{P_j^{(k)}} = T_{j+1}^T \cdot \underbrace{\begin{pmatrix} a_{11} & \dots & a_{1,j-1} & a_{1,j} & \dots & \dots & a_{1n} \\ \vdots & & \vdots & \vdots & & & \vdots \\ a_{j-1,1} & \dots & a_{j-1,j-1} & a_{j-1,j} & \dots & \dots & a_{j-1,n} \\ a_{j,1} & \dots & a_{j,j-1} & a_{j,j} & \dots & \dots & a_{j,n} \\ \vdots & & \vdots & \vdots & 0 & \dots & 0 \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ a_{n1} & \dots & a_{n,j-1} & a_{n,j} & 0 & \dots & 0 \end{pmatrix}}_{P_{j+1}^{(k)}} \cdot T_{j+1}. \quad (5.18)$$

Since the elements of $P_{j+1}^{(k)}$ are known, the elements of the matrix $P_j^{(k)}$ are given as quadratic functions in the unknown elements of the matrix T_{j+1} . Most of the elements of the matrix $P_j^{(k)}$ are unknown, but we know that the elements in the bottom right corner must be zero. On the other hand the zero elements in the bottom right $(n-j) \times (n-j)$ matrix of $P_j^{(k)}$ do not help us (they are automatically zero due to the design of T_{j+1}).

On the other hand, each of the zero elements in the j -th row/column yields one quadratic equation in the v unknown elements of T_{j+1} . Since the equations delivered by $0_{k,l}$ and $0_{l,k}$ are the same, (5.18) yields $n-j$ quadratic equations. Altogether we get $o(n-j)$ quadratic equations in v variables (for $k = 1, \dots, o$). By solving this

system, we can compute the elements of matrix T_{j+1} and can use (5.18) to compute $P_j^{(k)}$ ($k = 1, \dots, o$).

By repeating this process we can, starting with the matrices $P_n^{(k)} = P^{(k)}$, find o matrices $P_v^{(k)}$ ($k = 1, \dots, o$) which, together with the matrix $T^{(-1)} = T_n \cdot \dots \cdot T_{v+1}$, can be used as an alternative private key. An attacker can use this equivalent private key to generate signatures in the same way as a legitimate user.

During the attack we have to solve systems of $(n - j)o$ quadratic equations in v variables ($j = n - 1, \dots, v$). The complexity of the attack is mainly given by the complexity of solving the first system of o quadratic equations in v variables.

Algorithm 5.4 shows the UOV-Reconciliation attack in a compact form.

Algorithm 5.4 UOV-reconciliation attack

Input: matrices $P_n^{(k)}$ ($k = 1, \dots, o$) representing the homogeneous quadratic parts of the public polynomials

Output: private key (represented by matrices $F^{(k)}$ ($k = 1, \dots, o$) and a matrix T)

- 1: **for** $j = n - 1$ to v **do**
 - 2: Define a matrix T_{j+1} of the form (5.15).
 - 3: Define for $k = 1, \dots, o$ matrices $P_j^{(k)}$ of the form (5.17).
 - 4: Compute for $k = 1, \dots, o$ the matrix $U^{(k)} = T_{j+1}^T \cdot P_{j+1}^{(k)} \cdot T_{j+1}$. The equality of $P_j^{(k)}$ and $U^{(k)}$ (see Eq. (5.18)) yields, for every $k = 1, \dots, o$, $n - j$ quadratic equations in the elements of T_{j+1} . Altogether we get therefore a system of $o(n - j)$ equations in v variables.
 - 5: Solve the quadratic system generated in the previous step by any method such as XL or F_4/F_5 and put the solution into the matrix T_{j+1} .
 - 6: Compute, for $k = 1, \dots, o$, the matrices $P_j^{(k)}$ by

$$P_j^{(k)} = T_{j+1}^T \cdot P_{j+1}^{(k)} \cdot T_{j+1}.$$
 - 7: **end for**
 - 8: $M_T \leftarrow T_n \cdot \dots \cdot T_{v+1}$
 - 9: $F^{(k)} \leftarrow P_v^{(k)}$ ($k = 1, \dots, o$)
 - 10: **return** $F^{(k)}$ ($k = 1, \dots, o$), M_T
-

5.3.2 Practical Parameters

Table 5.1 shows parameter recommendations for the UOV signature scheme for our security categories I, II and III. The given parameters are chosen in a way that the resulting schemes provide the claimed level of security against the direct and the UOV Reconciliation attack, as well as the UOV Attack discussed in the previous section.

Table 5.1 Parameters and key sizes of UOV

Security category	Parameters (\mathbb{F} , v , o)	Public key size (kB)	Private key size (kB)	Signature size (bit)
I	(GF(256), 96, 48)	496.2	441.0	1152
II	(GF(256), 144, 72)	1663.1	1478.3	1728
III	(GF(256), 192, 96)	3982.6	3492.1	2304

5.4 The Rainbow Signature Scheme

The Rainbow signature scheme as proposed by Ding and Schmidt in [5] can be seen as a multilayer version of UOV. By combining several UOV layers into one scheme, it is possible to reduce key and signature sizes as well as to improve the performance of the scheme. The Rainbow signature scheme can be described as follows.

Key Generation Let \mathbb{F} be a finite field with q elements. Let v_1, \dots, v_{u+1} be integers such that $0 < v_1 < v_2 < \dots < v_u < v_{u+1} = n$ and define the sets of integers $V_i = \{1, \dots, v_i\}$ for $i = 1, \dots, u$. We set $o_i = v_{i+1} - v_i$ and $O_i = \{v_i + 1, \dots, v_{i+1}\}$ ($i = 1, \dots, u$). The number of elements in V_i is v_i and we have $|O_i| = o_i$.

The central map \mathcal{F} of the scheme consists of $m = n - v_1$ polynomials $f^{(v_1+1)}, \dots, f^{(n)} \in \mathbb{F}[x_1, \dots, x_n]$ of the form

$$f^{(k)}(\mathbf{x}) = \sum_{i,j \in V_\ell, i \leq j} \alpha_{ij}^{(k)} x_i x_j + \sum_{i \in O_\ell, j \in V_\ell} \beta_{ij}^{(k)} x_i x_j + \sum_{i \in V_\ell \cup O_\ell} \gamma_i^{(k)} x_i + \eta^{(k)} \quad (k = v_1 + 1, \dots, n), \quad (5.19)$$

where ℓ is the only integer such that $k \in O_\ell$.

The *central map* \mathcal{F} as defined above consists of u different levels of Oil and Vinegar. In the ℓ -th level, the variables $x_i \in V_\ell$ are the vinegar variables and $x_j \in O_\ell$ are the Oil variables. So, the polynomials of the ℓ -th level form a UOV scheme with v_ℓ vinegar variables and o_ℓ Oil variables. For $u = 1$ we get exactly the UOV signature scheme of Sect. 5.2.5.

The different levels of Oil and Vinegar in the map \mathcal{F} are called *Rainbow layers*.

The special structure of the Rainbow central map \mathcal{F} is illustrated by Fig. 5.1. Let $\hat{f}^{(i)}$ be the homogeneous quadratic part of the i -th component of the central map ($i = v_1 + 1, \dots, n$). Then, $\hat{f}^{(i)}$ can be written as a quadratic form, i.e.

$$\hat{f}^{(i)}(\mathbf{x}) = \mathbf{x}^T \cdot F^{(i)} \cdot \mathbf{x}.$$

Figure 5.1 shows the structure of the matrices $F^{(i)}$ ($i = 1, \dots, o$). The white parts of the matrices contain only zero values, while the gray parts can contain arbitrary field elements. As one can see, the matrices $F^{(v_1+1)}, \dots, F^{(v_2)}$,

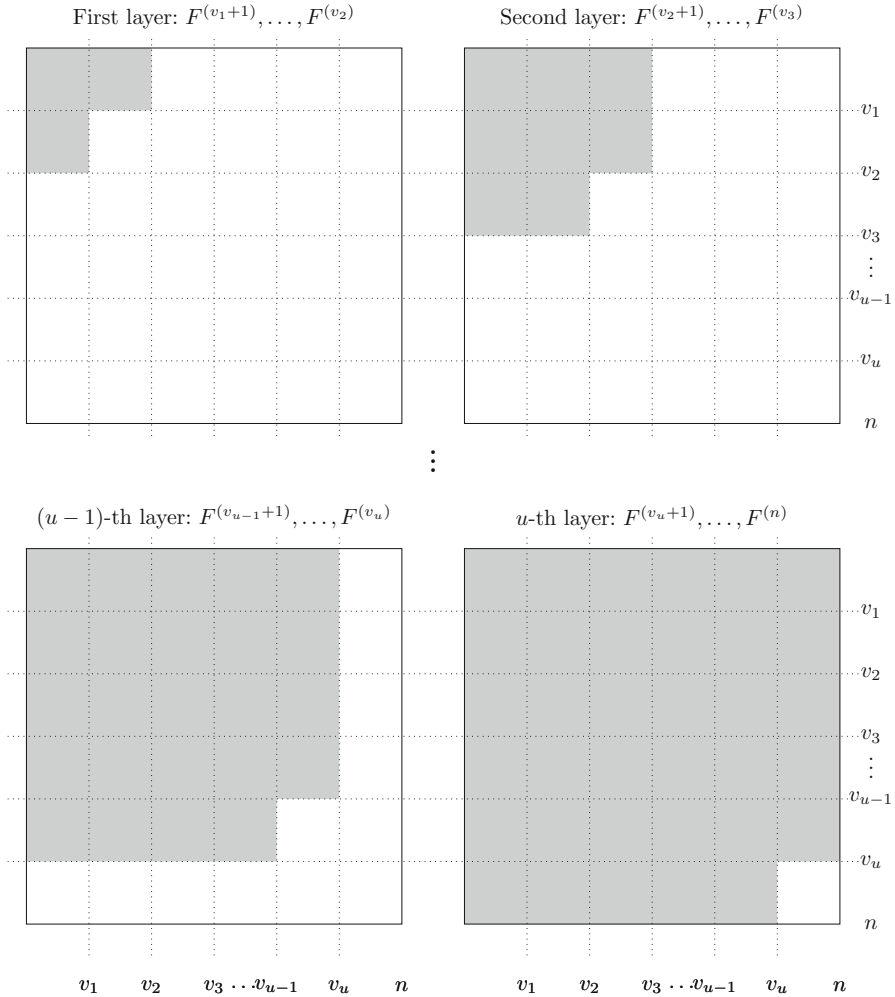


Fig. 5.1 Quadratic forms of the rainbow central map

corresponding to the central polynomials of the first Rainbow layer, contain non zero elements only in a small space in the top left corner, while, for higher layers, this space increases. Note that, for all $i = v_1 + 1, \dots, n$, the bottom right $o_u \times o_u$ submatrices of $F^{(i)}$ contains zero values. Furthermore, note that all of the matrices $F^{(i)}$ ($i = v_1 + 1, \dots, n$) have the form of Oil and Vinegar matrices (see (5.1)).

The map $\mathcal{F}(\mathbf{x}) = (f^{(v_1+1)}(\mathbf{x}), \dots, f^{(n)}(\mathbf{x}))$ can be inverted as follows. First, we choose the values of the variables x_1, \dots, x_{v_1} at random and substitute them into the polynomials $f^{(v_1+1)}, \dots, f^{(n)}$. By doing so, we get a system of o_1 linear equations (given by the polynomials $f^{(k)}$ ($k \in O_1$)) in the o_1 unknowns $x_{v_1+1}, \dots, x_{v_2}$, which can be solved by e.g. Gaussian elimination. The so computed values of the variables

x_i ($i \in O_1$) are substituted into the polynomials $f^{(k)}$ ($k > v_2$) and a system of o_2 linear equations (given by the polynomials $f^{(k)}$ ($k \in O_2$)) in the o_2 unknowns x_i ($i \in O_2$) is obtained. By repeating this process we can find the values of all the variables x_i ($i = 1, \dots, n$).⁴

To hide the structure of the central map \mathcal{F} in the public key, one composes it with two affine invertible maps $\mathcal{S} : \mathbb{F}^m \rightarrow \mathbb{F}^m$ and $\mathcal{T} : \mathbb{F}^n \rightarrow \mathbb{F}^n$.

The *public key* of the scheme is therefore given as $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$. The *private key* consists of the three maps \mathcal{S} , \mathcal{F} and \mathcal{T} and therefore allows to invert the public key.

The process of signature generation/verification can be described as follows:

Signature Generation In order to sign a document d , we use a hash function $\mathcal{H} : \{0, 1\}^* \rightarrow \mathbb{F}^m$ to compute the hash value $\mathbf{w} = \mathcal{H}(d) \in \mathbb{F}^m$ of the document. Then we compute recursively $\mathbf{x} = \mathcal{S}^{-1}(\mathbf{w})$, $\mathbf{y} = \mathcal{F}^{-1}(\mathbf{x})$ and $\mathbf{z} = \mathcal{T}^{-1}(\mathbf{y})$. The signature of the document is $\mathbf{z} \in \mathbb{F}^n$. Here, $\mathcal{F}^{-1}(\mathbf{x})$ means finding one (of approximately q^{v_1}) pre-image of \mathbf{x} under the central map \mathcal{F} (see above).

Signature Verification To check the authenticity of a signature $\mathbf{z} \in \mathbb{F}^n$, one simply computes $\mathbf{w}' = \mathcal{P}(\mathbf{z})$ and the hash value $\mathbf{w} = \mathcal{H}(d)$ of the document. If $\mathbf{w}' = \mathbf{w}$ holds, the signature is accepted, otherwise it is rejected.

5.4.1 Key Sizes and Efficiency

The size of the public key is

$$\text{size}_{\text{pk Rainbow}} = m \frac{(n+1)(n+2)}{2} \quad (5.20)$$

field elements, the size of the private key is

$$\text{size}_{\text{sk Rainbow}} = \underbrace{m(m+1)}_{\text{map } \mathcal{S}} + \underbrace{n(n+1)}_{\text{map } \mathcal{T}} + \underbrace{\sum_{i=1}^u o_i \left(\frac{v_i(v_i+1)}{2} + v_i o_i + v_{i+1} + 1 \right)}_{\text{map } \mathcal{F}} \quad (5.21)$$

field elements.

Similar to the case of Oil and Vinegar, the signature generation requires only the solution of linear systems and therefore can be implemented very efficiently. Furthermore, the size of the linear systems to be solved during the signature generation of Rainbow are smaller than in the UOV case, which leads to an additional speed up though one has to solve several linear systems.

⁴It may happen, that one of the linear systems does not have a solution. If so, one has to choose other values of x_1, \dots, x_{v_1} and try again. However, the probability of this is very small. Therefore, in most cases, one gets a pre-image \mathbf{x} at the first try.

5.4.2 Toy Example

In the following we demonstrate the workflow of the Rainbow signature scheme using a toy example. We choose $\mathbb{F} = \text{GF}(4)$ and the Rainbow parameters $(v_1, o_1, o_2) = (3, 2, 2)$. We therefore have 4 equations in 7 variables.

The private key consists of the two affine maps $\mathcal{S} : \mathbb{F}^4 \rightarrow \mathbb{F}^4$,

$$\mathcal{S}(x_1, \dots, x_4) = \begin{pmatrix} 1 & 0 & 1 & \alpha \\ 0 & 0 & \alpha & \alpha \\ \alpha & \alpha & 1 & \alpha \\ \alpha & \alpha & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} \alpha \\ 1 \\ 0 \\ \alpha \end{pmatrix}$$

and $\mathcal{T} : \mathbb{F}^7 \rightarrow \mathbb{F}^7$,

$$\mathcal{T}(x_1, \dots, x_7) = \begin{pmatrix} \alpha^2 & 1 & \alpha & \alpha^2 & 0 & 1 & \alpha \\ \alpha^2 & 1 & \alpha & \alpha & 1 & \alpha^2 & 0 \\ 0 & 0 & \alpha^2 & 1 & 0 & \alpha^2 & \alpha^2 \\ \alpha & 0 & \alpha & \alpha^2 & \alpha^2 & \alpha^2 & 0 \\ 0 & 0 & \alpha & \alpha^2 & 1 & \alpha^2 & 0 \\ \alpha^2 & \alpha & 1 & 0 & \alpha^2 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & \alpha & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \\ \alpha^2 \\ \alpha \\ \alpha^2 \\ \alpha^2 \end{pmatrix}.$$

and the central map $\mathcal{F} : \mathbb{F}^7 \rightarrow \mathcal{F}^4$ given by

$$\begin{aligned} f^{(4)} &= \alpha y_1^2 + y_1 y_4 + y_1 y_5 + \alpha^2 y_1 + \alpha y_2^2 + y_2 y_3 + \alpha^2 y_2 y_4 + \alpha y_2 + \alpha^2 y_3^2 \\ &\quad + \alpha y_3 y_4 + \alpha^2 y_3 y_5 + \alpha y_3 + \alpha y_4 + \alpha^2 y_5 + \alpha^2, \\ f^{(5)} &= x_1^2 + x_1 x_3 + x_1 x_4 + x_2 x_3 + \alpha x_2 x_4 + \alpha x_2 x_5 + \alpha^2 x_2 + \alpha^2 x_3 x_4 \\ &\quad + \alpha x_3 x_5 + x_3 + \alpha x_4, \\ f^{(6)} &= \alpha x_1^2 + x_1 x_2 + \alpha^2 x_1 x_3 + x_1 x_6 + \alpha^2 x_1 x_7 + x - 1 + \alpha x_2 x_3 + \alpha x_2 x_4 \\ &\quad + \alpha^2 x_2 x_7 + \alpha x_2 + \alpha^2 x_3 x_4 + x_3 x_5 + \alpha x_3 x_7 + \alpha x_3 + x_4^2 + \alpha x_4 x_6 \\ &\quad + \alpha^2 x_4 x_7 + x_4 + \alpha^2 x_5^2 + x_5 x_6 + x_5 x_7 + \alpha x_5 + \alpha^2 x_6 + \alpha x_7, \\ f^{(7)} &= x_1 x_2 + x_1 x_4 + x_1 x_5 + x_1 x_6 + \alpha^2 x_1 x_7 + \alpha^2 x_1 + \alpha x_2^2 + x_2 x_5 + \alpha x_2 x_7 \\ &\quad + \alpha^2 x_2 + \alpha x_3 x_4 + x_3 x_5 + x_3 x_6 + \alpha x_3 x_7 + x_3 + x_4^2 + \alpha x_4 x_6 + \alpha x_4 \\ &\quad + x_5^2 + \alpha^2 x_5 + \alpha^2 x_6 + x_7 + \alpha^2 \end{aligned}$$

Next, we compute the public key \mathcal{P} of the scheme as $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$, obtaining

$$p^{(1)} = \alpha^2 x_1^2 + x_1 x_2 + x_1 x_5 + x_1 x_6 + \alpha^2 x_1 x_7 + \alpha^2 x_1 + x_2^2 + x_2 x_3 + x_2 x_5$$

$$\begin{aligned}
& + \alpha^2 x_2 x_6 + \alpha^2 x_2 x_7 + x_2 + \alpha x_3 x_4 + x_3 x_7 + \alpha^2 x_3 + \alpha x_4^2 + \alpha x_4 x_5 \\
& + x_4 x_7 + x_4 + x_5^2 + x_5 x_6 + \alpha x_5 x_7 + \alpha x_5 + x_6^2 + \alpha x_6 x_7 + \alpha x_7^2 + \alpha^2 x_7, \\
p^{(2)} &= x_1^2 + x_1 x_2 + \alpha x_1 x_3 + \alpha x_1 x_4 + \alpha x_1 x_5 + \alpha x_1 x_6 + \alpha x_2^2 + \alpha x_2 x_3 \\
& + x_2 x_4 + \alpha^2 x_2 x_5 + \alpha x_2 x_6 + x_2 x_7 + \alpha x_2 + \alpha x_3^2 + \alpha^2 x_3 x_4 + \alpha x_3 x_5 \\
& + x_3 x_6 + \alpha^2 x_3 x_7 + \alpha^2 x_4^2 + \alpha^2 x_4 x_7 + \alpha^2 x_4 + x_5^2 + \alpha^2 x_5 x_6 + \alpha^2 x_5 x_7 \\
& + x_5 + \alpha^2 x_6^2 + x_6 x_7 + x_6 + \alpha^2 x_7^2 + \alpha^2 x_7, \\
p^{(3)} &= x_1 x_2 + x_1 x_3 + \alpha^2 x_1 x_5 + \alpha^2 x_1 x_6 + x_1 x_7 + \alpha^2 x_1 + \alpha^2 x_2^2 + \alpha^2 x_2 x_3 \\
& + \alpha^2 x_2 x_4 + x_2 x_5 + x_2 x_6 + x_2 x_7 + \alpha x_3 x_4 + \alpha^2 x_3 x_5 + \alpha x_3 x_6 + \alpha^2 x_4^2 \\
& + \alpha^2 x_4 x_5 + x_4 x_6 + \alpha^2 x_4 + \alpha x_5 x_7 + \alpha^2 x_5 + \alpha^2 x_6 x_7 + x_7^2 + \alpha^2 x_7 + 1, \\
p^{(4)} &= \alpha x_1^2 + \alpha x_1 x_3 + \alpha^2 x_1 x_4 + \alpha x_1 x_5 + \alpha^2 x_1 x_6 + x_1 x_7 + \alpha x_1 + \alpha^2 x_2^2 \\
& + x_2 x_3 + \alpha^2 x_2 x_4 + \alpha x_2 x_5 + \alpha x_2 x_6 + \alpha x_2 x_7 + \alpha^2 x_2 + \alpha x_3^2 + x_3 x_5 \\
& + x_3 x_6 + \alpha^2 x_3 x_7 + \alpha^2 x_3 + \alpha x_4 x_5 + x_4 + \alpha x_5^2 + \alpha^2 x_5 x_7 + \alpha^2 x_5 \\
& + \alpha x_6^2 + \alpha x_6 x_7 + \alpha^2 x_6 + x_7^2.
\end{aligned}$$

In order to generate a signature for the message (or hash value) $\mathbf{w} = (1, \alpha, \alpha, \alpha)$, we first compute $\mathbf{x} = \mathcal{S}^{-1}(\mathbf{w}) = (0, 1, \alpha, 1)$. We choose random values for the vinegar variables, setting $(x_1, x_2, x_3) = (0, \alpha, 0)$ and substitute these values into the central polynomials $f^{(4)}$ and $f^{(5)}$ of the first layer, obtaining

$$\begin{aligned}
\bar{f}^{(4)} &= x_4 + x_5 = \alpha, \\
\bar{f}^{(5)} &= x_4 = 0.
\end{aligned}$$

We therefore get $(x_4, x_5) = (0, \alpha)$ and substitute the values of x_1, \dots, x_5 into the central polynomials of the second layer, obtaining

$$\begin{aligned}
\bar{f}^{(6)} &= \alpha^2 x_6 + x_7 = \alpha, \\
\bar{f}^{(7)} &= x_6 + \alpha^2 x_7 = \alpha.
\end{aligned}$$

We find $(x_6, x_7) = (1, 1)$ and therefore

$$\mathcal{T}(\mathbf{z}) = (0, \alpha, 0, 0, \alpha, 1, 1).$$

By inverting the second affine map \mathcal{T} , we obtain the signature $\mathbf{z} \in \mathbb{F}^7$ as

$$\mathbf{z} = (\alpha^2, 1, \alpha, \alpha, 1, \alpha^2, 0).$$

In order to check, if \mathbf{z} is indeed a valid signature for the message \mathbf{w} , we substitute \mathbf{z} into the public key \mathcal{P} . Since we get

$$\mathbf{w}' = \mathcal{P}(\mathbf{z}) = (1, \alpha, \alpha, \alpha) = \mathbf{w},$$

the signature \mathbf{z} is accepted.

5.5 Attacks on Rainbow

The known attacks against the Rainbow signature scheme can be divided into three main groups

- Direct attacks,
- Rank attacks and
- Attacks against the underlying UOV structure.

In the following sections, we take a closer look on these attacks. For the simplicity of description, we assume that all the maps \mathcal{S} , \mathcal{F} and \mathcal{T} are homogeneous maps. We can therefore write the i -th component of the central map as a quadratic form $f^{(i)}(\mathbf{x}) = \mathbf{x}^T \cdot F^{(i)} \cdot \mathbf{x}$ with an $n \times n$ matrix $F^{(i)}$ and the maps \mathcal{S} and \mathcal{T} as matrices S and T respectively. We can compute the matrix $P^{(i)}$ representing the (now also homogeneous) i -th component of the public key as a matrix $P^{(i)}$ with

$$P^{(i)} = \sum_{j=1}^m s_{ij} \cdot T^T \cdot F^{(j)} \cdot T, \quad (5.22)$$

where s_{ij} is the (i, j) -th element of the $m \times m$ matrix S .

5.5.1 The Direct Attack

In a direct attack, the adversary considers the public equation $\mathcal{P}(\mathbf{z}) = \mathbf{w}$ as an instance of the MQ Problem and tries to solve it using an algorithm like XL or a Gröbner basis method such as F_4 or F_5 (see Chap. 8). Since the Rainbow public key is a slightly underdetermined system ($m \leq n \leq 2m$), the best strategy is usually to choose random values for $n - m$ variables to create a determined system before applying XL or the Gröbner basis algorithm.

One can expect that the projected system has exactly one solution. In some cases one gets even better results when guessing some more variables (Hybrid Approach [1]). Though one has to run the algorithm more often (approximately q^k times when guessing k variables over a field with q elements), the overall complexity of the attack might be smaller.

Experiments have shown that the multivariate quadratic systems given by Rainbow public keys behave very much like random systems. In particular, the degree of regularity of a Rainbow public key is the same as that of a random system of the same size. It is therefore relatively easy to estimate the complexity of a direct attack against the Rainbow signature scheme.

5.5.2 Rank Attacks

The goal of a Rank attack is to recover (parts of) the affine transformations \mathcal{S} and \mathcal{T} by looking at the rank of linear combinations of the public polynomials. For this, one considers the public polynomials $p^{(i)}$ as quadratic forms (see Sect. 2.1) and looks at the rank of the corresponding matrices $P^{(i)}$. In the case of the Rainbow signature scheme, Rank attacks come up in two flavors

- the MinRank attack and
- the HighRank attack.

5.5.3 The MinRank Attack

The goal of the MinRank attack [3, 8] is to find a linear combination of the matrices $P^{(k)}$ ($v_1 + 1 \leq k \leq n$) of very low rank (in the case of Rainbow, this rank is, as can be seen from Fig. 5.1, given by v_2). Such a matrix corresponds to a linear combination of the o_1 matrices $T^T \cdot F^{(k)} \cdot T$ ($k \in O_1$) representing the central polynomials of the first Rainbow layer. Therefore, we have to solve an instance of the

MinRank Problem Given a set of m $n \times n$ matrices P_1, \dots, P_m , find a linear combination $H = \sum_{i=1}^m \lambda_i P_i$ which has rank $\leq r$.

In the case of the Rainbow signature scheme, the m matrices of the MinRank Problem are the matrices $P^{(v_1+1)}, \dots, P^{(n)}$, while the minimal rank r is given by v_2 .

Let $P = \sum_{i=v_1+1}^n \lambda_i P^{(i)}$ be a linear combination of the matrices $P^{(i)}$ of rank v_2 . Then there exist $n - v_2$ linear independent vectors $\mathbf{b}_1, \dots, \mathbf{b}_{n-v_2}$ such that $P \cdot \mathbf{b}_i = 0 \forall i = 1, \dots, n - v_2$. The probability that $P \cdot \mathbf{b} = 0$ holds for a randomly chosen vector $\mathbf{b} \in \mathbb{F}^n$ is therefore q^{-v_2} . However, as the following result shows, we can, in the case of Rainbow, find the vectors $\mathbf{b}_1, \dots, \mathbf{b}_{n-v_2}$ much cheaper.

Proposition 5.11 (Billet, Gilbert) *Let $\mathbf{w} \in \mathbb{F}^n$ be a vector whose first v_1 components are zero. Then, with probability at least $1/q$, there exists a non trivial linear combination M of the matrices $F^{(v_1+1)}, \dots, F^{(v_2)}$ such that $M \cdot \mathbf{w} = 0$.*

Proof Let $\mathbf{w} \in \mathbb{F}^n$ be a vector whose first v_1 components are zero and let $\mathbf{w}^{(i)} = F^{(v_1+i)} \cdot \mathbf{w}$ ($i = 1, \dots, o_1$). Then, due to the structure of the matrices

$F^{(v_1+1)}, \dots, F^{(v_2)}$ (see Fig. 5.1), each vector $\mathbf{w}^{(i)}$ has at most v_2 non zero entries. Assuming that the vectors $\mathbf{w}^{(1)}, \dots, \mathbf{w}^{(o_1)}$ are uniformly distributed, the probability that they are linearly independent is given by

$$\prod_{i=0}^{o_1-1} \left(1 - \frac{q^i}{q^{o_1}}\right) < \frac{q-1}{q}.$$

Therefore, the probability that the vector \mathbf{w} lies in the kernel of some non trivial linear combination of the matrices $F^{(v_1+1)}, \dots, F^{(v_2)}$, is greater than $1/q$. \square

However, we do not know if, for a given vector $\mathbf{b} \in \mathbb{F}^n$, the vector $\mathbf{w} = \mathcal{T}(\mathbf{b})$ fulfills the condition of Proposition 5.11. The probability that, for a randomly chosen vector $\mathbf{b} \in \mathbb{F}^n$, the first v_1 components of $\mathbf{w} = \mathcal{T}(\mathbf{b})$ are zero, is given by q^{-v_1} . Therefore, the cost of finding a vector \mathbf{b} in the kernel of the linear combination P is about $q^{(v_1+1)}$ (plus some linear algebra cost).

Algorithm 5.5 shows how to find a low rank linear combination C of the matrices $P^{(v_1+1)}, \dots, P^{(n)}$.

Algorithm 5.5 MinRank attack

Input: matrices $P^{(v_1+1)}, \dots, P^{(n)}$

Output: Linear combination $C = \sum_{i=v_1+1}^n c_i \cdot P^{(i)}$ of rank $\leq v_2$

1: **repeat**

2: Choose randomly a vector $\lambda \in \mathbb{F}^m$ and compute $P = \sum_{i=v_1+1}^n \lambda_i P^{(i)}$.

3: **if** Rank $(P) > 1$ and Rank $(P) < n$ **then**

4: Choose randomly a vector γ from $\ker(P)$.

5: $C \leftarrow \sum_{i=v_1+1}^n \gamma_i P^{(i)}$

6: **end if**

7: **until** Rank $(C) \leq v_2$

8: **return** C

By finding o_1 linear independent low rank linear combinations of the matrices $P^{(v_1+1)}, \dots, P^{(n)}$, we can extract the first Rainbow layer. This step costs approximately $o_1 q^{v_1+1}$ operations, where q is the cardinality of the underlying field.

The remaining Rainbow layers can now be extracted using a similar technique. However, since the attacker has, from the first step, partial knowledge of the secret transformation of variables, the complexity of extracting the remaining layers is much lower than that of the first step and therefore can be neglected.

Having separated all the Rainbow layers, the attacker is able to generate signatures the same way as a legitimate user. The complexity of the attack is mainly given by the complexity of extracting the first Rainbow layer. So we have

$$\text{complexity}_{\text{MinRank}} = o_1 q^{v_1+1} \left(\frac{m^3}{3} - \frac{m^2}{6} \right). \quad (5.23)$$

5.5.4 The HighRank Attack

Before we come to the description of the attack itself, we introduce some notation. Recall that, for the description of the Kipnis–Shamir attack on balanced Oil and Vinegar (see Sect. 5.2), we defined the Oil subspace \mathcal{O} as $\mathcal{O} = \{\mathbf{x} \in \mathbb{F}^n \mid x_1 = \dots = x_v = 0\}$. Analogously to this, we now define the Oil spaces $\mathcal{O}_1, \dots, \mathcal{O}_u$ as

$$\mathcal{O}_i = \{\mathbf{x} \in \mathbb{F}^n \mid x_1 = \dots = x_{v_i} = 0\}.$$

In other words, the Oil space \mathcal{O}_i contains only those vectors $\mathbf{x} \in \mathbb{F}^n$, for which all the vinegar variables of the i -th Rainbow layer are zero. Note that we have

$$\mathcal{O}_u \subset \mathcal{O}_{u-1} \subset \dots \subset \mathcal{O}_2 \subset \mathcal{O}_1 \subset \mathbb{F}^n.$$

Further note that, for each matrix $F^{(k)}$ with $k \in \mathcal{O}_i$ and each vector $\mathbf{x} \in \mathcal{O}_i$, we have

$$\mathbf{x}^T \cdot F^{(k)} \cdot \mathbf{x} = 0$$

Therefore we have

$$\mathcal{O}_i \subset \ker(F^{(k)}) \quad \forall k \in \mathcal{O}_i.$$

With this notation, we now can interpret the MinRank attack in a different way: As the matrices C found by Algorithm 5.5 are linear combinations of the matrices $T^T \cdot F^{(k)} \cdot T$ ($k = v_1 + 1, \dots, v_2$), we have $C(\mathbf{x}) = 0$ for all $\mathbf{x} \in \mathcal{T}^{-1}(\mathcal{O}_1)$ and therefore $\mathcal{T}^{-1}(\mathcal{O}_1) \subset \ker(C)$. Hence the MinRank attack can be seen as an attack looking for the space $\mathcal{T}^{-1}(\mathcal{O}_1)$ or as an attack that finds a large kernel shared by a small number of linear combinations $C = \sum_{k=v_1+1}^n \lambda_k P^{(k)}$.

The HighRank attack as proposed by Coppersmith et al. in [4] turns this around. Now we look for a small kernel ($\mathcal{T}^{-1}(\mathcal{O}_u)$) shared by a large number of linear combinations $\sum_{k=v_1+1}^n \lambda_k P^{(k)}$.

The HighRank attack is based on the following observation. The variables x_{v_u+1}, \dots, x_n appear only in the quadratic cross terms of the central polynomials $f^{(v_u+1)}, \dots, f^{(n)}$ of the last Rainbow layer. Therefore we get $\mathcal{O}_u \subset \ker \sum_{k=v_1+1}^{v_u} \alpha_k F^{(k)}$ for arbitrary vectors $\alpha \in \mathbb{F}^{m-o_u}$ which means that $\mathcal{T}^{-1}(\mathcal{O}_u)$ lies in the kernel of certain linear combinations of the matrices $P^{(k)}$ ($k = 1, \dots, n$).

Algorithm 5.6 shows the functioning of this attack to find the space $\mathcal{T}^{-1}(\mathcal{O}_u)$.

The complexity of this step can be estimated by $q^{o_u} \frac{n^3}{6}$.

By studying the subspaces of $\mathcal{T}^{-1}(\mathcal{O}_u)$ we can find bigger kernels (which correspond to $\mathcal{T}^{-1}(\mathcal{O}_i)$ ($i = u - 1, \dots, 1$)). Since the complexity of this step can be neglected, we get

$$\text{complexity}_{\text{HighRank}} = q^{o_u} \frac{n^3}{6}. \quad (5.24)$$

Algorithm 5.6 HighRank attack**Input:** public matrices $P^{(v_1+1)}, \dots, P^{(n)}$ **Output:** $\mathcal{T}^{-1}(\mathcal{O}_u)$

- 1: Form an arbitrary linear combination $H = \sum_{k=v_1+1}^n \lambda_k P^{(k)}$. Find $V = \ker H$.
- 2: If $\dim V \geq 1$, set $(\sum_{k=v_1+1}^n \lambda_k P^{(k)}) V = 0$. Test, if the solution set has dimension $m - o_u$.
- 3: With probability q^{-o_u} , we have therefore found $V \subset \mathcal{T}^{-1}(\mathcal{O}_u)$. We continue this process, until we have found the whole space $\mathcal{T}^{-1}(\mathcal{O}_u)$.
- 4: **return** $\mathcal{T}^{-1}(\mathcal{O}_u)$

5.5.5 Attacks Using the Underlying UOV Structure

An instance of the Rainbow signature scheme with parameters v_1, o_1, \dots, o_u can be seen as a (special) instance of a UOV scheme with v_u vinegar variables and o_u Oil variables. Therefore, all attacks against UOV handled in Sect. 5.3 work against Rainbow, too. For the complexities of the attacks we get:

- UOV attack

$$\text{complexity}_{\text{UOV attack Rainbow}}(q, v_u, o_u, n) = q^{v_u - o_u - 1} o_u^4 = q^{n - 2o_u - 1} \cdot o_u^4, \quad (5.25)$$

- The complexity of the UOV Reconciliation attack is mainly determined by the complexity of solving the first system of m quadratic equations in m variables. For details see Sect. 8.7.

Furthermore, we can use the additional structure in the Rainbow central map (see Fig. 5.1) to improve the UOV Reconciliation attack. This approach leads to the so called Rainbow Band Separation (RBS) attack.

5.5.6 The Rainbow Band Separation Attack

The Rainbow Band Separation (or RBS) attack [6] can be seen as an extension of the UOV Reconciliation attack (see Sect. 5.3.1.2) to Rainbow and uses the layer structure of this scheme (i.e. the additional blocks of zeros in the polynomials of the Rainbow central map). Similar to the UOV Reconciliation attack, the RBS attack looks for private keys of a special form.

5.5.6.1 Equivalent Keys for Rainbow

Let $((\mathcal{S}, \mathcal{F}, \mathcal{T}), \mathcal{P})$ be a key pair of the Rainbow signature scheme. As in Sect. 5.3, we denote a second Rainbow private key $(\mathcal{S}', \mathcal{F}', \mathcal{T}')$ with $\mathcal{S}' \circ \mathcal{F}' \circ \mathcal{T}' = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T} = \mathcal{P}$ as an *equivalent key*.

The following theorem was proven in [16].

Theorem 5.12 *Let $(\mathcal{S}, \mathcal{F}, \mathcal{T})$ be a Rainbow private key and let $\Sigma : \mathbb{F}^m \rightarrow \mathbb{F}^m$ and $\Omega : \mathbb{F}^n \rightarrow \mathbb{F}^n$ be linear maps with*

$$\Sigma = \begin{pmatrix} \Sigma_{o_1 \times o_1}^{(1,1)} & 0_{o_1 \times o_2} & \cdots & 0_{o_1 \times o_u} \\ \Sigma_{o_2 \times o_1}^{(2,1)} & \Sigma_{o_2 \times o_2}^{(2,2)} & \ddots & \vdots \\ \vdots & & \ddots & 0_{o_{u-1} \times o_u} \\ \Sigma_{o_u \times o_1}^{(u,1)} & \Sigma_{o_u \times o_2}^{(u,2)} & \cdots & \Sigma_{o_u \times o_u}^{(u,u)} \end{pmatrix},$$

$$\Omega = \begin{pmatrix} \Omega_{v_1 \times v_1}^{(1,1)} & 0_{v_1 \times o_1} & \cdots & 0_{v_1 \times o_u} \\ \Omega_{o_1 \times v_1}^{(2,1)} & \Omega_{o_1 \times o_1}^{(2,2)} & \ddots & \vdots \\ \vdots & & \ddots & 0_{o_{u-1} \times o_u} \\ \Omega_{o_u \times v_1}^{(u+1,1)} & \Omega_{o_u \times o_1}^{(u+1,2)} & \cdots & \Omega_{o_u \times o_u}^{(u+1,u+1)} \end{pmatrix}. \quad (5.26)$$

Then $(\mathcal{S}', \mathcal{F}', \mathcal{T}')$ with $\mathcal{S}' = \mathcal{S} \circ \Sigma^{-1}$, $\mathcal{F}' = \Sigma \circ \mathcal{F} \circ \Omega$ and $\mathcal{T}' = \Omega^{-1} \circ \mathcal{T}$ is an equivalent Rainbow private key.

Note that Σ^{-1} and Ω^{-1} have the same form as Σ and Ω respectively.

The next theorem states that for any Rainbow public key \mathcal{P} there exists, with overwhelming probability, a corresponding Rainbow private key of a very special form.

Theorem 5.13 *Let \mathcal{P} be a Rainbow public key. Then, with overwhelming probability, there exists a Rainbow private key $(\tilde{\mathcal{S}}, \tilde{\mathcal{F}}, \tilde{\mathcal{T}})$ with $\tilde{\mathcal{S}} \circ \tilde{\mathcal{F}} \circ \tilde{\mathcal{T}} = \mathcal{P}$ such that*

$$\tilde{\mathcal{S}} = \begin{pmatrix} 1_{o_1 \times o_1} & \tilde{\mathcal{S}}_{o_1 \times o_2}^{(1,2)} & \cdots & \tilde{\mathcal{S}}_{o_1 \times o_u}^{(1,u)} \\ 0_{o_2 \times o_1} & 1_{o_2 \times o_2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \tilde{\mathcal{S}}_{o_{u-1}, o_u}^{(u-1,u)} \\ 0_{o_u, o_1} & \cdots & 0_{o_u, o_{u-1}} & 1_{o_u \times o_u} \end{pmatrix} \text{ and}$$

$$\tilde{\mathcal{T}} = \begin{pmatrix} 1_{v_1 \times v_1} & \tilde{\mathcal{T}}_{v_1 \times o_1}^{(1,2)} & \cdots & \tilde{\mathcal{T}}_{v_1 \times o_u}^{(1,u+1)} \\ 0_{o_1 \times v_1} & 1_{o_1 \times o_1} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \tilde{\mathcal{T}}_{o_{u-1}, o_u}^{(u,u+1)} \\ 0_{o_u, v_1} & \cdots & 0_{o_u, o_{u-1}} & 1_{o_u \times o_u} \end{pmatrix}. \quad (5.27)$$

Proof (sketch) Since \mathcal{P} is a Rainbow public key, we can be sure that there exists a Rainbow private key $(\mathcal{S}, \mathcal{F}, \mathcal{T})$ with $\mathcal{S} \circ \mathcal{F} \circ \mathcal{T} = \mathcal{P}$. We have to show that there exist linear maps Σ and Ω of form (5.26) which transform the maps \mathcal{S} and \mathcal{T} into linear transformations of the form (5.27). In the following, we restrict to a Rainbow scheme with two layers. The general case can be shown analogously. Therefore we have

$$\tilde{\mathcal{S}} = \begin{pmatrix} 1_{o_1 \times o_1} & \mathcal{S}'_{o_1 \times o_2} \\ 0_{o_2 \times o_1} & 1_{o_2 \times o_2} \end{pmatrix}, \quad \tilde{\mathcal{T}} = \begin{pmatrix} 1_{v_1 \times v_1} & T'_{v_1 \times o_1}(1) & T'_{v_1 \times o_2}(2) \\ 0_{o_1 \times v_1} & 1_{o_1 \times o_1} & T'_{o_1 \times o_2}(3) \\ 0_{o_2 \times v_1} & 0_{o_2 \times o_1} & 1_{o_2 \times o_2} \end{pmatrix}. \quad (5.28)$$

Let

$$\mathcal{S} = \begin{pmatrix} \mathcal{S}_{o_1 \times o_1}^{(1)} & \mathcal{S}_{o_1 \times o_2}^{(2)} \\ \mathcal{S}_{o_2 \times o_1}^{(3)} & \mathcal{S}_{o_2 \times o_2}^{(4)} \end{pmatrix}, \quad \mathcal{T} = \begin{pmatrix} T_{v_1 \times v_1}^{(1)} & T_{v_1 \times o_1}^{(2)} & T_{v_1 \times o_2}^{(3)} \\ T_{o_1 \times v_1}^{(4)} & T_{o_1 \times o_1}^{(5)} & T_{o_1 \times o_2}^{(6)} \\ T_{o_2 \times v_1}^{(7)} & T_{o_2 \times o_1}^{(8)} & T_{o_2 \times o_2}^{(9)} \end{pmatrix}.$$

We have to show that there exist linear maps Σ and Ω of the form

$$\Sigma = \begin{pmatrix} \Sigma_{o_1 \times o_1}^1 & 0_{o_1 \times o_2} \\ \Sigma_{o_2 \times o_1}^3 & \Sigma_{o_2 \times o_2}^4 \end{pmatrix}, \quad \Omega = \begin{pmatrix} \Omega_{v_1 \times v_1}^{(1)} & 0_{v_1 \times o_1} & 0_{v_1 \times o_2} \\ \Omega_{o_1 \times v_1}^{(3)} & \Omega_{o_1 \times o_1}^{(4)} & 0_{o_1 \times o_2} \\ \Omega_{o_2 \times v_1}^{(7)} & \Omega_{o_2 \times o_1}^{(8)} & \Omega_{o_2 \times o_2}^{(9)} \end{pmatrix}.$$

such that $\tilde{\mathcal{S}} = \mathcal{S} \circ \Sigma^{-1}$ and $\tilde{\mathcal{T}} = \Omega^{-1} \circ \mathcal{T}$ have the form of (5.28). If $\mathcal{S}^{(4)}$ is invertible,⁵ we get from $\mathcal{S} = \tilde{\mathcal{S}} \circ \Sigma$

- $\Sigma^{(1)} = \mathcal{S}^{(1)} - \mathcal{S}^{(2)} \cdot (\mathcal{S}^{(4)})^{-1} \cdot \mathcal{S}^{(3)}$,
- $\Sigma^{(3)} = \mathcal{S}^{(3)}$ and
- $\Sigma^{(4)} = \mathcal{S}^{(4)}$.

The matrix \mathcal{S}' of (5.28) is given as

$$\mathcal{S}' = \mathcal{S}^{(2)} \cdot (\mathcal{S}^{(4)})^{-1}. \quad (5.29)$$

For the transformation of the variables we get: If $T^{(1)}$ and $T^{(5)} - T^{(4)} \cdot (T^{(1)})^{-1} \cdot T^{(2)}$ are invertible,⁶ we get from $\mathcal{T} = \Omega \circ \tilde{\mathcal{T}}$

⁵If $\mathcal{S}^{(4)}$ is not invertible, we can switch the rows and columns of \mathcal{S} by renumbering the equations of \mathcal{F} until we get an invertible matrix.

⁶Again it is possible to switch rows and columns of \mathcal{T} by renumbering the variables.

- $\Omega^{(1)} = T^{(1)}$,
- $\Omega^{(4)} = T^{(4)}$,
- $\Omega^{(7)} = T^{(7)}$,
- $T'^{(1)} = (T^{(1)})^{-1} \cdot T^{(2)}$,
- $T'^{(2)} = (T^{(1)})^{-1} \cdot T^{(3)}$,
- $\Omega^{(5)} = T^{(5)} - T^{(4)} \cdot (T^{(1)})^{-1} \cdot T^{(2)}$,
- $T'^{(3)} = (\Omega^{(5)})^{-1} \cdot (T^{(6)} - T^{(4)} \cdot (T^{(1)})^{-1} \cdot T^{(3)})$,
- $\Omega^{(8)} = T^{(8)} - T^{(7)} \cdot (T^{(1)})^{-1} \cdot T^{(2)}$ and
- $\Omega^{(9)} = T^{(9)} - T^{(7)} \cdot (T^{(1)})^{-1} \cdot T^{(3)} - \Omega^{(8)} \cdot T'^{(3)}$.

□

The matrix \tilde{T} of (5.27) can be written as a product of matrices

$$\tilde{T} = T_{v_1+1} \cdot \dots \cdot T_n$$

with

$$T_i = \begin{pmatrix} 1 & 0 & 0 & t_{1,i} & 0 \\ & \ddots & \vdots & \vdots & \vdots \\ 0 & 1 & 0 & t_{v_\ell,i} & 0 \\ 0 & \dots & 0 & 1 & 0 & 0 \\ \vdots & & \vdots & & \ddots & \\ 0 & \dots & 0 & 0 & & 1 \end{pmatrix} \quad (i = v_1 + 1, \dots, n). \quad (5.30)$$

Here, ℓ is the uniquely determined integer $1 \leq \ell \leq u$ such that $i \in O_\ell$. Note that, besides the 1's on the main diagonal, the matrices T_i ($i = v_1 + 1, \dots, n$) contain exactly v_ℓ non zero elements, which are located at the first v_ℓ positions in the i -th column of the matrix T_i .

By inversion we get $\tilde{T}^{-1} = T_n^{-1} \cdot \dots \cdot T_{v_1+1}^{-1}$. Note that the matrices T_i^{-1} have the same structure as the matrices T_i ($i = v_1 + 1, \dots, n$).

5.5.6.2 The Attack

The Rainbow Band Separation attack is based on the following observation:

Let \mathcal{P} be a Rainbow public key and $(\mathcal{S}, \mathcal{F}, \mathcal{T})$ be a corresponding private key such that S and T are of the form (5.27). We write T^{-1} as a product of matrices $\tilde{T}_n \cdot \dots \cdot \tilde{T}_{v_1+1}$ as shown in (5.30) and get

$$F^{(k)} = \sum_{l=1}^m \tilde{s}_{kl} \left(\tilde{T}_{v_1+1}^T \cdot \dots \cdot \tilde{T}_n^T \cdot P^{(l)} \cdot \tilde{T}_n \cdot \dots \cdot \tilde{T}_{v_1+1} \right) \quad (5.31)$$

(see (5.22)). Here, \tilde{s}_{kl} ($k, l = 1, \dots, m$) denote the elements of the matrix S^{-1} .

As for the UOV Reconciliation attack (see Sect. 5.3.1.2), the goal of the Rainbow Band Separation attack is to find, for every $k = v_1 + 1, \dots, n$, a sequence of matrices $P_n^{(k)} = P^{(k)}, P_{n-1}^{(k)}, \dots, P_{v_1}^{(k)}$, such that $P_{v_1}^{(k)}$ has the form of $F^{(k)}$. The matrices $P_{v_1}^{(k)}$ ($k = v_1 + 1, \dots, n$) (together with the matrices \tilde{T}_i ($i = v_1 + 1, \dots, n$) and the elements \tilde{s}_{kl}) yield therefore an equivalent Rainbow private key which can be used by the attacker to generate signatures in the same way as a legitimate user.

As in the case of the UOV Reconciliation attack, the block of zeros in the bottom right corner increases when going from $P_j^{(k)}$ to $P_{j-1}^{(k)}$. Every additional zero position in the matrix $P_{j-1}^{(k)}$ yields one non linear polynomial equation in the unknown elements of the matrix \tilde{T}_j and $\tilde{s}_{k,\ell}$.

By considering all matrices $P_{j-1}^{(k)}$, we obtain a system \mathcal{Q}_j of $(n-j+1)(m+n-1)$ (mostly quadratic) equations in n variables, which can be solved by XL or a Gröbner basis method. By doing so, the attacker is able to find the matrix \tilde{T}_j and the elements \tilde{s}_{kl} and therefore can compute the matrices $P_{j-1}^{(k)}$ ($k = v_1 + 1, \dots, n$).

In contrast to the UOV Reconciliation attack, the equations in the system \mathcal{P} are no longer homogeneous quadratic in the elements of the matrix \tilde{T}_j , but some of them are now cubic in the elements of \tilde{T}_j and $\tilde{s}_{k,\ell}$. This fact also increases the number of variables in the system.

On the other hand, we can make use of the additional zero blocks in the Rainbow central map (see Fig. 5.1) to increase the number of equations in the system. By balancing out these two facts, we find that the systems produced by the Rainbow Band Separation attack can be solved more efficiently than those produced by the UOV Reconciliation attack.

To find the matrices $P_{v_1}^{(k)}$ and therefore an equivalent private key, the attacker has to solve m systems \mathcal{Q}_j (for $j = n, \dots, v_1 + 1$). However, since the number of equations in the system \mathcal{Q}_j increases from step to step, the systems \mathcal{Q}_j become easier to solve with decreasing j .

Therefore, the complexity of the Rainbow Band Separation attack is mainly determined by the complexity of solving the first system \mathcal{Q}_n which consists of

- one cubic equation
- $m - 1$ quadratic equations in the variables of \tilde{T}_n
- $n - 1$ bilinear equations (linear both in the elements of \tilde{T}_n and the \tilde{s}_{nk} ($k = v_u + 1, \dots, n$)).

5.5.7 Practical Parameters

By considering the above attacks, we propose in Table 5.2 practical parameters for the Rainbow signature scheme for the security categories I II and III.

Table 5.2 Parameters and key sizes of rainbow

Security category	Parameters (\mathbb{F} , v_1 , o_1 , o_2)	Public key size (kB)	Private key size (kB)	Signature size (bit)
I	(GF(256), 40, 24, 24)	187.7	140.0	704
II	(GF(256), 68, 36, 36)	703.9	525.2	1120
III	(GF(256), 92, 48, 48)	1683.3	1244.4	1504

5.6 Reducing the Public Key Size

The main disadvantage of the UOV signature scheme (and, to some degree, also of Rainbow) is the large size of the public and private keys of the schemes. In this section we describe techniques to deal with this problem.

Section 5.6.1 describes the SUOV (structured UOV) signature scheme proposed by Petzoldt et al. in [14]. The scheme allows to choose the public key of UOV as a structured matrix and therefore makes it possible to reduce the public key size of the scheme significantly. Furthermore, the additional structure in the public key can be used to speed up the signature verification process of the scheme [15]. Until now, no attacks using the special structure of the public key are known.

Section 5.6.2 shows how to extend this technique to the Rainbow signature scheme.

Finally, in Sect. 5.6.4, we present the LUOV signature scheme of Beullens et al., which can be seen as a special instance of SUOV. LUOV chooses the coefficients of the public and private key from a small subfield of \mathbb{F} , whereas messages and signatures are defined over the field \mathbb{F} itself. To preserve the security of the scheme, we have to choose larger parameters than in the case of the standard UOV scheme. However, since major parts of the keys can be stored as small seeds, the technique still allows one to reduce the key sizes significantly.

5.6.1 StructuredUOV

In [14], Petzoldt et al. developed a technique to reduce the public key size of UOV by a large factor. The basic idea of their technique is to insert a structured matrix B into the Macaulay matrix M_P of the public key (see Fig. 5.2). The matrix B can be chosen in cyclic form or can be generated from a small random seed, which reduces the memory needed to store the public key drastically.

In a sense, the technique turns around the standard key generation process of a public key cryptosystem. During the standard key generation process, the public key is computed out of the private key. Instead of this, we compute, for the StructuredUOV scheme, the central map out of the public key and the linear transformation \mathcal{T} (see Fig. 5.2).

In this section we describe Petzoldt's technique to reduce the public key of the UOV signature scheme, while, in the next section, we show how to extend the technique to Rainbow. To simplify our description, we restrict ourselves to homogeneous maps \mathcal{F} and \mathcal{T} . Note that this leads to a homogeneous quadratic public key \mathcal{P} .

Recall that the public key of the UOV signature scheme is given by

$$\mathcal{P} = \mathcal{F} \circ \mathcal{T}, \quad (5.32)$$

where \mathcal{F} is a UOV central map and \mathcal{T} is a randomly chosen invertible linear map (given by an $n \times n$ matrix T).

Let $f_{ij}^{(k)}$ and $p_{ij}^{(k)}$ be the coefficients of the monomial $x_i x_j$ in the k -th component of \mathcal{F} and \mathcal{P} respectively ($1 \leq i \leq j \leq n$, $1 \leq k \leq o$). Note that, due to the special structure of the UOV central map \mathcal{F} , some of the coefficients $f_{ij}^{(k)}$ are fixed to 0. In particular, we have

$$f_{ij}^{(k)} = 0 \quad \forall i \in O \wedge j \in O, 1 \leq k \leq o \Leftrightarrow v+1 \leq i \leq j \leq n, 1 \leq k \leq o. \quad (5.33)$$

The key observation of [14] is the following. Equation (5.32) implies

$$p_{ij}^{(k)} = \sum_{r=1}^n \sum_{s=r}^n \alpha_{ij}^{rs} f_{rs}^{(k)} \stackrel{(5.33)}{=} \sum_{r=1}^v \sum_{s=r}^n \alpha_{ij}^{rs} f_{rs}^{(k)} \quad (1 \leq i \leq j \leq n, 1 \leq k \leq o) \quad (5.34)$$

with

$$\alpha_{ij}^{rs} = \begin{cases} t_{ri} t_{sj} & (i = j) \\ t_{ri} t_{sj} + t_{rj} t_{si} & \text{otherwise} \end{cases}. \quad (5.35)$$

After fixing the elements of the matrix T to some random values of \mathbb{F} , (5.34) becomes a linear relation between the coefficients $p_{ij}^{(k)}$ and $f_{rs}^{(k)}$ ($1 \leq i \leq j \leq n$, $1 \leq r \leq v$, $r \leq s \leq n$, $1 \leq k \leq o$).

To simplify our notation, we define two integers D and D' as follows. Let

- $D := \frac{v(v+1)}{2} + ov$ be the number of non zero quadratic terms in the components of \mathcal{F} and
- $D' := \frac{n(n+1)}{2}$ be the number of quadratic terms in the public polynomials.

Let M_P and M_F be the Macaulay matrices of \mathcal{P} and \mathcal{F} respectively (w.r.t. the graded lexicographic ordering of monomials, see Definition 2.10). Note that, due to the absence of oil \times oil terms in the central polynomials, M_F is of the form $M_F = (Q|0)$ with a block of zeros on the right and an $o \times D$ matrix Q .

Analogously, we divide the matrix M_P into two submatrices as $M_P = (B|C)$, where B is an $o \times D$ matrix and C is an $o \times (D' - D)$ matrix with elements in \mathbb{F} .

Furthermore we define a transformation matrix $\hat{A}_{\text{UOV}} \in \mathbb{F}^{D \times D'}$ containing the coefficients α_{ij}^{rs} of (5.34) by

$$\hat{A}_{\text{UOV}} = (\alpha_{ij}^{rs})$$

$1 \leq r \leq v, r \leq s \leq n$ for the rows; $1 \leq i \leq j \leq n$ for the columns, i.e.

$$\hat{A}_{\text{UOV}} = \begin{pmatrix} \alpha_{11}^{11} & \alpha_{12}^{11} & \dots & \alpha_{nn}^{11} \\ \alpha_{11}^{12} & \alpha_{12}^{12} & \dots & \alpha_{nn}^{12} \\ \vdots & & & \vdots \\ \alpha_{11}^{vn} & \alpha_{12}^{vn} & \dots & \alpha_{nn}^{vn} \end{pmatrix}. \quad (5.36)$$

For both rows and columns, the elements of \hat{A}_{UOV} are ordered with respect to the graded lexicographic order.

With this notation, (5.34) yields

$$M_P = Q \cdot \hat{A}_{\text{UOV}} \quad (5.37)$$

Let A_{UOV} be the $D \times D$ submatrix obtained by restricting \hat{A}_{UOV} to its first D columns. With this, (5.37) yields

$$B = Q \cdot A_{\text{UOV}}. \quad (5.38)$$

or, if A_{UOV} is invertible

$$Q = B \cdot A_{\text{UOV}}^{-1}. \quad (5.39)$$

Equation (5.39) allows us to fix the matrix B and, after having assigned random values to the elements of the matrix T , to derive from it the non zero part of the UOV central map \mathcal{F} . Algorithm 5.7 shows this alternative key generation process for the UOV signature scheme in algorithmic form.

The algorithm takes as input a matrix $B \in \mathbb{F}^{o \times D}$ and chooses randomly an $n \times n$ matrix T (representing the secret linear transformation \mathcal{T}). If both T and the corresponding transformation matrix A_{UOV} are invertible,⁷ it computes out of B and T the non zero coefficients of the quadratic monomials in \mathcal{F} . Finally, the algorithm computes the public key \mathcal{P} and returns the UOV key pair $((\mathcal{F}, T), \mathcal{P})$.

By applying Algorithm 5.7 we can insert arbitrary matrices B into the Macaulay matrix of the public key. For example, we can choose B to be

- a (partially) cyclic matrix (\rightarrow cyclicUOV)
- generated by a linear feedback shift register (\rightarrow UOVLFRS)

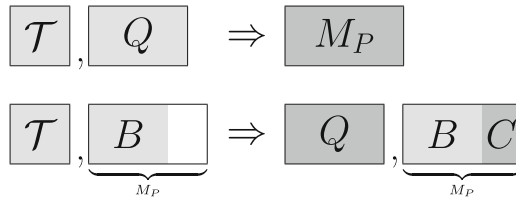
⁷Experiments have shown that this is the case with high probability. Therefore, in most cases, we need only one run of the loop in line 1 to 4 of Algorithm 5.7.

Algorithm 5.7 Key generation of StructuredUOV**Input:** parameters (\mathbb{F}, o, v) , matrix $B \in \mathbb{F}^{o \times D}$ **Output:** UOV key pair $((\mathcal{F}, \mathcal{T}), \mathcal{P})$

- 1: **repeat**
- 2: Choose randomly a linear map \mathcal{T} (represented by an $n \times n$ -matrix T).
 If T is not invertible, choose again.
- 3: Compute for \mathcal{T} the corresponding transformation matrix A_{UOV}
 (using Eqs. (5.35) and (5.36)).
- 4: **until** $\text{IsInvertible}(A_{\text{UOV}}) = \text{TRUE}$.
- 5: Compute the matrix Q containing the quadratic coefficients of the central polynomials by (5.39).
- 6: Compute the public key as $\mathcal{P} = \mathcal{F} \circ \mathcal{T}$.
- 7: **return** $((\mathcal{F}, \mathcal{T}), \mathcal{P})$

Table 5.3 Parameters and key sizes of StructuredUOV

Security category	Parameters (\mathbb{F}, v, o)	Public key size (kB)	Private key size (kB)	Signature size (bit)
I	(GF(256), 96, 48)	71.0	441.0	1152
II	(GF(256), 144, 72)	220.4	1478.3	1728
III	(GF(256), 192, 96)	500.0	3492.1	2304

**Fig. 5.2** Standard key generation (above) and alternative key generation of the UOV signature scheme. The light gray parts are chosen by the user, while the dark gray parts are computed during the key generation process

- generated by a PRNG (\rightarrow UOVPRNG)

Until now, no attack is known which uses the additional structure in the public key of StructuredUOV. Therefore it seems that we can use the same parameters as for standard UOV (see Table 5.3).

Key Sizes and Efficiency

The public key size of StructuredUOV is given as

$$\text{size}_{\text{pk StructuredUOV}} = o \left(\frac{o(o+1)}{2} \right)$$

field elements plus whatever it takes to store the structured part of the public key. For cyclicUOV, we need for this $\frac{v(v+1)}{2} + ov + n + 1$ field elements, for all other cases we can store the structured part of the public key as a 256 bit seed. This leads to a reduction of the public key size of up to 86 % compared to the standard UOV scheme.

The size of the private key is the same as for the standard UOV scheme, i.e.

$$\text{size}_{\text{sk StructuredUOV}} = \underbrace{n^2}_{\mathcal{T}} + \underbrace{o \left(\frac{v(v+1)}{2} + ov \right)}_{\mathcal{F}}$$

(Note for this formula that we are dealing with homogeneous maps \mathcal{T} and \mathcal{F}).

The efficiency of the key generation algorithm is, using the technique described in the next section, basically the same as for the standard scheme. The same holds for the signature generation process. For the signature verification process, we can, in the case of cyclicUOV or UOVLSR, use the technique described in [15] to get a speed up of up to 50%.

5.6.2 The Case of Rainbow

For simplicity, we again assume that all the maps \mathcal{S} , \mathcal{F} , \mathcal{T} and \mathcal{P} are homogeneous maps. When trying to extend the technique described in the previous section to the Rainbow signature scheme, we have to take into account the second linear map \mathcal{S} used in Rainbow. To cover this, we introduce a new map $\mathcal{Q} = \mathcal{F} \circ \mathcal{T}$ and apply the technique presented in the previous section for each Rainbow layer separately.

By doing so, we can, for every component $p^{(i)}$ of the public key, fix $\frac{v_i(v_i+1)}{2} + o_i v_i$ of the quadratic coefficients. Note that this is exactly the number of non zero coefficients in the i -th component of the central map. This leads to the “staircase” structure as seen in Figs. 5.3 and 5.4.

In order to describe the algorithm in detail, we need some additional notations. First, we introduce integers D_1, \dots, D_u where $D_i = \frac{v_i(v_i+1)}{2} + o_i v_i$ denotes the number of non zero quadratic terms in the central polynomials of the i -th Rainbow layer ($i = 1, \dots, u$). Additionally, we set $D_0 = 0$ and denote by $D_{u+1} = \frac{n(n+1)}{2}$ the number of quadratic terms in the public key.

Secondly, we introduce a special “blockwise” ordering of monomials. Each block \mathcal{B}_i ($i = 1, \dots, u$) contains exactly the quadratic monomials appearing in the central polynomials of the i -th, but not the $(i-1)$ -th Rainbow layer. Note that each block \mathcal{B}_i contains exactly $D_i - D_{i-1}$ monomials. Inside the blocks we use the lexicographical

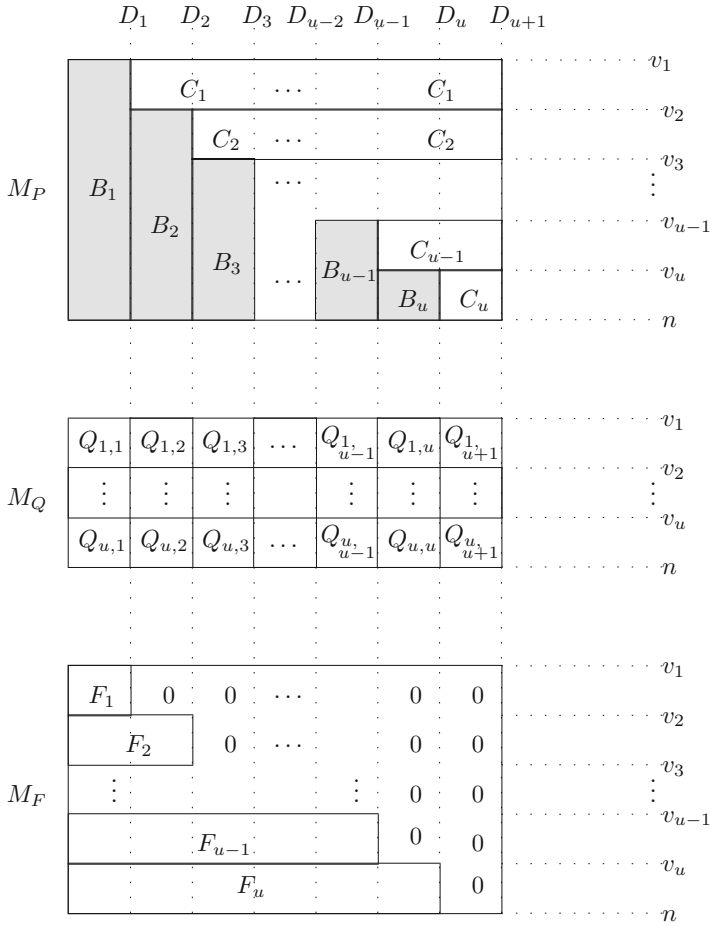


Fig. 5.3 Layout of the matrices M_P , M_Q and M_F

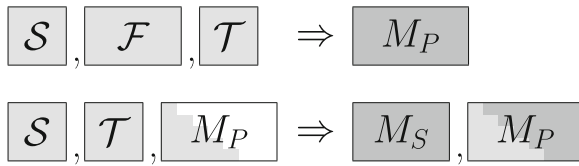


Fig. 5.4 Standard key generation (above) and alternative key generation of the rainbow signature scheme. The light gray parts are chosen by the user, the dark gray parts are computed during the key generation process

ordering. The elements of the transformation matrix $A_{Rb} \in \mathbb{F}^{D_u \times D_{u+1}}$ for cyclic Rainbow are defined as shown in (5.35). However we sort them according to the monomial ordering defined above. Finally we divide the matrix A_{Rb} into blocks

$$A_{Rb} = \begin{pmatrix} A_{1,1} & A_{1,2} & \dots & A_{1,u} & A_{1,u+1} \\ A_{2,1} & \vdots & \dots & \vdots & A_{2,u+1} \\ \vdots & & & & \vdots \\ A_{u,1} & A_{u,2} & \dots & A_{u,u} & A_{u,u+1} \end{pmatrix}.$$

Here, $A_{k,l}$ is a $(D_k - D_{k-1}) \times (D_l - D_{l-1})$ matrix.

Similarly to this, we divide the $m \times m$ -matrix S into u^2 submatrices.

$$S = \begin{pmatrix} S_{1,1} & \dots & S_{1,u} \\ \vdots & & \vdots \\ S_{u,1} & \dots & S_{u,u} \end{pmatrix}$$

Here, $S_{k,l}$ is a $o_k \times o_l$ matrix. While, for StructuredUOV, we required only A_{UOV} to be invertible, we need here some additional conditions.

- (C1) Every submatrix $\begin{pmatrix} S_{i,i} & \dots & S_{i,u} \\ \vdots & & \vdots \\ S_{u,i} & \dots & S_{u,u} \end{pmatrix}$ ($i = 1, \dots, u$) of S must be invertible.
- (C2) Every submatrix $\begin{pmatrix} A_{1,1} & \dots & A_{1,i} \\ \vdots & & \vdots \\ A_{i,1} & \dots & A_{i,i} \end{pmatrix}$ ($i = 1, \dots, u$) of A must be invertible.

Finally, we divide the $m \times D_{u+1}$ Macaulay matrices M_F , M_Q and M_P in submatrices as shown in Fig. 5.3. Algorithm 5.8 shows, how to create a structured public key for the Rainbow signature scheme.

At the beginning, the algorithm assigns values to the entries of the matrices B_1, \dots, B_u (marked gray in Fig. 5.3). Then it assigns random values to the elements of the matrices S and T representing the linear transformations S and \mathcal{T} . It computes the transformation matrix A_{Rb} and checks, if the conditions (C1) and (C2) are fulfilled.⁸ If this is the case, it computes recursively the matrices $Q_{i,j}$ ($i = 1, \dots, u$, $j = 1, \dots, u+1$) and F_1, \dots, F_u of Fig. 5.3. Finally, the algorithm composes the central map \mathcal{F} from the matrices F_1, \dots, F_u and the public key \mathcal{P} from the matrices B_1, \dots, B_u and C_1, \dots, C_u and returns the Rainbow key pair $((S, \mathcal{F}, T), \mathcal{P})$.

⁸Experiments have shown that, for randomly chosen matrices S and T , the conditions are fulfilled with high probability. Therefore, in most cases, we need only one run of the loop in line 1 to 5 of Algorithm 5.8.

Altogether, the technique allows us to fix

$$\sum_{i=1}^u o_i \left(\frac{v_i(v_i + 1)}{2} + v_i o_i \right)$$

coefficients of the public key, which leads to a reduction of the Rainbow public key by up to 60%. Similar to the case of UOV, we can choose these coefficients to be

- partially cyclic (\rightarrow cyclicRainbow)
- generated by a linear feedback shift register (\rightarrow RainbowLFSR)
- generated by a PRNG (\rightarrow RainbowPRNG)

Similar to the case of StructuredUOV, there exists no known attack which uses the additional structure in the public key of cyclicRainbow and its variants. Therefore, it seems that we can use the same values of o and v as for the standard Rainbow scheme.

5.6.3 Key Sizes and Efficiency

The public key size of StructuredRainbow is given as

$$\text{size}_{\text{pk StructuredRainbow}} = m \frac{o_u(o_u + 1)}{2}$$

field elements plus whatever it takes to store the structured part of the public key. For cyclic Rainbow this is D_u , for all other cases we can store the structured part of the public key as a small seed. All in all, the above technique allows us to reduce the public key size of Rainbow by up to 63%.

The size of the StructuredRainbow private key is the same as the private key size of the standard Rainbow scheme, i.e.

$$\text{size}_{\text{sk StructuredRainbow}} = \underbrace{m^2}_{\mathcal{S}} + \underbrace{n^2}_{\mathcal{T}} + \underbrace{\sum_{i=1}^u o_i \left(\frac{v_i(v_i + 1)}{2} + v_i o_i \right)}_{\mathcal{F}}$$

field elements. Note again that, in this section, we restricted to homogeneous maps \mathcal{S} , \mathcal{F} , \mathcal{T} and \mathcal{P} .

Using the technique described in the next section, the key generation time of Structured Rainbow nearly equals the key generation time of the standard Rainbow scheme. Regarding signature generation, both schemes are identical. In the signature verification process, we can make use of the additional structure in the cyclicRainbow or RainbowLFRS public keys to speed up signature verification by up to 50%. (see [15] for details).

Algorithm 5.8 Key generation of StructuredRainbow**Input:** Matrices B_1, \dots, B_u **Output:** Rainbow key pair $((S, \mathcal{F}, \mathcal{T}), \mathcal{P})$

```

1: repeat
2:   Choose randomly invertible linear transformations  $S$  and  $\mathcal{T}$ 
   (represented by matrices  $S \in \mathbb{F}^{m \times m}$  and  $T \in \mathbb{F}^{n \times n}$ ).
3:   Build the transformation matrix  $A$  as shown by Eq. (5.36) and divide it into
   submatrices  $A_{i,j}$  ( $i = 1, \dots, u$ ,  $j = 1, \dots, u + 1$ ) as shown above
4:   Divide  $S$  into submatrices  $S_{i,j}$  ( $i, j = 1, \dots, u$ ).
5: until all the conditions of (C1) and (C2) are fulfilled
6: for  $i = 1$  to  $u$  do
7:   Compute 
$$\begin{pmatrix} Q_{i,i} \\ \vdots \\ Q_{u,i} \end{pmatrix} = \begin{pmatrix} S_{i,i} & \dots & S_{i,u} \\ \vdots & & \vdots \\ S_{u,i} & \dots & S_{u,u} \end{pmatrix}^{-1} \cdot \left( B_i - \sum_{j=1}^{i-1} \begin{pmatrix} S_{i,j} \\ \vdots \\ S_{u,j} \end{pmatrix} \cdot Q_{ji} \right).$$

8:   Compute  $F_i = (Q_{i,1} || \dots || Q_{i,i}) \cdot \begin{pmatrix} A_{11} & \dots & A_{1,i} \\ \vdots & & \vdots \\ A_{i,1} & \dots & A_{i,i} \end{pmatrix}^{-1}.$ 
9:   Compute  $(Q_{i,i+1} || \dots || Q_{i,u+1}) = F_i \cdot \begin{pmatrix} A_{1,i+1} & \dots & A_{1,u+1}^T \\ \vdots & & \vdots \\ A_{i,i+1} & \dots & A_{i,u+1} \end{pmatrix}.$ 
10: end for
11: for  $i = 1$  to  $u$  do
12:   Compute  $C_i = (S_{i,1} || \dots || S_{i,u}) \cdot \begin{pmatrix} Q_{1,i} & \dots & Q_{1,u+1} \\ \vdots & & \vdots \\ Q_{u,i} & \dots & Q_{u,u+1} \end{pmatrix}.$ 
13: end for
14: Compose  $\mathcal{F}$  from  $F_1, \dots, F_u$ .
15: Compose  $\mathcal{P}$  from  $B_1, \dots, B_u, C_1, \dots, C_u$ .
16: return  $((S, \mathcal{F}, \mathcal{T}), \mathcal{P})$ 

```

Table 5.4 shows the resulting key and signature sizes for our three security levels.

5.6.4 LUOV

The LUOV (LiftedUOV) signature scheme was proposed by the research group in Leuven in [2] and later submitted to the NIST standardization process for post-quantum cryptosystems. The basic idea is to choose the coefficients of the public and private key from a small subfield of the field \mathbb{F}_2 , while messages (hash values) and signatures are defined over the larger field \mathbb{F}_{2^r} . The scheme uses the same parameters as the standard UOV signature scheme (see Sect. 5.1). The only difference is that we restrict to fields of even characteristic. The workflow of the scheme can be described as follows:

Table 5.4 Parameters and key sizes of StructuredRainbow

Security category	Parameters (\mathbb{F} , v_1 , o_1 , o_2)	Public key size (kB)	Private key size (kB)	Signature size (bit)
I	(GF(256), 40, 24, 24)	53.1	140.0	704
II	(GF(256), 68, 36, 36)	192.6	525.2	1120
III	(GF(256), 92, 48, 48)	463.8	1244.4	1504

Key Generation To create a key pair of LUOV, a user Alice performs the following steps

- Choose randomly a seed \mathbf{s} of length 256 bit (32 byte), which is used to generate a matrix $T' \in \mathbb{F}_2^{v \times o}$ and a public seed \mathbf{s}_p .
- Define the linear transformation \mathcal{T} using the matrix $T = \begin{pmatrix} 1_{v \times v} & T' \\ 0_{o \times v} & 1_{o \times o} \end{pmatrix}$.
- Use the seed \mathbf{s}_p to create a matrix $B \in \mathbb{F}_2^{o(v(v+1)/2 + ov)}$. View B as the matrix containing the coefficients of the vinegar \times vinegar and vinegar \times oil terms of the public map \mathcal{P} .
- Use the technique described in Sect. 5.6.1 to generate from B and \mathcal{T} the central map \mathcal{F} and the coefficients of the oil \times oil terms of the public polynomials. Store these coefficients as a matrix $Q \in \mathbb{F}_2^{o \times o(o+1)/2}$.

The *private key* of the LUOV signature scheme is just the seed \mathbf{s} , the *public key* consists of the public seed \mathbf{s}_p and the matrix $Q \in \mathbb{F}_2^{o \times o(o+1)/2}$.

Remark 5.14 Despite of the fact that all coefficients of the private and public maps are elements of the subfield GF(2), all three maps \mathcal{F} , \mathcal{T} and \mathcal{P} are viewed as maps over the extension field $\mathbb{F} = \mathbb{F}_{2^r}$. This process is denoted as “Lifting”, hence the name LiftedUOV. Therefore, hash values and signatures are vectors over the extension field \mathbb{F}_{2^r} . This enabled the authors to choose the parameters of their scheme (compared to those of a UOV scheme over GF(2)), which reduces the key sizes significantly. Furthermore, direct attacks have to be performed over the larger field, which is more costly than a direct attack over GF(2).

To reduce the private and public key sizes further, major parts of the keys are stored as small seeds of size 256 bits (in fact, the private key consists only of the seed \mathbf{s} , the public key contains, besides of the seed \mathbf{s}_p , only the matrix Q containing the coefficients of the oil \times oil terms of the public key).

The downside of this extreme key size reduction is that, in the signature generation process, we have to recover the UOV private key $(\mathcal{F}, \mathcal{T})$ out of the seed \mathbf{s} . In the signature verification process, we have to recover the public key \mathcal{P} out of \mathbf{s}_p and Q . Therefore, the key size reduction leads to a slow down of the signature generation and verification processes.

Signature Generation The signature generation process of LUOV consists of two parts. In the first part, we recover the private UOV key $(\mathcal{F}, \mathcal{T})$ out of the seed \mathbf{s} . In

the second part, we use this UOV private key to generate a UOV signature in the standard way.

1. Step: Recover the UOV private key $(\mathcal{F}, \mathcal{T})$ from the seed \mathbf{s} .

- (a) Generate from \mathbf{s} the matrix $T' \in \mathbb{F}_2^{v \times o}$ and the seed \mathbf{s}_p . Define the linear transformation $\mathcal{T} : \mathbb{F}^n \rightarrow \mathbb{F}^n$ as $\mathcal{T} = \begin{pmatrix} 1 & T' \\ 0 & 1 \end{pmatrix}$.
- (b) Generate from \mathbf{s}_p the matrix $B \in \mathbb{F}_2^{o(v+1)/2+ov}$.
- (c) Use the technique described in Sect. 5.6.1 to generate from B and \mathcal{T} the central map \mathcal{F} .

2. Step: Generate a UOV signature for the document d .

- (a) Use a hash function $\mathcal{H} : \{0, 1\}^* \rightarrow \mathbb{F}_{2^r}^o$ to compute a hash value $\mathbf{w} \in \mathbb{F}_{2^r}^o$ of the document d .
- (b) Derive from \mathbf{w} and the seed \mathbf{s} the values of the vinegar variables $x_1, \dots, x_v \in \mathbb{F}_{2^r}$ and substitute them into the central map \mathcal{F} to obtain a linear map in the oil variables (denoted by a matrix $F'(x_{v+1}, \dots, x_n) = \mathbf{c}'$).
- (c) Solve the linear system $F'(x_{v+1}, \dots, x_n) = \mathbf{c}'$ for x_{v+1}, \dots, x_n and set $\mathbf{x} = (x_1, \dots, x_v, x_{v+1}, \dots, x_n) \in \mathbb{F}_{2^r}$.
- (d) Compute the signature $\mathbf{z} = \mathcal{T}^{-1}(\mathbf{x}) \in \mathbb{F}_{2^r}$.

Signature Verification Similar to the signature generation process, the signature verification process of LUOV consists of two parts. In the first part, we generate from the seed \mathbf{s}_p and the matrix Q the UOV public key \mathcal{P} . In the second part, we use the key \mathcal{P} to check the authenticity of a signature $\mathbf{z} \in \mathbb{F}_{2^r}^n$ in the standard way.

- Step: Generate from \mathbf{s}_p the matrix $B \in \mathbb{F}_2^{o(v+1)/2+ov}$. Concatenate B with the matrix Q to get the UOV public key $\mathcal{P} : \mathbb{F}_{2^r}^n \rightarrow \mathbb{F}_{2^r}^o$.
- Step Compute the hash value $\mathbf{w} = \mathcal{H}(d)$ of the document and compute $\mathbf{w}' = \mathcal{P}(\mathbf{z})$. If $\mathbf{w}' = \mathbf{w}$ holds, accept the signature, otherwise reject.

Regarding the details on how the matrices T' and B as well as the vinegar variables are derived from the seeds \mathbf{s} and \mathbf{s}_p , we refer the interested reader to [2].

5.6.5 Security

Since the LUOV scheme can be seen as a special instance of the UOV signature scheme, all known attacks against UOV apply against LUOV, too. However, since major parts of the private coefficients of LUOV are chosen from $\text{GF}(2)$, we have to recompute the complexities of these attacks.

- direct attack: Since both the hash value and signature are defined over $\text{GF}(2^r)$, the direct attack behaves like a direct attack against a standard UOV scheme over $\text{GF}(2^r)$.

- UOV attack: Against LUOV, the UOV attack can be performed over the field $\text{GF}(2)$ (since it only uses the public key). Therefore, we have to choose the parameters of LUOV in a way that

$$2^{v-o+1} o^4 > 2^k,$$

where k is the security parameter.

- UOV Reconciliation attack: Like the UOV attack, the UOV Reconciliation attack depends only on the public key. Therefore we can perform this attack over the field $\text{GF}(2)$. We therefore have to choose the number o of oil variables in such a way that a Gröbner basis attack against a determined system of o equations is infeasible.

To summarize, in order to get the same security, the parameters o and v of LUOV have to be chosen significantly larger than for a standard UOV scheme over $\text{GF}(2^r)$. However, since major parts of the public and private key can be stored as small seeds, we still get a significant reduction of the key sizes (see Table 5.5).

Recently, Ding et al. [7] presented a new attack on the LUOV scheme. The method is called the Subfield Differential Attack (SDA). This attack does not rely on the Oil and Vinegar structure of LUOV but merely on the fact that the coefficients of the quadratic terms are contained in a small subfield, which is the basis for the design of LUOV. It is argued heuristically that there is a high probability that there exists a way to find a solution using differentials from a small subfield.

More precisely, the problem of finding a signature to a problem is reduced to solving a set of underdetermined quadratic equations over the subfield. The complexity of such an attack is much reduced. For each proposed set of parameters in the LUOV submission, Ding et al. showed that the attack will make it impossible for LUOV to fulfill the requirements of the NIST security levels. Furthermore they show that UOV and Rainbow (both the standard and cyclic versions) are unaffected by this attack. The new attack method casts serious doubt if the basic idea of LUOV is sound.

5.6.6 Key Sizes and Efficiency

The public key size of LUOV is given as

$$\text{size}_{\text{pk LUOV}} = \underbrace{o^2 \frac{(o+1)}{2}}_{\text{matrix Q}} + \underbrace{256}_{\text{seeds}} \text{ bits},$$

the size of the private key is

$$\text{size}_{\text{sk LUOV}} = 256 \text{ bits}.$$

Table 5.5 Parameters and key sizes of LUOV

Security category	Parameters (\mathbb{F} , v , o)	Public key size (kB)	Private key size (byte)	Signature size (bit)
Ia	(GF(256), 256, 63)	15.5	32	2552
Ila	(GF(256), 351, 90)	45	32	3528
III	(GF(256), 404, 117)	98.6	32	4168

Since the parameters are significantly larger, the key generation takes longer than for the standard UOV scheme. During the signature generation process, we have to recover the private key from the seed s , which leads to a drastic slow down of the signature generation. The same can be said for signature verification, though the slow down factor is not as large as for signature generation.

Table 5.5 shows practical parameters for the LUOV scheme (without consideration of the new attack mentioned above).

5.7 Efficient Key Generation of Rainbow

While the signature generation and verification processes of Rainbow are very fast, the straightforward key generation process as presented in Sect. 5.4 is far too costly. In this section we describe a much more efficient way to generate a Rainbow key pair. To simplify our description, we restrict to Rainbow schemes with two layers.

Note that this is the standard setup for implementing Rainbow. Furthermore, we restrict to homogeneous maps \mathcal{S} , \mathcal{F} and \mathcal{T} . Therefore, the public key $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$ will be a homogeneous quadratic map, too. Note that, by this restriction, the security of our Rainbow instance is not weakened.

Remember from Sect. 5.5 that, for every Rainbow public key \mathcal{P} , there exists a Rainbow private key $(\mathcal{S}, \mathcal{F}, \mathcal{T})$ with linear maps \mathcal{S} and \mathcal{T} of the form

$$\mathcal{S} = \begin{pmatrix} I_{o_1 \times o_1} & S'_{o_1 \times o_2} \\ 0_{o_2 \times o_1} & I_{o_2 \times o_2} \end{pmatrix}, \quad \mathcal{T} = \begin{pmatrix} I_{v_1 \times v_1} & T_{v_1 \times o_1}^{(1)} & T_{v_1 \times o_2}^{(2)} \\ 0_{o_1 \times v_1} & I_{o_1 \times o_1} & T_{o_1 \times o_2}^{(3)} \\ 0_{o_2 \times v_1} & 0_{o_2 \times o_1} & I_{o_2 \times o_2} \end{pmatrix}. \quad (5.40)$$

Therefore, without weakening the security of the scheme, we can assume that the maps \mathcal{S} and \mathcal{T} of our Rainbow instance are of the form (5.40).

For our special choice of \mathcal{S} and \mathcal{T} we have $\det(\mathcal{S}) = \det(\mathcal{T}) = 1$ and (for fields of characteristic 2)

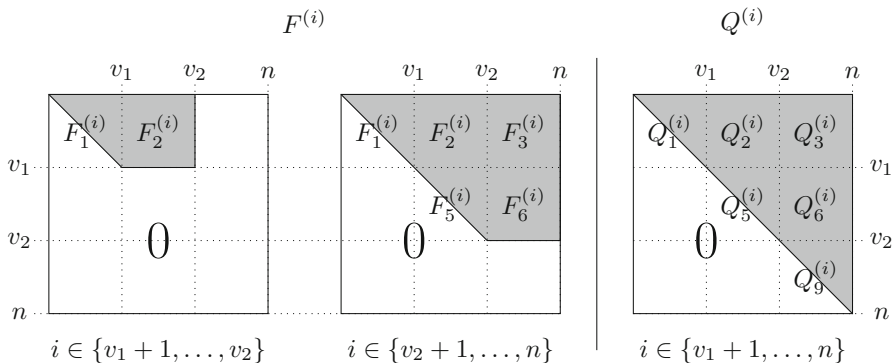


Fig. 5.5 Matrices $F^{(i)}$ (left) and $Q^{(i)}$ (right) representing the polynomials of the Rainbow central and intermediate maps. The only non-zero elements are contained in the gray spaces

$$\mathcal{S}^{-1} = \begin{pmatrix} I_{o_1 \times o_1} & S'_{o_1 \times o_2} \\ 0_{o_2 \times o_1} & I_{o_2 \times o_2} \end{pmatrix} = \mathcal{S}, \quad \mathcal{T}^{-1} = \begin{pmatrix} I & T^{(1)} & T^{(1)} \cdot T^{(3)} + T^{(2)} \\ 0 & I & T^{(3)} \\ 0 & 0 & I \end{pmatrix}. \quad (5.41)$$

For abbreviation, we set $T^{(4)} := T^{(1)} \cdot T^{(3)} + T^{(2)}$.

We introduce an intermediate map $\mathcal{Q} = \mathcal{F} \circ \mathcal{T}$. Note that we can write the components of the maps \mathcal{F} and \mathcal{Q} as quadratic forms

$$\begin{aligned} f^{(i)}(\mathbf{x}) &= \mathbf{x}^T \cdot F^{(i)} \cdot \mathbf{x} \\ q^{(i)}(\mathbf{x}) &= \mathbf{x}^T \cdot Q^{(i)} \cdot \mathbf{x} \end{aligned} \quad (5.42)$$

with upper triangular matrices $F^{(i)}$ and $Q^{(i)}$ ($i = v_1 + 1, \dots, n$). Note that, due to the relation $\mathcal{Q} = \mathcal{F} \circ \mathcal{T}$, we get

$$Q^{(i)} = T^T \cdot F^{(i)} \cdot T \quad (i = v_1 + 1, \dots, n). \quad (5.43)$$

Note further that, due to the special form of the Rainbow central map, the matrices $F^{(i)}$ look as shown in Fig. 5.5. The matrices $Q^{(i)}$ are divided into submatrices $Q_1^{(i)}, \dots, Q_9^{(i)}$ analogously.

In order to generate a Rainbow key pair, we choose the non-zero elements of the matrices \mathcal{S} , \mathcal{T} and $F^{(v_1+1)}, \dots, F^{(n)}$ uniformly at random from the field \mathbb{F} and perform the following three steps.

5.7.1 First Step: Compute the Matrices $Q^{(i)}$ of the First Layer

In the first step, we compute from the matrices $F^{(v_1+1)}, \dots, F^{(v_2)}$ the matrices $Q^{(v_1+1)}, \dots, Q^{(v_2)}$. Since the only non-zero elements of the matrices $F^{(i)}$ ($i = v_1 + 1, \dots, v_2$) are contained in the matrices $F_1^{(i)}$ and $F_2^{(i)}$, we obtain from $Q^{(i)} = T^T \cdot F^{(i)} \cdot T$

$$\begin{aligned}
 Q_1^{(i)} &= F_1^{(i)}, \\
 Q_2^{(i)} &= (F_1^{(i)} + (F_1^{(i)})^T) \cdot T_1 + F_2^{(i)}, \\
 Q_3^{(i)} &= (F_1^{(i)} + (F_1^{(i)})^T) \cdot T_2 + F_2^{(i)} \cdot T_3, \\
 Q_5^{(i)} &= \text{UT}(T_1^T \cdot F_1^{(i)} \cdot T_1 + T_1^T \cdot F_2^{(i)}), \\
 Q_6^{(i)} &= T_1^T (F_1^{(i)} + (F_1^{(i)})^T) \cdot T_2 + T_1^T \cdot F_2^{(i)} \cdot T_3 + (F_2^{(i)})^T \cdot T_2, \\
 Q_9^{(i)} &= \text{UT}(T_2^T \cdot F_1^{(i)} \cdot T_2 + T_2^T \cdot F_2^{(i)} \cdot T_3).
 \end{aligned} \tag{5.44}$$

Here, $\text{UT}(A)$ transforms a matrix A into an equivalent upper triangular matrix (i.e. $a_{ij} = a_{ij} + a_{ji}$ for $i < j$, $a_{ij} = 0$ for $i > j$).

5.7.2 Second Step: Compute the Matrices $Q^{(i)}$ of the Second Layer

In the second step, we compute from the matrices $F^{(v_2+1)}, \dots, F^{(n)}$ the matrices $Q^{(v_2+1)}, \dots, Q^{(n)}$. Since the only non-zero elements of the matrices $F^{(i)}$ ($i = v_2 + 1, \dots, n$) are contained in the submatrices $F_1^{(i)}, F_2^{(i)}, F_3^{(i)}, F_5^{(i)}$ and $F_6^{(i)}$, we obtain from $Q^{(i)} = T^T \cdot F^{(i)} \cdot T$

$$\begin{aligned}
 Q_1^{(i)} &= F_1^{(i)}, \\
 Q_2^{(i)} &= (F_1^{(i)} + (F_1^{(i)})^T) \cdot T_1 + F_2^{(i)}, \\
 Q_3^{(i)} &= (F_1^{(i)} + (F_1^{(i)})^T) \cdot T_2 + F_2^{(i)} \cdot T_3 + F_3^{(i)}, \\
 Q_5^{(i)} &= \text{UT}(T_1^T \cdot F_1^{(i)} \cdot T_1 + T_1^T \cdot F_2^{(i)} + F_5^{(i)}), \\
 Q_6^{(i)} &= T_1^T \cdot (F_1^{(i)} + (F_1^{(i)})^T) \cdot T_2 + T_1^T \cdot F_2^{(i)} \cdot T_3 \\
 &\quad + T_1^T \cdot F_3^{(i)} + (F_2^{(i)})^T \cdot T_2 + (F_5^{(i)} + (F_5^{(i)})^T) \cdot T_3 + F_6^{(i)}, \\
 Q_9^{(i)} &= \text{UT}(T_2^T \cdot F_1^{(i)} \cdot T_2 + T_2^T \cdot F_2^{(i)} \cdot T_3 + T_3^T \cdot F_5^{(i)} \cdot T_3 + T_2^T \cdot F_3^{(i)} + T_3^T \cdot F_6^{(i)}).
 \end{aligned} \tag{5.45}$$

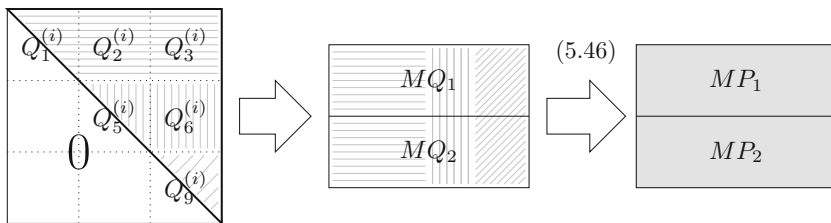


Fig. 5.6 Computing the public key

Here, again, $UT(A)$ transforms the matrix A into an equivalent upper triangular matrix.

5.7.3 Third Step: Compute the Public Key

In the third step, we compute from the matrices $Q^{(i)}$ ($i = v_1 + 1, \dots, n$) the public key \mathcal{P} of the scheme. To do this, we first transform the matrices $Q^{(i)}$ into a Macaulay matrix MQ . For $i = v_1 + 1, \dots, v_2$, we copy the entries of the matrix $Q^{(i)}$ into the $(i - v_1)$ -th row of the matrix MQ_1 (from left to right and top to bottom). Similarly, we copy the elements of the matrices $Q^{(i)}$ of the second layer into the matrix MQ_2 (see Fig. 5.6). After this, we compute the Macaulay matrix MP of the public key as $MP = S \cdot MQ$ or

$$\begin{aligned} MP_1 &= MQ_1 + S' \cdot MQ_2 \\ MP_2 &= MQ_2. \end{aligned} \tag{5.46}$$

This process is illustrated in Fig. 5.6.

Algorithm 5.9 shows our key generation algorithm for the standard Rainbow scheme in compact form.

5.7.4 Efficient Key Generation for StructuredRainbow

As in the previous section, we restrict here to Rainbow schemes with two layers and homogeneous maps \mathcal{S} , \mathcal{F} , \mathcal{T} and \mathcal{P} . As above, we choose the matrices \mathcal{S} and \mathcal{T} of form (5.40). Furthermore, we assign arbitrary values to the entries of the three matrices B_1 , B_2 and B_3 of Fig. 5.7. Note that, in this section, we assume that the monomials of \mathcal{F} , \mathcal{Q} and \mathcal{P} are ordered according to the monomial order defined in Sect. 5.6.2.

Algorithm 5.9 Efficient key generation of the standard rainbow signature scheme

Input: linear transformations \mathcal{S}, \mathcal{T} of form (5.40), Rainbow central map \mathcal{F} (given as matrices $F^{(i)}$ ($i = v_1 + 1, \dots, n$); see Fig. 5.5)

Output: Rainbow public key \mathcal{P} (consisting of the matrices MP_1 and MP_2)

```

1: for  $i = v_1 + 1$  to  $v_2$  do
2:   Compute the matrices  $Q_1^{(i)}, Q_2^{(i)}, Q_3^{(i)}, Q_5^{(i)}, Q_6^{(i)}, Q_9^{(i)}$  using (5.44).
3: end for
4: for  $i = v_2 + 1$  to  $n$  do
5:   Compute  $Q_1^{(i)}, Q_2^{(i)}, Q_3^{(i)}, Q_5^{(i)}, Q_6^{(i)}, Q_9^{(i)}$  using (5.45).
6: end for
7: for  $i = 1$  to  $m$  do
8:   Insert the elements of the matrix  $Q^{(i)}$  into the  $(i - v_1)$ -th row of the
      matrices  $MQ_1$  and  $MQ_2$  (as described above)
9: end for
10: Compute the Rainbow public key using (5.46).
11: return  $MP_1, MP_2$ .
```

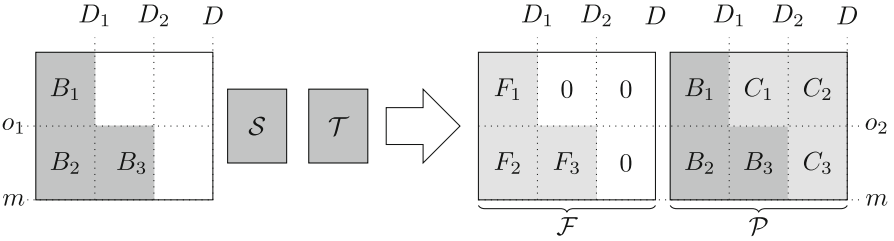


Fig. 5.7 The 2-layer StructuredRainbow Signature Scheme. The dark gray parts are chosen by the user, while the light gray parts are computed from them

The key generation process of StructuredRainbow computes from these matrices the matrices F_1, F_2, F_3 and C_1, C_2, C_3 of Fig. 5.7. To do this, it performs the following four steps.

5.7.4.1 First Step: Compute the Matrices $MQ_{1,1}$, $MQ_{2,1}$ and $MQ_{2,2}$

Similarly to the Macaulay matrices of \mathcal{F} and \mathcal{P} of Fig. 5.7, we divide the Macaulay matrix MQ of the intermediate map \mathcal{Q} into six submatrices $MQ_{1,1}, MQ_{1,2}, MQ_{1,3}, MQ_{2,1}, MQ_{2,2}$ and $MQ_{3,2}$. Due to the relation $\mathcal{P} = \mathcal{S} \circ \mathcal{Q}$ we find

$$\begin{aligned}
 MQ_{1,1} &= B_1 + S' \cdot B_2, \\
 MQ_{2,1} &= B_2, \\
 MQ_{2,2} &= B_2.
 \end{aligned} \tag{5.47}$$

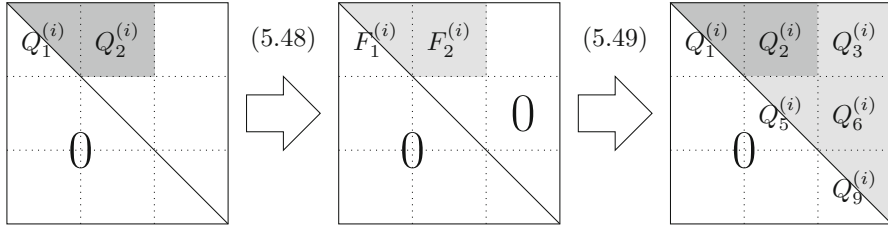


Fig. 5.8 Computing the central polynomials of the first layer

5.7.4.2 Second Step: Compute the Central Polynomials of the First Rainbow Layer

For this, we represent the first o_1 components of the map \mathcal{Q} as upper triangular matrices $Q^{(i)}$.

We insert the D_1 elements of the i -th row of $MQ_{1,1}$ into the dark gray parts of the matrices $Q_1^{(i)}$ and $Q_2^{(i)}$ (from left to right and top to bottom; see Fig. 5.8 (left)). The corresponding matrices $F^{(i)}$ representing the i -th central polynomial look as shown in Fig. 5.8 (middle). Note that the only non-zero elements are located in the submatrices $F_1^{(i)}$ and $F_2^{(i)}$. Due to the relation $F^{(i)} = (T^{-1})^T \cdot Q^{(i)} \cdot T^{-1}$ we get

$$\begin{aligned} F_1^{(i)} &= Q_1^{(i)}, \\ F_2^{(i)} &= (Q_1^{(i)} + (Q_1^{(i)})^T) \cdot T_1 + Q_2^{(i)}. \end{aligned} \quad (5.48)$$

All the other elements of the matrices $F^{(i)}$ ($i \in \{1, \dots, o_1\}$) are zero. So, after having determined the elements of $F_1^{(i)}$ and $F_2^{(i)}$, we can use the inverse relation $Q^{(i)} = T^T \cdot F^{(i)} \cdot T$ to compute the light gray parts of $Q^{(i)}$. We find

$$\begin{aligned} Q_3^{(i)} &= (F_1^{(i)} + (F_1^{(i)})^T) \cdot T_2 + F_2 \cdot T_3, \\ Q_5^{(i)} &= \text{UT}(T_1^T \cdot F_1^{(i)} \cdot T_1 + T_1^T \cdot F_2^{(i)}), \\ Q_6^{(i)} &= T_1^T (F_1^{(i)} + (F_1^{(i)})^T) \cdot T_2 + T_1^T \cdot F_2^{(i)} \cdot T_3 + (F_2^{(i)})^T \cdot T_2, \\ Q_9^{(i)} &= \text{UT}(T_2^T \cdot F_1^{(i)} \cdot T_2 + T_2^T \cdot F_2^{(i)} \cdot T_3). \end{aligned} \quad (5.49)$$

(see Fig. 5.8 (right)).

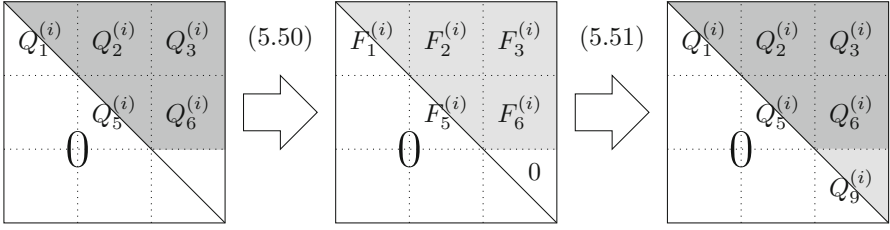


Fig. 5.9 Computing the central polynomials of the second layer

5.7.4.3 Third Step: Compute the Central Polynomials of the Second Rainbow Layer

For this, we insert the D_1 elements of the i -th row of $MQ_{2,1}$ into the dark gray parts of the matrices $Q_1^{(i)}$ and $Q_2^{(i)}$ (from left to right and top to bottom). The $D_2 - D_1$ elements of the i -th row of the matrix $MQ_{2,2}$ are inserted into the dark gray parts of the matrices $Q_3^{(i)}$, $Q_5^{(i)}$ and $Q_6^{(i)}$ (again left to right and top to bottom; i.e. we fill the matrix $Q_3^{(i)}$ first). Therefore, the matrices $Q^{(i)}$ look as shown in Fig. 5.9 (left).

Due to the relation $F^{(i)} = (T^{-1})^T \cdot Q^{(i)} \cdot T^{-1}$ we can compute the non zero parts of the matrices $F^{(i)}$ ($i = v_2 + 1, \dots, n$) as

$$\begin{aligned}
 F_1^{(i)} &= Q_1^{(i)}, \\
 F_2^{(i)} &= (Q_1^{(i)} + (Q_1^{(i)})^T) \cdot T_1 + Q_2^{(i)}, \\
 F_3^{(i)} &= (Q_1^{(i)} + (Q_1^{(i)})^T) \cdot T_4 + Q_2^{(i)} \cdot T_3 + Q_3^{(i)}, \\
 F_5^{(i)} &= UT(T_1^T \cdot Q_1^{(i)} \cdot T_1 + T_1^T \cdot Q_2^{(i)} + Q_5^{(i)}), \\
 F_6^{(i)} &= T_1^T \cdot (Q_1^{(i)} + (Q_1^{(i)})^T) \cdot T_4 + T_1^T \cdot Q_2^{(i)} \cdot T_3 \\
 &\quad + T_1^T \cdot Q_3^{(i)} + (Q_2^{(i)})^T \cdot T_4 + (Q_5^{(i)} + (Q_5^{(i)})^T) \cdot T_3 + Q_6^{(i)}. \quad (5.50)
 \end{aligned}$$

After this, we can use the inverse relation $Q^{(i)} = T^T \cdot F^{(i)} \cdot T$ to compute the matrices $Q_9^{(i)}$. We get

$$Q_9^{(i)} = UT(T_2^T \cdot F_1^{(i)} \cdot T_2 + T_2^T \cdot F_2^{(i)} \cdot T_3 + T_3^T \cdot F_5^{(i)} \cdot T_3 + T_2^T \cdot F_3^{(i)} + T_3^T \cdot F_6^{(i)}). \quad (5.51)$$

5.7.4.4 Fourth Step: Compute the Public Key

For this last step, we transform the matrices $Q^{(i)}$ ($i = v_1 + 1, \dots, n$) back into a Macaulay matrix MQ . This is done as shown in Fig. 5.10. For $i = v_1, \dots, n$, we perform the following 4 steps

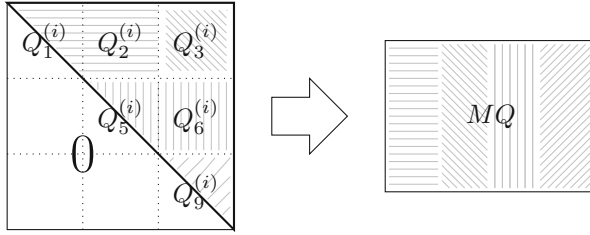


Fig. 5.10 Building the matrix MQ of StructuredRainbow

Table 5.6 Running times of our efficient key generation algorithms of the standard and the structured rainbow signature schemes on an intel xeon @ 3.6 GHz (Skylake) using AVX2 vector instructions

Security category	I/Ia		II/IIa		III/IIIa	
Parameter set	(GF(16), 32, 32, 32)		(GF(256), 68, 36, 36)		(GF(256), 92, 48, 48)	
	Standard	Structured	Standard	Structured	Standard	Structured
Mcycles	8.29	9.28	94.8	110	126	137
Time (ms)	2.30	2.58	26.3	30.5	34.9	38.0
Memory (MB)	3.5	3.5	4.6	4.6	7.0	7.0

- First, we write the D_1 elements of the submatrix $(Q_1^{(i)} || Q_2^{(i)})$ into the $(i - v_1)$ -th row of the matrix MQ (from left to right and top to bottom).
- The following $v_1 o_2$ columns of the $(i - v_1)$ -th row of the matrix MQ are filled with the elements of the matrix $Q_3^{(i)}$. Again, these are read from left to right and top to bottom.
- We continue with the elements of the submatrix $(Q_5^{(i)} || Q_6^{(i)})$.
- The last $D_3 - D_2$ columns of the $(i - v_1)$ -row are filled with the entries of the matrix $Q_9^{(i)}$ (again from left to right and top to bottom).

We divide the matrix MQ into submatrices as described above.

Finally, we compute the matrix MP by $MP = S \cdot MQ$ or, with the special form of our matrix S ,

$$\begin{aligned}
 B_1 &= MQ_{1,2} + S' \cdot MQ_{2,2}, \\
 B_2 &= MQ_{1,3} + S' \cdot MQ_{2,3}, \\
 B_3 &= MQ_{2,3}.
 \end{aligned} \tag{5.52}$$

Note that the coefficients in MP are ordered according to the special monomial order defined above.

Algorithm 5.10 presents this key generation algorithm in compact form. Table 5.6 shows the running time of our key generation algorithms for the standard and the StructuredRainbow signature scheme.

Algorithm 5.10 Efficient Key Generation of the StructuredRainbow Signature Scheme

Input: linear transformations \mathcal{S}, \mathcal{T} of form (5.40), matrices $B_1 \in \mathbb{F}^{o_1 \times D_1}$, $B_2 \in \mathbb{F}^{o_2 \times (D_1)}$ and $B_3 \in \mathbb{F}^{o_2 \times (D_2 - D_1)}$.

Output: Rainbow central map \mathcal{F} , matrices C_1, C_2, C_3 (see Fig. 5.7).

- 1: Compute the matrices $MQ_{1,1}$, $MQ_{2,1}$ and $MQ_{2,2}$ using (5.48)
 - 2: **for** $i = 1$ to o_1 **do**
 - 3: Insert the coefficients of the i -th row of the matrix $MQ_{1,1}$ into the submatrices $Q_1^{(i)}$ and $Q_2^{(i)}$.
 - 4: Set $F_1^{(i)} = Q_1^{(i)}$ and $F_2^{(i)} = (Q_1^{(i)} + (Q_1^{(i)})^T) \cdot T_1 + Q_2^{(i)}$.
 - 5: Compute the matrices $Q_3^{(i)}$, $Q_5^{(i)}$, $Q_6^{(i)}$, $Q_9^{(i)}$ using (5.49).
 - 6: **end for**
 - 7: **for** $i = o_1 + 1$ to m **do**
 - 8: Insert the coefficients of the i -th row of the matrix $MQ_{2,1}$ into the submatrices $Q_1^{(i)}$ and $Q_2^{(i)}$.
 - 9: Insert the coefficients of the i -th row of the matrix $MQ_{2,2}$ into the submatrices $Q_3^{(i)}$, $Q_5^{(i)}$ and $Q_6^{(i)}$.
 - 10: Compute $F_1^{(i)}$, $F_2^{(i)}$, $F_3^{(i)}$, $F_5^{(i)}$, $F_6^{(i)}$ using (5.50).
 - 11: Compute $Q_9^{(i)}$ using (5.51).
 - 12: **end for**
 - 13: **for** $i = 1$ to m **do**
 - 14: Insert the elements of the matrix $Q^{(i)}$ into the i -th row of the matrix MQ (as described above)
 - 15: **end for**
 - 16: Compute the remaining parts of the public key by (5.52).
 - 17: **return** $F^{(1)}, \dots, F^{(m)}, C_1, C_2, C_3$.
-

Note that the algorithms presented in this section can also be used to speed up the key generation process of (Structured)UOV.

References

1. L. Bettale, J.-C. Faugère, L. Perret, Hybrid approach for solving multivariate systems over finite fields. *J. Math. Cryptology* **3**, 177–197 (2009)
2. W. Beullens, B. Preneel, Field lifting for smaller UOV public keys, in *International Conference on Cryptology in India (INDOCRYPT 2017)*. Lecture Notes in Computer Science, vol. 10698 (Springer, Berlin, 2017), pp. 227–246
3. O. Billet, H. Gilbert, Cryptanalysis of rainbow, in *International Conference on Security and Cryptography for Networks (SCN 2006)*. Lecture Notes in Computer Science, vol. 4116 (Springer, Berlin, 2006), pp. 336–347
4. D. Coppersmith, J. Stern, S. Vaudenay, Attacks on the birational signature scheme, in *Annual International Cryptology Conference (CRYPTO 1994)*. Lecture Notes in Computer Science, vol. 773 (Springer, Berlin, 1994), pp. 435–443
5. J. Ding, D. Schmidt, Rainbow, a new multivariable polynomial signature scheme, in *International Conference on Applied Cryptography and Network Security (ACNS 2005)*. Lecture Notes in Computer Science, vol. 3531 (Springer, Berlin, 2005), pp. 164–175

6. J. Ding, B.-Y. Yang, C.-H.O. Chen, M.-S. Chen, C.-M. Cheng, New differential-algebraic attacks and reparametrization of rainbow, in *International Conference on Applied Cryptography and Network Security (ACNS)*. Lecture Notes in Computer Science, vol. 5037 (Springer, Berlin, 2008), pp. 242–257
7. J. Ding, Z. Zhang, J. Deaton, K. Schmidt, F. Vishakha, New attacks on lifted unbalanced oil vinegar schemes, in *The Second NIST PQC Standardization Conference, 2019*. <https://csrc.nist.gov/Events/2019/second-pqc-standardization-conference>
8. L. Goubin, N. Courtois, Cryptanalysis of the TTM cryptosystem, in *International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2000)*. Lecture Notes in Computer Science, vol. 1976 (Springer, Berlin, 2000), pp. 44–57
9. A. Kipnis, J.-J. Patarin, L. Goubin, Unbalanced oil and vinegar schemes, in *International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 1999)*. Lecture Notes in Computer Science, vol. 1592 (Springer, Berlin, 1999), pp. 206–222
10. A. Kipnis, A. Shamir, Cryptanalysis of the oil and vinegar signature scheme, in *Annual International Cryptology Conference (CRYPTO 1998)*. Lecture Notes in Computer Science, vol. 1462 (Springer, Berlin, 1998), pp. 257–266
11. J. Patarin, The oil and vinegar signature scheme, in *Presented at the Dagstuhl Workshop on Cryptography* (1997)
12. A. Petzoldt, Efficient key generation for rainbow, in *International Conference on Post-Quantum Cryptography (PQCrypto 2020)* (2020)
13. A. Petzoldt, S. Bulygin, J.A. Buchmann, CyclicRainbow - a multivariate signature scheme with a partially cyclic public key, in *International Conference on Cryptology in India (INDOCRYPT 2010)*. Lecture Notes in Computer Science, vol. 6498 (Springer, Berlin, 2010), pp. 33–48
14. A. Petzoldt, S. Bulygin, J.A. Buchmann, Linear recurring sequences for the UOV key generation, in *International Conference on Information Security and Cryptology (PKC 2011)*. Lecture Notes in Computer Science, vol. 6571 (Springer, Berlin, 2011), pp. 335–350
15. A. Petzoldt, S. Bulygin, J.A. Buchmann, Fast verification for improved versions of the UOV and rainbow signature schemes, in *International Workshop on Post-Quantum Cryptography (PQCrypto 2013)*. Lecture Notes in Computer Science, vol. 7932 (Springer, Berlin, 2013), pp. 188–202
16. C. Wolf, B. Preneel, Equivalent keys in HFE, C^* , and variations, in *International Conference on Cryptology in Malaysia (MyCrypt 2005)*. Lecture Notes in Computer Science, vol. 3715 (Springer, Berlin, 2005), pp. 33–49

Chapter 6

MQDSS



Abstract In this chapter we introduce the MQDSS signature scheme, which is one of the few provably secure multivariate public key cryptosystems. We start by a description of the MQ based identification scheme which allows a prover to identify himself using a zero knowledge proof based on the knowledge of the solution of a random system. We then describe the Fiat-Shamir construction of transforming an identification to a signature scheme and finally present the MQDSS signature scheme.

In [1], Chen et al. proposed a new multivariate signature scheme called MQDSS (Multivariate Quadratic Digital Signature Scheme). In contrast to the multivariate public key schemes discussed so far, the security of MQDSS is based solely on the MQ Problem of solving a system of multivariate quadratic equations, which makes the MQDSS signature scheme provably secure. However, the provable security of MQDSS comes with the price of relatively large signatures.

The MQDSS signature scheme is based on the MQ based identification scheme of Sakumoto et al. [4]. Using the Fiat-Shamir transformation [2], it is possible to convert this identification scheme into a signature scheme.

We start this chapter by describing the MQ based identification scheme of Sakumoto et al. in Sect. 6.1. In Sect. 6.2, we then introduce the Fiat-Shamir transformation to convert an identification scheme into a signature scheme. Finally, in Sect. 6.3, we describe the MQDSS signature scheme.

6.1 The MQ Based Identification Scheme

In [4], Sakumoto et al. presented a new identification scheme whose security is based solely on the hardness of the MQ Problem. It is therefore one of the very few provable secure multivariate public key cryptosystems. The scheme exists both as a 3-pass and a 5-pass version.

The scheme uses a multivariate quadratic system $\mathcal{P} : \mathbb{F}^n \rightarrow \mathbb{F}^m$ with randomly chosen coefficients which can be seen as a system parameter and is fixed for a large number of users.¹ Every user chooses a random vector $\mathbf{s} \in \mathbb{F}^n$ as his *private key* and computes his *public key* as $\mathbf{v} = \mathcal{P}(\mathbf{s}) \in \mathbb{F}^m$.

To identify himself to a verifier, a user (called *prover*) has to show that he knows a solution \mathbf{s} of the quadratic system $\mathcal{P}(\mathbf{x}) = \mathbf{v}$ without revealing any information about \mathbf{s} . This is done using a so called zero-knowledge proof.

To create a zero-knowledge proof of knowledge for the vector \mathbf{s} , we need the so called *polar form* of the multivariate system \mathcal{P} , which is defined as

$$\mathcal{G}(\mathbf{x}, \mathbf{y}) = \mathcal{P}(\mathbf{x} + \mathbf{y}) - \mathcal{P}(\mathbf{x}) - \mathcal{P}(\mathbf{y}). \quad (6.1)$$

Note that $\mathcal{G}(\mathbf{x}, \mathbf{y})$ is bilinear in \mathbf{x} and \mathbf{y} .

Remark 6.1 In general, the polar form of a multivariate system \mathcal{P} is defined as $\mathcal{G}(\mathbf{x}, \mathbf{y}) = \mathcal{P}(\mathbf{x} + \mathbf{y}) - \mathcal{P}(\mathbf{x}) - \mathcal{P}(\mathbf{y}) + \mathcal{P}(\mathbf{0})$ (c.f. Chap. 2). But, since we assume the system \mathcal{P} to have no constant terms, the term $\mathcal{P}(\mathbf{0})$ vanishes.

The basic observation of [4] is the following: The knowledge of \mathbf{s} is equivalent to knowing a tuple $(\mathbf{r}_0, \mathbf{r}_1, \mathbf{t}_0, \mathbf{t}_1, \mathbf{e}_0, \mathbf{e}_1)$ with $\mathbf{r}_0, \mathbf{r}_1, \mathbf{t}_0, \mathbf{t}_1 \in \mathbb{F}^n$ and $\mathbf{e}_0, \mathbf{e}_1 \in \mathbb{F}^m$ satisfying

$$\mathcal{G}(\mathbf{t}_0, \mathbf{r}_1) + \mathbf{e}_0 = \mathbf{v} - \mathcal{P}(\mathbf{r}_1) - \mathcal{G}(\mathbf{t}_1, \mathbf{r}_1) - \mathbf{e}_1 \text{ and} \quad (6.2)$$

$$(\mathbf{t}_0, \mathbf{e}_0) = (\mathbf{r}_0 - \mathbf{t}_1, \mathcal{P}(\mathbf{r}_0) - \mathbf{e}_1). \quad (6.3)$$

Under the assumption that there exists a computationally binding and statistically hiding commitment scheme *Com*,² the authors of [4] used this observation to create a zero-knowledge proof of knowledge of a solution of the system $\mathcal{P}(\mathbf{x}) = \mathbf{v}$. This proof can be used by a *prover* \mathcal{P} to identify himself to a *verifier* \mathcal{V} using an interactive protocol. In the paper [4], the authors proposed both a 3- and a 5- pass version of the resulting identification scheme. The workflow of these two schemes is shown in Figs. 6.1 and 6.2.

The identification protocols as shown in Figs. 6.1 and 6.2 are not error free, i.e. there is a non negligible probability that an attacker, who is not the owner of the private key \mathbf{s} , can pass one round of the protocol (soundness error). In case of the 3-pass version, this soundness error is $\frac{2}{3}$, for the 5-pass version it is $\frac{1}{2} + \frac{1}{2q}$, where q is the cardinality of the underlying field. To make the impersonation probability reasonably small, it is therefore necessary to perform several rounds of the protocol.

¹To simplify the description of the scheme, we assume that the system \mathcal{P} does not contain constant terms.

²In practice this is realized by a collision- and pre-image resistant hash function.

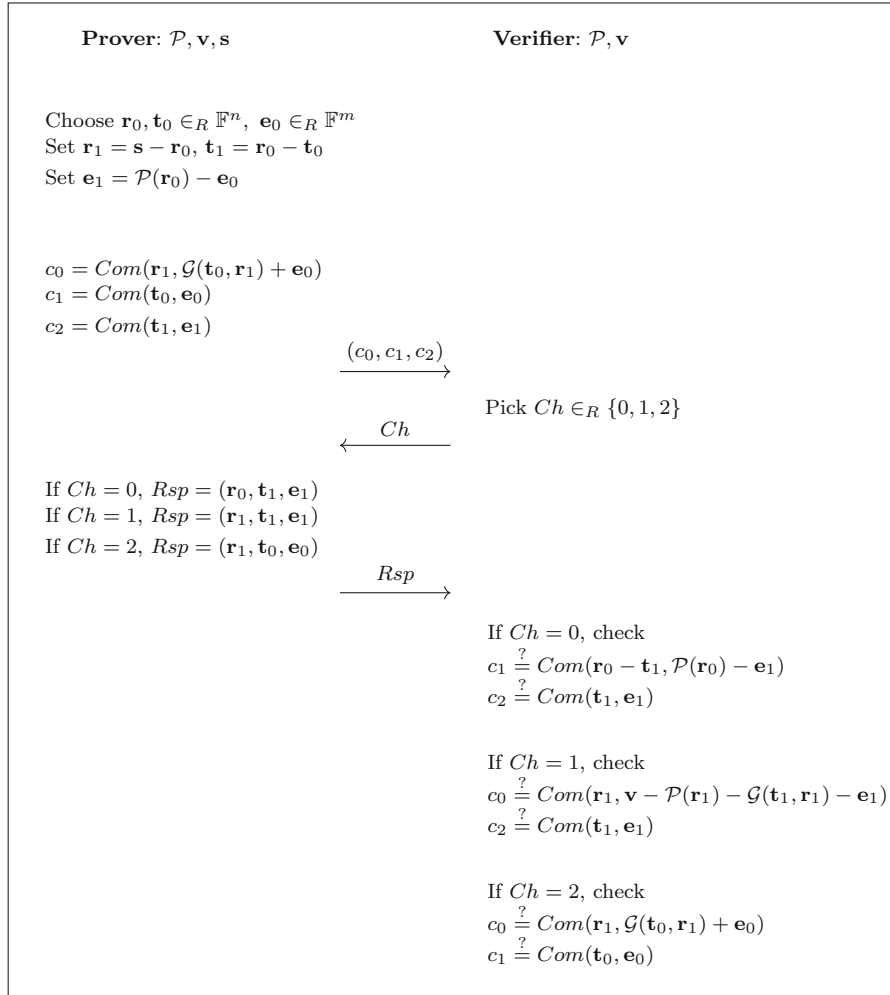


Fig. 6.1 The MQ based identification scheme (3-pass version)

In order to show that the two versions of the MQ based identification scheme really provide zero-knowledge proofs of knowledge, we have to prove the following three theorems, of which the first is easy to show.

Theorem 6.2 *The MQ based identification scheme as shown in Figs. 6.1 and 6.2 is correct, i.e. an honest user can respond to all challenges correctly and therefore will pass the identification protocol with probability 1.*

Theorem 6.3 *The MQ based identification scheme as shown in Figs. 6.1 and 6.2 is statistically zero-knowledge, if the underlying commitment scheme Com is statistically hiding.*

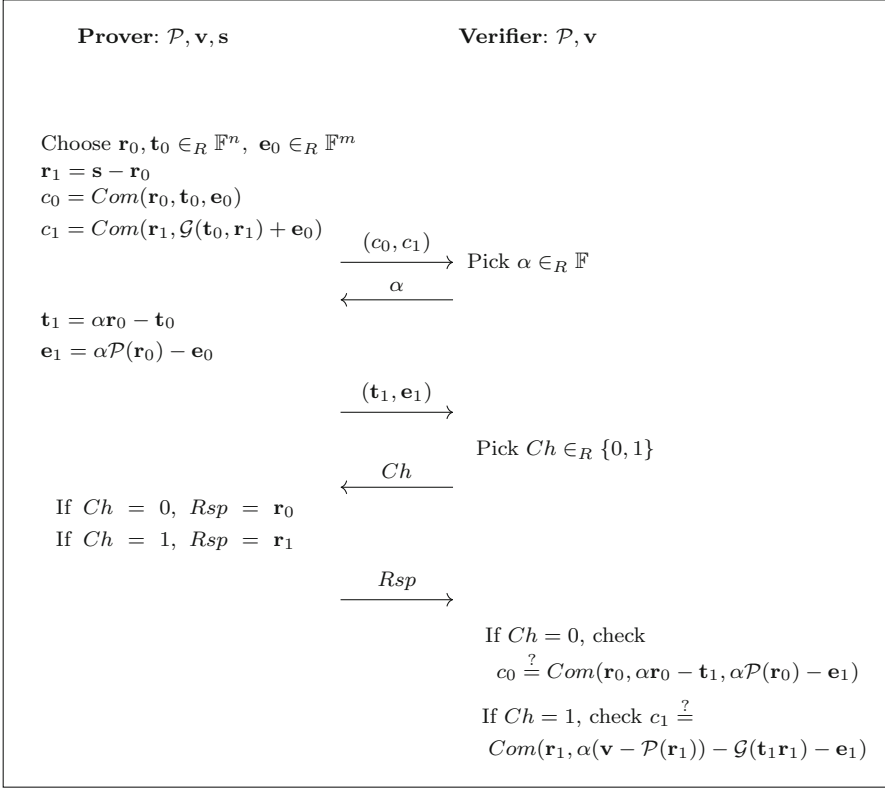


Fig. 6.2 The MQ based identification scheme (5-pass version)

Proof (sketch) We restrict to the 3-pass version (see Fig. 6.1). Let \mathcal{S} be a simulator which only gets the public key $(\mathcal{P}, \mathbf{v})$ interacting with a cheating verifier \mathcal{CV} . We have to show that \mathcal{S} can impersonate an honest prover with probability $2/3$. To do this, \mathcal{S} chooses randomly a value $Ch^* \in \{0, 1, 2\}$ and vectors $\mathbf{s}', \mathbf{r}'_0, \mathbf{t}'_0 \in \mathbb{F}^n, \mathbf{e}'_0 \in \mathbb{F}^m$. Here, Ch^* is a prediction which challenge the cheating verifier \mathcal{CV} will **not** choose. Then, \mathcal{S} computes $\mathbf{r}'_1 = \mathbf{s}' - \mathbf{r}'_0$ and $\mathbf{t}'_1 = \mathbf{r}'_0 - \mathbf{t}'_0$. Furthermore, he computes

- If $Ch^* = 0$: $\mathbf{e}'_1 = \mathbf{v} - \mathcal{P}(\mathbf{s}') + \mathcal{P}(\mathbf{r}'_0) - \mathbf{e}'_0$ and $c'_0 = \text{Com}(\mathbf{r}'_1, \mathcal{G}(\mathbf{t}'_0, \mathbf{r}'_1) + \mathbf{e}'_0)$
- If $Ch^* = 1$: $\mathbf{e}'_1 = \mathcal{P}(\mathbf{r}'_0) - \mathbf{e}'_0$ and $c'_0 = \text{Com}(\mathbf{r}'_1, \mathcal{G}(\mathbf{t}'_0, \mathbf{r}'_1) + \mathbf{e}'_0)$
- If $Ch^* = 2$: $\mathbf{e}'_1 = \mathcal{P}(\mathbf{r}'_0) - \mathbf{e}'_0$ and $c'_0 = \text{Com}(\mathbf{r}'_1, \mathbf{v} - \mathcal{P}(\mathbf{r}'_1) - \mathcal{G}(\mathbf{t}'_1, \mathbf{r}'_1) + \mathbf{e}'_1)$.

Then, \mathcal{S} computes $c'_1 = \text{Com}(\mathbf{t}'_0, \mathbf{e}'_0)$ and $c'_2 = \text{Com}(\mathbf{t}'_1, \mathbf{e}'_1)$ and sends the commitments (c'_0, c'_1, c'_2) to \mathcal{CV} .

Due to the statistical hiding property of Com , the challenge chosen by \mathcal{CV} will be different from Ch^* with probability $2/3$. In this case, the responses Rsp_i ($i \neq Ch^*$)

will be accepted. Therefore, the simulator \mathcal{S} can produce a valid transcript of one round of the identification scheme with probability $\frac{2}{3}$. \square

Theorem 6.4 *If the commitment scheme Com is computationally binding, the 3- and 5-pass versions of the MQ based identification scheme are arguments of knowledge for a solution \mathbf{s} of $\mathcal{P}(\mathbf{x}) = \mathbf{v}$ with knowledge errors of $\frac{2}{3}$ and $\frac{1}{2} + \frac{1}{2q}$ respectively.*

Proof (Sketch) Again we restrict to the 3-pass version of the identification scheme. We have to show that a user, who is able to respond to all three challenges correctly, can either break the binding property of the commitment scheme Com or extract a solution of the given instance of the MQ Problem. So, let $Rsp_0 = (\tilde{\mathbf{r}}_0^{(0)}, \tilde{\mathbf{t}}_1^{(0)}, \tilde{\mathbf{e}}_1^{(0)})$, $Rsp_1 = (\tilde{\mathbf{r}}_1^{(1)}, \tilde{\mathbf{t}}_1^{(1)}, \tilde{\mathbf{e}}_1^{(1)})$ and $Rsp_2 = (\tilde{\mathbf{r}}_1^{(2)}, \tilde{\mathbf{t}}_0^{(2)}, \tilde{\mathbf{e}}_0^{(2)})$ be valid responses to the challenges 0, 1 and 2 respectively (for fixed commitments c_0, c_1 and c_2). Since all three responses are found to be correct, we have

$$\begin{aligned} c_0 &= Com(\tilde{\mathbf{r}}_1^{(1)}, \mathbf{v} - \mathcal{P}(\tilde{\mathbf{r}}_1^{(1)}) - \mathcal{G}(\tilde{\mathbf{t}}_1^{(1)}, \tilde{\mathbf{r}}_1^{(1)}) - \tilde{\mathbf{e}}_1^{(1)}) \\ &= Com(\tilde{\mathbf{r}}_1^{(2)}, \mathcal{G}(\tilde{\mathbf{t}}_0^{(2)}, \tilde{\mathbf{r}}_1^{(2)}) + \tilde{\mathbf{e}}_0^{(2)}), \end{aligned} \quad (6.4)$$

and

$$\begin{aligned} c_1 &= Com(\tilde{\mathbf{r}}_0^{(0)} - \tilde{\mathbf{t}}_1^{(0)}, \mathcal{P}(\tilde{\mathbf{r}}_0^{(0)}) - \tilde{\mathbf{e}}_1^{(0)}) \\ &= Com(\tilde{\mathbf{t}}_0^{(2)}, \tilde{\mathbf{e}}_0^{(2)}), \end{aligned} \quad (6.5)$$

and

$$\begin{aligned} c_2 &= Com(\tilde{\mathbf{t}}_1^{(0)}, \tilde{\mathbf{e}}_1^{(0)}) \\ &= Com(\tilde{\mathbf{t}}_1^{(1)}, \tilde{\mathbf{e}}_1^{(1)}). \end{aligned} \quad (6.6)$$

If, in any of the Eqs. (6.4)–(6.6), the input values of Com in the two rows are distinct, the bounding property of the commitment scheme is broken. Otherwise we find from (6.4) $\tilde{\mathbf{r}}_1^{(1)} = \tilde{\mathbf{r}}_1^{(2)}$ and therefore

$$\mathbf{v} = \mathcal{P}(\tilde{\mathbf{r}}_1^{(2)}) + \mathcal{G}(\tilde{\mathbf{t}}_1^{(1)} + \tilde{\mathbf{t}}_0^{(2)}, \tilde{\mathbf{r}}_1^{(2)}) + \tilde{\mathbf{e}}_0^{(2)} + \tilde{\mathbf{e}}_1^{(1)}.$$

Furthermore we find

$$\tilde{\mathbf{t}}_1^{(1)} + \tilde{\mathbf{t}}_0^{(2)} \stackrel{(6.6)}{=} \tilde{\mathbf{t}}_1^{(0)} + \tilde{\mathbf{t}}_0^{(2)} \stackrel{(6.5)}{=} \tilde{\mathbf{r}}_0^{(0)}$$

and

$$\tilde{\mathbf{e}}_0^{(2)} + \tilde{\mathbf{e}}_1^{(1)} \stackrel{(6.6)}{=} \tilde{\mathbf{e}}_0^{(2)} + \tilde{\mathbf{e}}_1^{(0)} \stackrel{(6.5)}{=} \mathcal{P}(\tilde{\mathbf{r}}_0^{(0)}).$$

Therefore we have

$$\mathbf{v} = \mathcal{P}(\tilde{\mathbf{r}}_1^{(2)}) + \mathcal{G}(\tilde{\mathbf{r}}_0^{(0)}, \tilde{\mathbf{r}}_1^{(2)}) + \mathcal{P}(\tilde{\mathbf{r}}_0^{(0)}) \stackrel{(6.1)}{=} \mathcal{P}(\tilde{\mathbf{r}}_0^{(0)} + \tilde{\mathbf{r}}_1^{(2)}).$$

We have therefore found a solution of the equation $\mathcal{P}(\mathbf{x}) = \mathbf{v}$. \square

6.1.1 Reducing the Communication Cost

For the 3-pass identification scheme, we can reduce the overall communication cost by using the following technique. Instead of sending the three commitments c_0, c_1, c_2 to the verifier, the prover sends the hash value $com = \mathcal{H}(c_0, c_1, c_2)$ for a collision resistant hash function \mathcal{H} . To enable verification, he has to add the commitment c_{Ch} to the response. The verifier computes the commitments c_j ($j \neq Ch$) and finally checks if $com = \mathcal{H}(c_0, c_1, c_2)$ holds. By doing so, the communication cost of the 3-pass identification scheme is reduced by the length of one hash value per round.

For the 5-pass identification scheme, this technique does not lead to a reduction of communication directly. However, when using the 5-pass identification scheme as basis of a Fiat-Shamir signature, we can use a similar technique to reduce the signature size significantly (see Sect. 6.3).

6.1.2 Security

An attacker against the identification scheme can have one or both of the goals

1. Impersonate himself as a different user
2. Break the underlying hardness assumption (in this case find a solution of $\mathcal{P}(\mathbf{x}) = \mathbf{v}$).

Note that, due to the correctness of the scheme, a successful attack against (2) implies an attack against (1).

Usually it is demanded that the hardness of the underlying problem is at least 2^{128} , which requires the system \mathcal{P} to consist of at least 148 equations in the same number of variables over $\text{GF}(2)$ (or 32 equations over $\text{GF}(256)$). On the other hand, the security of the scheme against impersonation attacks might be significantly smaller.

In order to reduce the impersonation probability of the 3-pass scheme to 2^{-k} , we need

$$\ell \geq \frac{k}{\log_2 3/2}$$

Table 6.1 Minimal number of rounds needed in the MQ based identification scheme

Impersonation probability	Number of rounds	
	3-pass identification	5-pass identification ($q = 256$)
2^{-32}	55	33
2^{-64}	110	65
2^{-80}	137	81
2^{-128}	219	129

rounds. For the 5-pass version, the minimal number of rounds is given by

$$\ell \geq \frac{k}{\log_2(1/2 + 1/(2q))}.$$

Table 6.1 shows, for different values of k , the minimal number of rounds needed to reduce the impersonation probability below 2^{-k} .

6.1.3 Key Sizes and Efficiency

Since the public system \mathcal{P} of the MQ based identification scheme is a completely random system, it can be stored as a small seed of e.g. 256 bits. Together with the vector \mathbf{v} , we therefore obtain a public key size of

$$\text{size}_{\text{pk MQident}} = m \lceil \log_2 q \rceil + 256 \text{ bits.}$$

For a security level of 128 bit, the public key size of the MQ based identification scheme is therefore 404 bits (3-pass scheme, $q = 2$) or 512 bits (5-pass scheme, $q = 256$).

The private key consists of the vector \mathbf{s} , which can be stored using

$$\text{size}_{\text{sk MQident}} = n \lceil \log_2 q \rceil \text{ bits.}$$

For the 3-pass scheme ($q = 2$), this results in a private key size of 148 bits, for the 5-pass scheme ($q = 256$), we get a private key size of 256 bits.

For the 3-pass version, the communication cost between prover and verifier is

$$3|h| + 2 + (2n + m) \lceil \log_2 q \rceil \text{ bits.}$$

per round, where $|h|$ is the output length of the commitment scheme Com . Using the above technique to reduce the communication cost, we can reduce this to

$$2|h| + 2 + (2n + m) \lceil \log_2 q \rceil \text{ bits.}$$

Table 6.2 Key sizes (in bit) and communication cost of the MQ based identification scheme

Impersonation probability	Communication cost (kB)	
	3-pass scheme ($q = 2$)	5-pass scheme ($q = 256$)
2^{-32}	8.16 (6.43)	5.19
2^{-64}	16.3 (12.9)	10.2
2^{-80}	20.3 (16.0)	12.7
2^{-128}	32.5 (25.6)	20.3

The hardness of the underlying MQ instance is 128 bit. For the 3-pass scheme, the number in brackets shows the overall communication cost using the mentioned idea to reduce it

For the 5-pass version, the communication cost per round is

$$2|h| + 1 + (2n + m + 1)\lceil \log_2 q \rceil \text{ bits.}$$

Table 6.2 shows the overall communication cost of the 3- and 5-pass MQ based identification scheme for different impersonation probabilities (128 bit security of the underlying MQ Problem, $|h| = 256$). In the fourth column, the number in brackets shows the overall communication cost using the above technique to reduce it.

6.1.4 Toy Example

We choose $m = n = 4$ and take $\mathbb{F} = GF(4)$ as the underlying finite field, whose addition and multiplication tables are given in Fig. 3.3. We choose randomly a multivariate quadratic system $\mathcal{P} = (p^{(1)}, \dots, p^{(4)}) : \mathbb{F}^4 \rightarrow \mathbb{F}^4$ of the form

$$\begin{aligned}
p^{(1)}(x_1, \dots, x_4) &= \alpha^2 x_1 x_3 + x_1 x_4 + x_2^2 + \alpha^2 x_2 x_3 + x_2 x_4 + \alpha^2 x_2 \\
&\quad + x_3^2 + \alpha^2 x_3 + \alpha x_4 + 1, \\
p^{(2)}(x_1, \dots, x_4) &= \alpha x_1^2 + \alpha^2 x_1 x_2 + x_1 x_4 + \alpha x_2^2 + \alpha x_2 x_3 + \alpha x_2 x_4 \\
&\quad + \alpha^2 x_2 + \alpha^2 x_3^2 + \alpha^2 x_3 x_4 + \alpha x_4 + \alpha^2, \\
p^{(3)}(x_1, \dots, x_4) &= \alpha^2 x_1^2 + \alpha^2 x_1 x_3 + x_1 + x_2 x_3 + \alpha^2 x_2 x_4 + x_2 + \alpha x_3^2 \\
&\quad + x_3 x_4 + \alpha x_3 + \alpha x_4 + \alpha^2, \\
p^{(4)}(x_1, \dots, x_4) &= x_1 x_2 + x_2^2 + \alpha x_2 x_3 + \alpha^2 x_2 x_4 + \alpha^2 x_2 + \alpha^2 x_3^2 \\
&\quad + \alpha^2 x_3 x_4 + \alpha^2 x_4^2 + \alpha.
\end{aligned}$$

The prover randomly chooses a vector $\mathbf{s} \in \mathbb{F}^4$ as his private key, for example

$$\mathbf{s} = (\alpha, \alpha, 1, 1),$$

and computes $\mathbf{v} = \mathcal{F}(\mathbf{s})$ as his public key. He obtains

$$\mathbf{v} = (\alpha^2, 0, 0, \alpha).$$

In the following we perform two rounds of the 3-pass identification scheme.

1. Round

We choose randomly 3 vectors $\mathbf{r}_0, \mathbf{t}_0$ and $\mathbf{e}_0 \in \mathbb{F}^4$, e.g.

$$\mathbf{r}_0 = (0, \alpha, \alpha, 1),$$

$$\mathbf{t}_0 = (\alpha, 0, 1, \alpha^2),$$

$$\mathbf{e}_0 = (\alpha, 0, \alpha, 1)$$

and compute

$$\mathbf{r}_1 = \mathbf{s} - \mathbf{r}_0 = (\alpha, 0, \alpha^2, 0),$$

$$\mathbf{t}_1 = \mathbf{r}_0 - \mathbf{t}_0 = (\alpha, \alpha, \alpha^2, \alpha),$$

$$\mathbf{e}_1 = \mathcal{P}(\mathbf{r}_0) - \mathbf{e}_0 = (1, 0, \alpha^2, 1).$$

Finally, we compute the three commitments

$$c_0 = \text{Com}(\mathbf{r}_1, \mathcal{G}(\mathbf{t}_0, \mathbf{r}_1) + \mathbf{e}_0) = \text{Com}(\alpha, 0, \alpha^2, 0, 1, 0, \alpha, 0)$$

$$c_1 = \text{Com}(\mathbf{t}_0, \mathbf{e}_0) = \text{Com}(\alpha, 0, 1, \alpha^2, \alpha, 0, \alpha, 1)$$

$$c_2 = \text{Com}(\mathbf{t}_1, \mathbf{e}_1) = \text{Com}(\alpha, \alpha, \alpha^2, \alpha, 1, 0, \alpha^2, 1)$$

and send them to the verifier. Let us assume that we get $Ch_1 = 0$ as the challenge for round 1. Therefore, we have to compute the response as

$$Rsp_1 = (\mathbf{r}_0, \mathbf{t}_1, \mathbf{e}_1) = (0, \alpha, \alpha, 1, \alpha, \alpha, \alpha^2, \alpha, 1, 0, \alpha^2, 1).$$

2. Round

We choose randomly 3 vectors r_0, t_0 and $e_0 \in \mathbb{F}^4$, e.g.

$$\mathbf{r}_0 = (\alpha, 0, \alpha, \alpha),$$

$$\mathbf{t}_0 = (\alpha, 1, 1, 1),$$

$$\mathbf{e}_0 = (0, 1, \alpha^2, \alpha^2)$$

and compute

$$\begin{aligned}
\mathbf{r}_1 &= \mathbf{s} - \mathbf{r}_0 = (0, \alpha, \alpha^2, \alpha^2), \\
\mathbf{t}_1 &= \mathbf{r}_0 - \mathbf{t}_0 = (0, 1, \alpha^2, \alpha^2), \\
\mathbf{e}_1 &= \mathcal{P}(\mathbf{r}_0) - \mathbf{e}_0 = (1, \alpha^2, 0, \alpha^2).
\end{aligned}$$

Finally, we compute the three commitments

$$\begin{aligned}
c_0 &= \text{Com}(\mathbf{r}_1, \mathcal{G}(\mathbf{t}_0, \mathbf{r}_1) + \mathbf{e}_0) = \text{Com}(0, \alpha, \alpha^2, \alpha^2, 0, \alpha, \alpha, 1) \\
c_1 &= \text{Com}(\mathbf{t}_0, \mathbf{e}_0) = \text{Com}(\alpha, 1, 1, 1, 0, 1, \alpha^2, \alpha^2) \\
c_2 &= \text{Com}(\mathbf{t}_1, \mathbf{e}_1) = \text{Com}(0, 1, \alpha^2, \alpha^2, 1, \alpha^2, 0, \alpha^2)
\end{aligned}$$

and send them to the verifier. Let us assume that we get $Ch_2 = 2$ as the challenge for round 2. Therefore, we have to compute the response as

$$Rsp_2 = (\mathbf{r}_1, \mathbf{t}_0, \mathbf{e}_0) = (0, \alpha, \alpha^2, \alpha^2, \alpha, 1, 1, 1, 0, 1, \alpha^2, \alpha^2).$$

6.2 The Fiat-Shamir Transformation

In [2], Fiat and Shamir proposed a general construction to convert an identification scheme into a digital signature scheme. The basic idea is to produce a transcript of the interactive identification protocol over r rounds. The challenges Ch_1, \dots, Ch_r are hereby derived from the hash value of the message to be signed. The resulting signature scheme can be described as follows:

Public Key The public key of the underlying identification scheme.

Private Key The private key of the underlying identification scheme.

Signature Generation To generate a signature for a message $d \in \{0, 1\}^*$, the signer uses a hash function $\mathcal{H} : \{0, 1\}^* \rightarrow \{0, 1\}^n$ to compute a hash value $\mathbf{h} = \mathcal{H}(d)$ of the message and two derivation functions \mathcal{D}_{Ch} and \mathcal{D}_{Com} to derive from \mathbf{h} the challenges Ch_1, \dots, Ch_r and the commitments (Com_1, \dots, Com_r) of the identification scheme. He computes, for $i = 1, \dots, r$, the responses Rsp_i of the identification scheme corresponding to Com_i and Ch_i . The signature σ of the message d is given by

$$\sigma = (Com_1, \dots, Com_r, Rsp_1, \dots, Rsp_r).$$

Since the functions \mathcal{H} and \mathcal{D}_{Ch} are publicly known, the challenges Ch_1, \dots, Ch_r do not have to be part of the signature.

Signature Verification To check if σ is indeed a valid signature for the document d , the verifier uses the functions \mathcal{H} and \mathcal{D}_{Ch} to compute the challenges Ch_1, \dots, Ch_r . Then he parses σ into $Com_1, \dots, Com_r, Rsp_1, \dots, Rsp_r$ and checks, for $i = 1, \dots, r$, if Rsp_i is the correct response according to Com_i and Ch_i . The signature σ is accepted if and only if all these tests are fulfilled. If one of the test fails, the signature is rejected.

6.2.1 Security Analysis

The security of the Fiat-Shamir signature scheme was proven by Stern et al. in [3]. The proof is given in the so called random oracle model.

Definition 6.5 A **random oracle** is an oracle which, on every unique query, outputs a response chosen uniformly at random from its output domain. If a query is repeated, it gives the same response every time the query is submitted.

In the random oracle model, one assumes that every hash function behaves like a random oracle.

The security proof for the Fiat-Shamir transform is based on

Theorem 6.6 (Forking Lemma) *Let \mathcal{A} be a PPT Turing Machine receiving only public data as input. If \mathcal{A} can find a valid signature $(m, \sigma_1, h_1, \sigma_2)$, then a replay of the same machine, with the same random tape but a different random oracle, can find another valid signature $(m, \sigma_1, h', \sigma'_2)$ with $h \neq h'$.*

Proof see [3], Lemma 2. □

Using the Forking Lemma it is now possible to prove

Theorem 6.7 *Forging a Fiat-Shamir signature implies breaking the security assumption of the underlying identification scheme.*

Proof We proof the theorem for the case of the MQ based identification scheme of Sect. 6.1 being the scheme underlying the Fiat-Shamir construction.

Let \mathcal{A} be an algorithm forging a signature for a message d and commitments Com_1, \dots, Com_r . Due to the forking lemma, we find that, by applying the algorithm \mathcal{A} several times (with different random oracles), we can get valid signatures $\sigma^{(1)}, \dots, \sigma^{(k)}$ for (d, Com_1, \dots, Com_r) of the form

$$\sigma^{(i)} = (Com_1, \dots, Com_r, Ch_1^{(i)}, \dots, Ch_r^{(i)}, Rsp_1^{(i)}, \dots, Rsp_r^{(i)})$$

where $(Ch_1^{(i)}, \dots, Ch_r^{(i)}) \neq (Ch_1^{(j)}, \dots, Ch_r^{(j)})$ for $i \neq j$.

In particular, we can find three signatures $\tilde{\sigma}_0, \tilde{\sigma}_1$ and $\tilde{\sigma}_2$ such that $Ch_l^{(0)} = 0$, $Ch_l^{(1)} = 1$ and $Ch_l^{(2)} = 2$ for some $l \in \{1, \dots, r\}$. Since $\tilde{\sigma}_1, \tilde{\sigma}_2$ and $\tilde{\sigma}_3$ are valid Fiat-Shamir signatures, we know that $Rsp_l^{(i)}$ is a valid response to

$Com_i, Ch_i^{(i)}$ ($i = 0, 1, 2$). As shown by Theorem 6.4, we can extract from this a solution of the MQ instance $\mathcal{P}(\mathbf{x}) = \mathbf{v}$. \square

Remark 6.8 The security proof of the Fiat-Shamir construction as given above is only true in the classical random oracle model (i.e. the attacker has only classical access to the random oracle). However, under some additional assumptions, the Fiat-Shamir construction still leads to a provably secure signature scheme, even when quantum attacks are considered [5].

Remark 6.9 The number r of rounds of the identification scheme underlying a Fiat-Shamir signature must be chosen in a way that the security requirements of the signature scheme are fulfilled. In particular, we do not longer distinguish between the security level against impersonation attacks and that of attacks against the underlying problem, but require that both meet a high security level of at least 128 bit. In the case of signatures based on the MQ based identification scheme, this means that we need at least 219 rounds (3-pass version) or 129 rounds (5-pass version; $q = 256$).

6.3 The MQDSS Signature Scheme

The MQDSS signature scheme was proposed by Chen et al. in [1] and later submitted to the NIST competition for post-quantum public key cryptosystems. The scheme uses the Fiat-Shamir construction to transform the 5-pass version of the MQ based identification scheme (see Sect. 6.1) into a signature scheme. Therefore, the security of the scheme is solely based on the MQ Problem, which makes the scheme to be one of the very few provable secure multivariate public key schemes.

The scheme uses the following parameters:

- a positive integer k —the security parameter
- a positive integer $n = n(k)$ —the number of variables and equations in the multivariate quadratic system
- a positive integer $q = q(k)$ (a prime or prime power)—the size of the underlying field
- a positive integer $r = r(k)$ —the number of rounds

Key Generation In order to generate a key pair of MQDSS, the signer proceeds as follows:

- He chooses randomly a bitstring sk of length k
- using a PRNG, he derives from sk the three seeds $S_{\mathcal{P}}$, $S_{\mathbf{s}}$ and S_{rte} :
 - $S_{\mathcal{P}}$ is used to generate the system parameter \mathcal{P} (a multivariate quadratic system of n equations in n variables)
 - $S_{\mathbf{s}}$ is used to generate the input $\mathbf{s} \in \mathbb{F}^n$ of the multivariate system

- S_{rte} is used in the signature generation process to derive all the values $\mathbf{r}_0^{(i)}, \mathbf{t}_0^{(i)}$ and $\mathbf{e}_0^{(i)}$ ($i = 1, \dots, r$). (This step is not performed during key generation.)
- Finally, he computes $\mathbf{v} = \mathcal{P}(\mathbf{s})$.

The *private key* of the scheme is the seed sk , the *public key* consists of the seed $S_{\mathcal{P}}$ and the vector $\mathbf{v} \in \mathbb{F}^n$.

Signature Generation In order to generate a signature for a document $d \in \{0, 1\}^*$, the signer generates a valid transcript of r rounds of the 5-pass MQ based identification scheme of Sect. 6.1. In particular, he performs the following steps

1. Derive the seeds $S_{\mathcal{P}}, S_{\mathcal{S}}$ and S_{rte} from sk .
2. Derive the multivariate quadratic system \mathcal{P} from $S_{\mathcal{P}}$.
3. Derive the vector \mathbf{s} from $S_{\mathcal{S}}$ and compute $\mathbf{v} = \mathcal{P}(\mathbf{s})$. Set $pk = (S_{\mathcal{P}}, \mathbf{v})$.
4. Use a hash function \mathcal{H} to compute a message dependent random value R by $R = \mathcal{H}(sk, d)$ and a randomized message digest D by $D = \mathcal{H}(pk, R, d)$. R must be included in the signature to enable verification.
5. Derive from S_{rte} and D the values $\mathbf{r}_0^{(1)}, \dots, \mathbf{r}_0^{(r)}, \mathbf{t}_0^{(1)}, \dots, \mathbf{t}_0^{(r)}, \mathbf{e}_0^{(1)}, \dots, \mathbf{e}_0^{(r)}$.
6. For $i = 1, \dots, r$ compute

$$\begin{aligned}\mathbf{r}_1^{(i)} &= \mathbf{s} - \mathbf{r}_0^{(i)} \\ c_0^{(i)} &= \text{Com}(\mathbf{r}_0^{(i)}, \mathbf{t}_0^{(i)}, \mathbf{e}_0^{(i)}) \\ c_1^{(i)} &= \text{Com}(\mathbf{r}_1^{(i)}, \mathcal{G}(\mathbf{t}_0^{(i)}, \mathbf{r}_1^{(i)}) + \mathbf{e}_0^{(i)}) \\ \text{com}^{(i)} &= (c_0^{(i)}, c_1^{(i)})\end{aligned}$$

7. Set $\sigma_0 = \mathcal{H}(\text{com}^{(1)}, \dots, \text{com}^{(r)})$
8. Derive from D and σ_0 the first challenge ch_1 and parse it into $\alpha^{(1)}, \dots, \alpha^{(r)} \in \mathbb{F}$.
9. For $i = 1, \dots, r$ compute

$$\begin{aligned}\mathbf{t}_1^{(i)} &= \alpha^{(i)} \mathbf{r}_0^{(i)} - \mathbf{t}_0^{(i)} \\ \mathbf{e}_1^{(i)} &= \alpha^{(i)} \mathcal{F}(\mathbf{r}_0^{(i)}) - \mathbf{e}_0^{(i)}\end{aligned}$$

and set $Rsp_1^{(i)} = (\mathbf{t}_1^{(i)}, \mathbf{e}_1^{(i)})$.

10. Set $\sigma_1 = (Rsp_1^{(1)}, \dots, Rsp_1^{(r)})$ and derive from D, σ_0, ch_1 and σ_1 the challenges $Ch^{(1)}, \dots, Ch^{(r)} \in \{0, 1\}$.
11. For $i = 1, \dots, r$ compute the response $Rsp_2^{(i)}$
 - If $Ch^{(i)} = 0$, $Rsp^{(i)} = \mathbf{r}_0^{(i)}$.
 - If $Ch^{(i)} = 1$, $Rsp^{(i)} = \mathbf{r}_1^{(i)}$.

12. Set $\sigma_2 = (Rsp_2^{(1)}, \dots, Rsp_2^{(r)}, c_{1-Ch(1)}^{(1)}, \dots, c_{1-Ch(r)}^{(r)})$.

The signature for the message d is given as

$$\sigma = (R, \sigma_0, \sigma_1, \sigma_2).$$

The length of the signature σ is $|\sigma| = (2 + r)k + 3rn \lceil \log_2 q \rceil$ bits.

Remark 6.10 For a standard Fiat-Shamir signature, we would have to include all the commitments $c_0^{(1)}, c_1^{(1)}, \dots, c_0^{(r)}, c_1^{(r)}$ into the signature. However, as shown in step 7 of the signature generation process, it is enough to include the hash value $com = \mathcal{H}(c_0^{(1)}, c_1^{(1)}, \dots, c_0^{(r)}, c_1^{(r)})$, when we include the commitments $c_{1-Ch(1)}^{(1)}, \dots, c_{1-Ch(r)}^{(r)}$ (see step 12). During verification, the verifier computes the commitments $c_{Ch(1)}^{(1)}, \dots, c_{Ch(r)}^{(r)}$ and checks, if $com = \mathcal{H}(c_0^{(1)}, c_1^{(1)}, \dots, c_0^{(r)}, c_1^{(r)})$ holds. If this is the case, the signature is accepted, otherwise it is rejected.

By this modification, we can reduce the signature size by $(r - 1)$ hash lengths.

Signature Verification In order to check, if σ is a valid MQDSS signature for the message d , the verifier parses σ into R, σ_0, σ_1 and σ_2 . He derives from R, σ_0 and σ_1 the challenges $\alpha^{(1)}, \dots, \alpha^{(r)}, Ch^{(1)}, \dots, Ch^{(r)}$, computes the commitments $c_{Ch(1)}^{(1)}, \dots, c_{Ch(r)}^{(r)}$ and finally checks, if

$$\sigma_0 = \mathcal{H}(c_0^{(1)}, c_1^{(1)}, \dots, c_0^{(r)}, c_1^{(r)})$$

holds. If this is the case, he accepts the signature σ , otherwise he rejects it.

In detail, the verifier performs the following steps.

1. Derive from \mathcal{SP} the multivariate quadratic system \mathcal{P} .
2. Parse σ into R, σ_0, σ_1 and σ_2 and derive from R the randomized message digest $D = \mathcal{H}(pk, R, d)$.
3. Derive from D and σ_0 the first challenge ch_1 and parse it into $\alpha^{(1)}, \dots, \alpha^{(r)} \in \mathbb{F}$.
4. Derive from D, σ_0, ch_1 and σ_1 the challenges $Ch^{(1)}, \dots, Ch^{(r)} \in \{0, 1\}$.
5. Parse σ_1 into $Rsp_1^{(1)}, \dots, Rsp_1^{(r)}$ and σ_2 into $Rsp_2^{(1)}, \dots, Rsp_2^{(r)}, c_{1-Ch(1)}^{(1)}, \dots, c_{1-Ch(r)}^{(r)}$.
6. Recover the missing commitments $c_{Ch(1)}^{(1)}, \dots, c_{Ch(r)}^{(r)}$ by performing for $i = 1, \dots, r$ the following steps
 - (a) Parse $Rsp_1^{(i)}$ into $\mathbf{t}_1^{(i)}, \mathbf{e}_1^{(i)}$.
 - (b) If $Ch^{(i)} = 0$, compute $c_0^{(i)}$ as

$$c_0^{(i)} = Com(\mathbf{r}_0^{(i)}, \alpha^{(i)} \mathbf{r}_0^{(i)} - \mathbf{t}_1^{(i)}, \alpha^{(i)} \mathcal{P}(\mathbf{r}_0^{(i)}) - \mathbf{e}_1^{(i)}),$$

otherwise compute $c_1^{(i)}$ as

$$c_1 = \text{Com}(\mathbf{r}_1^{(i)}, \alpha^{(i)}(\mathbf{v} - \mathcal{P}\mathbf{r}_1^{(i)}) - \mathcal{G}(\mathbf{t}_1^{(i)}, \mathbf{r}_1^{(i)}) - e_1^{(i)}).$$

(c) Set $\text{com}^{(i)} = (c_0^{(i)}, c_1^{(i)})$.

7. Compute $\sigma'_0 = \mathcal{H}(\text{com}^{(1)}, \dots, \text{com}^{(r)})$.

8. If $\sigma'_0 = \sigma_0$ holds, accept the signature, otherwise reject it.

6.3.1 Security

Due to its construction using the Fiat-Shamir transformation, the security of the MQDSS Signature scheme is directly based on the security of the underlying MQ based identification scheme and therefore on the hardness of inverting the system \mathcal{P} . Since \mathcal{P} is a completely random system,³ the security of the scheme is based solely on the hardness of the MQ Problem, which makes the scheme one of the very few provable secure multivariate public key schemes.

6.3.2 Key Sizes and Efficiency

The public key size of the MQDSS signature scheme is given as

$$\text{size}_{\text{pk MQDSS}} = k + \lceil \log_2 q \rceil n \text{ bits},$$

the size of the private key is

$$\text{size}_{\text{sk MQDSS}} = k \text{ bits}.$$

In [1], the authors used the field $\text{GF}(31)$ as the underlying field for MQDSS. Furthermore, they decided to use a determined system \mathcal{P} as the public system parameter.

Since \mathcal{P} is a completely random system, we only have to consider the direct attack. To defend a determined multivariate quadratic system over $\text{GF}(31)$ against this attack, we need, in order to achieve a security level of ℓ bits, about $\ell/3$ equations and variables.

Regarding the signature size, the most important factor is the number of rounds of the identification scheme we need to achieve the given level of security. For the 5-pass identification scheme and $\text{GF}(31)$ as the underlying field, this number is given as

³Here, we assume that the PRNG used to generate the system \mathcal{P} from the seed sk works fine.

Table 6.3 Parameters and key sizes of MQDSS

Security category	Parameters (k, q, n, r)	Public key size (bytes)	Secret key size (bytes)	Signature size (bytes)
I	(256, 31, 48, 269)	62	32	32,882
II	(384, 31, 64, 403)	88	48	67,800

$$r = \left\lceil \frac{\ell}{\log_2(1/2 + 1/62)} \right\rceil \approx 0.95\ell.$$

Table 6.3 shows the parameter sets recommended for MQDSS.

As can be seen from the table, the key sizes of MQDSS are much smaller than those of HFEv- or UOV/Rainbow. The reason for this is the choice of \mathcal{P} as a completely random system, which enables to create the system from a small random seed.

On the other hand, the signatures are much larger than those of other multivariate schemes. Also with regard of the efficiency of the signature generation algorithm, the scheme can't compete with Rainbow.

References

1. M. Chen, A. Hülsing, J. Rijneveld, S. Samardjiska, P. Schwabe, From 5-pass *MQ*-based identification to *MQ*-based signatures, in *Advances in Cryptology — ASIACRYPT 2016 Part II*. Lecture Notes in Computer Science, vol. 10032, (Springer, Berlin, 2016), pp. 135–165
2. A. Fiat, A. Shamir, How to prove yourself: Practical solutions to identification and signature problems, in *Advances in Cryptology — CRYPTO 1986*. Lecture Notes in Computer Science, vol. 263 (Springer, Berlin, 1986), pp. 186–194
3. D. Pointcheval, J. Stern, Security proofs for signature schemes, in *Advances in Cryptology — EUROCRYPT '96*. Lecture Notes in Computer Science, vol. 1070 (Springer, Berlin, 1996), pp. 387–398
4. K. Sakumoto, T. Shirai, H. Hiwatari, Public-key identification schemes based on multivariate quadratic polynomials, in *Advances in Cryptology — CRYPTO 2011*. Lecture Notes in Computer Science, vol. 6841 (Springer, Berlin, 2011), pp. 706–723
5. D. Unruh, Post-quantum security of fiat-shamir, in *Advances in Cryptology — ASIACRYPT 2017 - Part I*. Lecture Notes in Computer Science, vol. 10624 (Springer, Berlin, 2017), pp. 65–95

Chapter 7

The SimpleMatrix Encryption Scheme



Abstract In this chapter we study the Simple Matrix encryption scheme, which is one of the very few multivariate encryption schemes from the SingleField family. After introducing the basic Simple Matrix encryption scheme, we consider a variation called rectangular Simple Matrix encryption scheme. This scheme allows a more flexible parameter choice which leads to smaller key sizes and a decreases the probability of decryption failures. We end this chapter by a security analysis of the Simple Matrix scheme.

In the previous chapters, we already discussed a number of multivariate encryption schemes from the BigField family (e.g. MI, PMI+, HFE, ZHFE). In this chapter we now introduce a SingleField encryption scheme, called the SimpleMatrix (or ABC) encryption scheme.

The central map of the SimpleMatrix encryption scheme is a quadratic map from \mathbb{F}^n to \mathbb{F}^{2n} , whose components are defined as the product of linear maps. In particular, we start with three matrices A , B and C containing linear combinations of the plaintext variables and define the central map \mathcal{F} as $\mathcal{F} = (E_1, E_2)$, where E_1 and E_2 are given as the matrix products $E_1 = A \cdot B$ and $E_2 = A \cdot C$ (hence the name of the scheme).

Although the structure of the SimpleMatrix scheme seems to be very simple, it has (for suitable parameter sets) withstood all cryptanalytic attempts. Since both encryption and decryption are very efficient (we only have to solve linear systems), the scheme is considered to be one of the most promising encryption schemes on the basis of multivariate polynomials.

Section 7.1 describes the basic SimpleMatrix encryption scheme as proposed by Tao et al. in [4]. In Sect. 7.2, we then describe an extension of the scheme called rectangular SimpleMatrix [5], which allows a flexible balance between security and efficiency and an even more efficient decryption process. Finally, in Sect. 7.3, we handle the most important attacks against the SimpleMatrix encryption scheme.

7.1 The Basic SimpleMatrix Encryption Scheme

The SimpleMatrix encryption scheme, as proposed by Tao et al. in [4], can be described as follows.

Key Generation Let \mathbb{F} be a finite field with q elements, r be a small integer and

$$A = \begin{pmatrix} x_1 & \dots & x_r \\ \vdots & & \vdots \\ x_{n-r+1} & \dots & x_n \end{pmatrix} \in \mathbb{F}^{r \times r}.$$

We set $n = r^2$ and $m = 2n$. In order to generate a key pair for the SimpleMatrix scheme, Alice chooses two $r \times r$ matrices B and C containing randomly chosen linear combinations of the variables y_1, \dots, y_n . She computes $E_1 = A \cdot B$ and $E_2 = A \cdot C$. The *central map* of the scheme is given as $\mathcal{F} = (E_1, E_2) : \mathbb{F}^n \rightarrow \mathbb{F}^{2n}$. In order to hide the structure of the central map in the public key, \mathcal{F} is composed with two invertible affine transformations $\mathcal{S} : \mathbb{F}^{2n} \rightarrow \mathbb{F}^{2n}$ and $\mathcal{T} : \mathbb{F}^n \rightarrow \mathbb{F}^n$.

Public Key $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T} : \mathbb{F}^n \rightarrow \mathbb{F}^{2n}$

Private Key The private key consists of

- the three matrices A , B and C and
- the two affine maps \mathcal{S} and \mathcal{T} .

Encryption In order to encrypt a plaintext $\mathbf{z} \in \mathbb{F}^n$, Bob computes $\mathbf{w} = \mathcal{P}(\mathbf{z}) \in \mathbb{F}^{2n}$.

Decryption To decrypt the ciphertext $\mathbf{w} \in \mathbb{F}^{2n}$, Alice performs the following three steps

1. Compute $\mathbf{x} = \mathcal{S}^{-1}(\mathbf{w}) \in \mathbb{F}^{2n}$ and set

$$\bar{E}_1 = \begin{pmatrix} x_1 & \dots & x_r \\ \vdots & & \vdots \\ x_{n-r+1} & \dots & x_n \end{pmatrix}, \quad \bar{E}_2 = \begin{pmatrix} x_{n+1} & \dots & x_{n+r} \\ \vdots & & \vdots \\ x_{m-r+1} & \dots & x_m \end{pmatrix}.$$

2. Assume that, for the (so far unknown) plaintext \mathbf{z} , the matrix $\bar{A} = A(\mathcal{T}(\mathbf{z}))$ is invertible and set $W = \bar{A}^{-1}$. From $E_1 = A \cdot B$ and $E_2 = A \cdot C$ we obtain

$$B = W \cdot \bar{E}_1 \text{ and } C = W \cdot \bar{E}_2.$$

These two relations yield a system of m linear equations in the n unknown elements of W and the n variables y_1, \dots, y_n . By solving this system using Gaussian elimination, we get the vector $\mathbf{y} = (y_1, \dots, y_n) \in \mathbb{F}^n$.

3. Compute the plaintext $\mathbf{z} \in \mathbb{F}^n$ by $\mathbf{z} = \mathcal{T}^{-1}(\mathbf{y})$.

Remark 7.1 If the linear system in step 2 of the decryption process is not invertible, decryption is not possible. In this case, we get a decryption failure. This happens with a probability of

$$\text{Prob}_{\text{decryption failure}} = 1 - (1 - 1/q^n)(1 - 1/q^{n-1}) \cdots (1 - 1/q) \approx \frac{1}{q}.$$

In order to decrease the probability of decryption failures, the SimpleMatrix scheme is therefore mainly used over fields of large characteristic such as $\text{GF}(2^{16})$ or $\text{GF}(2^{32})$.

Remark 7.2 It might happen that the linear system in step 2 of the decryption process has multiple solutions. In this case, one can use redundancy in the plaintext to make the solution unique. One possibility for this is to choose the first coordinate of each plaintext to be one (see the Toy Example below).

7.1.1 Key Sizes and Efficiency

The public key size of SimpleMatrix is

$$\text{size}_{\text{pk SimpleMatrix}} = 2n \frac{(n+1)(n+2)}{2}$$

\mathbb{F} -elements. The size of the private key is

$$\text{size}_{\text{sk SimpleMatrix}} = \underbrace{n(n+1)}_S + \underbrace{n(n+1)}_T + \underbrace{2n(n+1)}_F$$

\mathbb{F} -elements. So, in contrast to most other SingleField schemes such as UOV or Rainbow, the private key size of SimpleMatrix is only quadratic in n .

The decryption process of the SimpleMatrix encryption scheme requires only the solution of linear systems. This can be implemented much more efficiently than for example the decryption of HFE and its variants which requires inverting a univariate polynomial of high degree. Moreover, unlike schemes which use internal perturbation, the decryption process of Simple Matrix contains no guessing step. It is therefore much more efficient than the decryption step of the BigField multivariate encryption schemes discussed so far.

7.1.2 Toy Example

In the following we illustrate the workflow of the SimpleMatrix encryption scheme using a toy example. We choose $\mathbb{F} = \text{GF}(4)$ as the underlying field and $r = 2$.

Therefore, the public key of our scheme consists of 8 quadratic equations in 4 variables.

The private key of our scheme consists of the two affine maps $\mathcal{S} : \mathbb{F}^8 \rightarrow \mathbb{F}^8$,

$$\mathcal{S}(x_1, \dots, x_8) = \begin{pmatrix} \alpha^2 & 1 & 0 & 0 & \alpha^2 & \alpha & \alpha & 0 \\ \alpha^2 & 0 & 0 & \alpha^2 & \alpha^2 & 0 & 1 & \alpha \\ 0 & 0 & 1 & 0 & \alpha & \alpha & 0 & 0 \\ 1 & 0 & \alpha & \alpha^2 & 1 & \alpha^2 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & \alpha & 1 & 1 \\ \alpha & \alpha^2 & \alpha^2 & \alpha & \alpha^2 & 0 & \alpha & \alpha^2 \\ \alpha^2 & \alpha & 1 & 1 & 0 & \alpha^2 & \alpha & 0 \\ \alpha^2 & \alpha^2 & 0 & \alpha^2 & \alpha^2 & \alpha^2 & \alpha^2 & \alpha \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{pmatrix} + \begin{pmatrix} \alpha \\ \alpha \\ \alpha^2 \\ \alpha^2 \\ 1 \\ \alpha \\ 0 \\ 1 \end{pmatrix}$$

and $\mathcal{T} : \mathbb{F}^4 \rightarrow \mathbb{F}^4$,

$$\mathcal{T}(x_1, \dots, x_4) = \begin{pmatrix} 0 & 1 & 1 & \alpha^2 \\ \alpha^2 & 1 & 0 & \alpha^2 \\ \alpha^2 & \alpha^2 & \alpha & \alpha^2 \\ 1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_1 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

as well as the two matrices $B, C \in \mathbb{F}[x_1, x_2, x_3, x_4]^{2 \times 2}$

$$B = \begin{pmatrix} \alpha x_4 & x_1 + x_2 + \alpha^2 x_3 \\ \alpha x_1 + \alpha x_3 + \alpha^2 x_4 & \alpha^2 x_1 + \alpha x_3 + x_4 \end{pmatrix} \text{ and}$$

$$C = \begin{pmatrix} \alpha^2 x_2 + x_4 & x_2 + \alpha x_3 + x_4 \\ \alpha^2 x_2 + x_4 & x_1 + \alpha^2 x_2 + \alpha x_3 + \alpha^2 x_4 \end{pmatrix}.$$

In order to compute the public key, we first compute the matrices

$$\begin{aligned} E_1 &= A \cdot B = \begin{pmatrix} x_1 & x_2 \\ x_3 & x_4 \end{pmatrix} \cdot B \\ &= \begin{pmatrix} \alpha x_1 x_2 + \alpha x_1 x_4 + \alpha x_2 x_3 + \alpha^2 x_2 x_4 & x_1^2 + \alpha x_1 x_2 + \alpha^2 x_1 x_3 + \alpha x_2 x_3 + x_2 x_4 \\ \alpha x_1 x_4 + \alpha^2 x_4^2 & x_1 x_3 + \alpha^2 x_1 x_4 + x_2 x_3 + \alpha^2 x_3^2 + \alpha x_3 x_4 + x_4^2 \end{pmatrix} \end{aligned}$$

and

$$E_2 = A \cdot C = \begin{pmatrix} \alpha^2 x_1 x_2 + x_1 x_4 + \alpha^2 x_2^2 + x_2 x_4 & \alpha x_1 x_3 + x_1 x_4 + \alpha^2 x_2^2 + \alpha x_2 x_3 + \alpha^2 x_2 x_4 \\ \alpha^2 x_2 x_3 + \alpha^2 x_2 x_4 + x_3 x_4 + x_4^2 & x_1 x_4 + x_2 x_3 + \alpha^2 x_2 x_4 + \alpha x_3^2 + \alpha^2 x_3 x_4 + \alpha^2 x_4^2 \end{pmatrix}$$

and set $\mathcal{F} = (f^{(1)}, \dots, f^{(8)}) = (E_{1,11}, E_{1,12}, E_{1,21}, E_{1,22}, E_{2,11}, E_{2,12}, E_{2,21}, E_{2,22})$.

Finally, we compute the public key $\mathcal{P} = (p^{(1)}, \dots, p^{(8)}) : \mathbb{F}^4 \rightarrow \mathbb{F}^8$ by $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$, obtaining

$$\begin{aligned}
p^{(1)} &= \alpha^2 x_1^2 + \alpha x_2^2 + x_2 x_4 + \alpha x_2 + \alpha^2 x_3^2 + \alpha x_3 x_4 + \alpha^2 x_3 + \alpha^2 x_4^2 + \alpha, \\
p^{(2)} &= \alpha x_1^2 + \alpha x_1 x_2 + \alpha x_1 x_3 + \alpha^2 x_1 x_4 + x_1 + x_2 x_4 + x_2 + \alpha^2 x_3 x_4 + \alpha x_3 \\
&\quad + x_4 + 1, \\
p^{(3)} &= \alpha x_1^2 + \alpha x_1 x_2 + \alpha^2 x_1 x_3 + \alpha x_1 x_4 + \alpha^2 x_1 + \alpha^2 x_2^2 + \alpha x_2 x_3 + x_2 x_4 + x_2 \\
&\quad + \alpha x_3^2 + x_3 x_4 + \alpha^2 x_3 + x_4^2 + \alpha^2 x_4, \\
p^{(4)} &= \alpha^2 x_1 x_2 + \alpha x_2^2 + \alpha^2 x_2 x_3 + \alpha^2 x_2 x_4 + x_3^2 + \alpha x_3 x_4 + \alpha^2 x_3 + x_4^2, \\
p^{(5)} &= \alpha x_1^2 + \alpha x_1 x_2 + \alpha^2 x_1 x_4 + x_1 + \alpha x_2 x_3 + x_2 x_4 + x_2 + x_3 x_4 + \alpha x_3 + x_4^2 \\
&\quad + x_4 + 1, \\
p^{(6)} &= \alpha^2 x_1^2 + x_1 x_2 + x_1 x_3 + x_2^2 + x_2 x_4 + x_3 x_4 + \alpha^2 x_4^2 + \alpha, \\
p^{(7)} &= \alpha x_1^2 + \alpha^2 x_1 x_3 + \alpha x_2^2 + \alpha x_2 x_3 + x_2 x_4 + \alpha x_2 + \alpha^2 x_3^2 + \alpha^2 x_3 x_4 + \alpha^2 x_4^2 + \alpha^2, \\
p^{(8)} &= x_1^2 + \alpha x_1 x_2 + x_1 x_3 + \alpha^2 x_2^2 + \alpha x_2 x_3 + \alpha^2 x_2 + \alpha x_3^2 + x_3 x_4 + x_4^2 + 1.
\end{aligned}$$

In order to encrypt a message $\mathbf{z} = (1, \alpha, \alpha^2, \alpha^2) \in \{1\} \times \mathbb{F}^3$, we just compute $\mathbf{w} = \mathcal{P}(\mathbf{z})$ and obtain

$$\mathbf{w} = (\alpha^2, \alpha, 1, 1, 0, 1, \alpha^2, \alpha) \in \mathbb{F}^8.$$

In order to decrypt the ciphertext $(\alpha^2, \alpha, 1, 1, 0, 1, \alpha^2, \alpha) \in \mathbb{F}^8$, we first compute

$$\mathbf{x} = \mathcal{S}^{-1}(\mathbf{w}) = (\alpha^2, \alpha, \alpha^2, \alpha^2, 0, \alpha^2, 0, 0) \in \mathbb{F}^8$$

and write the result into the matrices \bar{E}_1 and $\bar{E}_2 \in \mathbb{F}^{2 \times 2}$ as

$$\bar{E}_1 = \begin{pmatrix} \alpha^2 & \alpha \\ \alpha^2 & \alpha^2 \end{pmatrix} \quad \text{and} \quad \bar{E}_2 = \begin{pmatrix} 0 & \alpha^2 \\ 0 & 0 \end{pmatrix}.$$

After that, we have to invert the central map, i.e. we have to find a vector $\mathbf{y} \in \mathbb{F}^4$ such that $\mathcal{F}(\mathbf{y}) = \mathbf{x}$ holds. We set $W = \begin{pmatrix} w_1 & w_2 \\ w_3 & w_4 \end{pmatrix}$ to be the inverse of the (unknown) matrix $\bar{A} = A(\mathbf{y})$ and build the linear system given by the equations $W \cdot \bar{E}_1 - B = 0$ and $W \cdot \bar{E}_2 - C = 0$, obtaining

$$\begin{pmatrix} 0 & 0 & 0 & \alpha & \alpha^2 & \alpha^2 & 0 & 0 \\ 1 & 1 & \alpha^2 & 0 & \alpha & \alpha^2 & 0 & 0 \\ \alpha & 0 & \alpha & \alpha^2 & 0 & 0 & \alpha^2 & \alpha^2 \\ \alpha^2 & 0 & \alpha & 1 & 0 & 0 & \alpha & \alpha^2 \\ 0 & \alpha^2 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & \alpha & 1 & \alpha^2 & 0 & 0 & 0 \\ 0 & \alpha^2 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & \alpha^2 & \alpha & \alpha^2 & 0 & 0 & \alpha^2 & 0 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

The rank of this system is 7. Therefore, after eliminating the variables w_1, \dots, w_4 , we get 3 linear equations in the 4 variables y_1, \dots, y_4 . By solving these equations we obtain

$$\mathbf{y} = k \cdot (1, \alpha, \alpha^2, 1) \text{ with } k \in \mathbb{F}.$$

Finally, we have to compute $\mathbf{z}^{(k)} = \mathcal{T}^{-1}(\mathbf{y}^{(k)})$ for each $k \in \mathbb{F}$. Only for $k = \alpha^2$, the first component of the vector $\mathbf{z}^{(k)}$ is 1, which guarantees that

$$\mathbf{z}^{(\alpha^2)} = (1, \alpha, \alpha^2, \alpha^2)$$

is the encrypted plaintext.

7.2 The Rectangular SimpleMatrix Encryption Scheme

A natural extension of the SimpleMatrix encryption scheme is the rectangular SimpleMatrix encryption scheme proposed by Tao et al. in [5]. Instead of using square matrices A , B and C , the rectangular SimpleMatrix scheme allows these matrices to be rectangular. It therefore introduces a lot of new parameters, which make the scheme more flexible and allow the user to find a balance between security, efficiency and the probability of a decryption failure. The scheme can be described as follows.

Key Generation Let \mathbb{F} be a finite field with q elements and $r, s, u, m, n \in \mathbb{N}$ be integers satisfying $m = 2su$. For the efficiency of the decryption process we furthermore choose the number n of variables such that $(n - r(2u - s)) \cdot (n - r(2u - s) + 1) \leq 2m$. We set

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1r} \\ a_{21} & a_{22} & \dots & a_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ a_{s1} & a_{s2} & \dots & a_{sr} \end{pmatrix}, \quad B = \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1u} \\ b_{21} & b_{22} & \dots & b_{2u} \\ \vdots & \vdots & \ddots & \vdots \\ b_{r1} & b_{r2} & \dots & b_{ru} \end{pmatrix}, \quad C = \begin{pmatrix} c_{11} & c_{12} & \dots & c_{1u} \\ c_{21} & c_{22} & \dots & c_{2u} \\ \vdots & \vdots & \ddots & \vdots \\ c_{r1} & c_{r2} & \dots & c_{ru} \end{pmatrix},$$

where A is an $s \times r$ matrix, whereas B and C are $r \times u$ matrices containing randomly chosen linear combinations of the variables x_1, \dots, x_n .

As in the case of the standard SimpleMatrix encryption scheme, we define $E_1 = A \cdot B$ and $E_2 = A \cdot C$. The *central map* \mathcal{F} of our scheme consists of the $m = 2su$ components of the matrices E_1 and E_2 . Note that each of these components is a homogeneous quadratic polynomial in $\mathbb{F}[x_1, \dots, x_n]$.

Additionally one chooses two invertible linear maps $\mathcal{S} : \mathbb{F}^m \rightarrow \mathbb{F}^m$ and $\mathcal{T} : \mathbb{F}^n \rightarrow \mathbb{F}^n$.

Public Key $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T} : \mathbb{F}^n \rightarrow \mathbb{F}^m$.

Private Key The private key consists of

- the matrices A , B , and C and
- the affine maps \mathcal{S} and \mathcal{T} .

Encryption For a message $\mathbf{z} = (z_1, z_2, \dots, z_n)$, the corresponding ciphertext $\mathbf{w} \in \mathbb{F}^m$ is given by $\mathbf{w} = \mathcal{P}(\mathbf{z})$.

Decryption To decrypt the ciphertext $\mathbf{w} = (w_1, w_2, \dots, w_m)$, the owner of the private key performs the following steps:

1. Compute $\mathbf{x} = (x_1, x_2, \dots, x_m) = \mathcal{S}^{-1}(\mathbf{w})$ and set

$$\bar{E}_1 = \begin{pmatrix} x_1 & x_2 & \dots & x_u \\ x_{u+1} & x_{u+2} & \dots & x_{2u} \\ \vdots & \vdots & \ddots & \vdots \\ x_{(s-1)u+1} & x_{(s-1)u+2} & \dots & x_{su} \end{pmatrix} \in \mathbb{F}^{s \times u};$$

$$\bar{E}_2 = \begin{pmatrix} x_{su+1} & x_{su+2} & \dots & z_{(s+1)u} \\ z_{(s+1)u+1} & z_{(s+1)u+2} & \dots & z_{(s+2)u} \\ \vdots & \vdots & \ddots & \vdots \\ z_{(2s-1)u+1} & z_{(2s-1)u+2} & \dots & z_{2su} \end{pmatrix} \in \mathbb{F}^{s \times u}.$$

Let $\bar{A} = A(\mathbf{y})$, where $\mathbf{y} = (y_1, \dots, y_n)$ is the (so far unknown) pre-image of \mathbf{x} under the central map \mathcal{F} .

- If the rank of \bar{A} is r , then there exists an $r \times s$ matrix W such that $W \cdot \bar{A} = I_r$, where I_r is the $r \times r$ identity matrix. From $\bar{E}_1 = \bar{A} \cdot B$ and $\bar{E}_2 = \bar{A} \cdot C$ we get $W \cdot \bar{E}_1 = W \cdot \bar{A} \cdot B$, $W \cdot \bar{E}_2 = W \cdot \bar{A} \cdot C$ and therefore $W \cdot \bar{E}_1 = B$ and $W \cdot \bar{E}_2 = C$. We interpret the elements of W as new variables and end up with $2ru$ linear equations in $sr + n$ unknowns. Then we eliminate the sr elements of W from these equations. We obtain roughly $r(2u - s)$ linear equations in the variables y_1, y_2, \dots, y_n .

The dimension of the solution space of this system is usually very small. Solving this system by Gaussian elimination enables us to eliminate most of the unknowns, say Z of them. Then we write these Z variables as linear combinations of the remaining unknown variables and substitute these equations into the central polynomials. By doing so, we obtain a new system of m quadratic equations in the remaining $n - Z$ unknowns. When the number

$n - Z$ is small enough, we can solve this system easily by the Relinearization algorithm of [3] (see below).

- In the case of $\text{Rank}(\bar{A}) < r$, decryption remains an open problem.

2. Compute the plaintext $\mathbf{z} = (z_1, z_2, \dots, z_n) = \mathcal{T}^{-1}(\mathbf{y})$.

7.2.1 Solving the Quadratic Systems

During the second step of the decryption process, we have to solve a system of quadratic equations. This system consists of m equations in about $n - r(2u - s)$ variables. If the condition

$$(n - r(2u - s))(n - r(2u - s) + 1) \leq 2m \quad (7.1)$$

is fulfilled, we can solve this system easily by the Relinearization technique [3].

1. Interpret each quadratic monomial $x_i x_j$ as a new variable x_{ij} .
2. Solve the resulting linear system by Gaussian elimination.

If the condition (7.1) is fulfilled, the linear system will have exactly one solution which coincides with the solution of the quadratic system. The Relinearization technique therefore enables us to find the solution of highly overdetermined systems in polynomial time.

7.2.2 Probability of Decryption Failures

In the case of the rectangular SimpleMatrix encryption scheme, a decryption failure occurs if and only if the rank of the matrix \bar{A} is less than r . The probability of this can be computed by

$$1 - \left(1 - \frac{1}{q^s}\right)\left(1 - \frac{1}{q^{s-1}}\right) \cdots \left(1 - \frac{1}{q^{s-r+1}}\right) \approx \frac{1}{q^{s-r+1}},$$

Therefore, the probability of a decryption failure occurring can be estimated by q^{r-s-1} . By choosing the parameters s and r of our scheme in an appropriate way, we can therefore reduce the probability of decryption failures to a negligible value.

7.2.3 Key Sizes and Efficiency

The public key size of the rectangular SimpleMatrix encryption scheme is given as

$$\text{size}_{\text{pk rectSM}} = 2su \frac{(n+1)(n+2)}{2}$$

field elements, the size of the private key is

$$\text{size}_{\text{sk rectSM}} = \underbrace{n(n+1)}_{\mathcal{T}} + \underbrace{2su(2su+1)}_{\mathcal{S}} + \underbrace{(sr+2ru)(n+1)}_{\mathcal{F}}$$

\mathbb{F} -elements. In order to decrypt a ciphertext we have to solve only systems of linear equations, just like in the case of the standard SimpleMatrix encryption scheme. However, there we had a system of size $m \times m$, but now we deal with significantly smaller systems.

Since the running time of Gaussian elimination is cubic in the size of the linear system, this leads to a significant speed up in running time. Furthermore we note that, while, for the standard SimpleMatrix scheme, the ratio between plain and ciphertext length was fixed to two, the rectangular SimpleMatrix scheme allows significantly smaller blow up factors.

7.2.4 Toy Example

In this section we demonstrate the workflow of the rectangular SimpleMatrix scheme. However, to reduce the dimensions of the public key, we choose the size of the matrices A, B and C as in the case of the standard SimpleMatrix scheme. Therefore, the main purpose of this toy example is to demonstrate the new decryption algorithm used for rectangular SimpleMatrix.

Let $\mathbb{F} = \text{GF}(4)$ and $r = s = u = 2$. Therefore, the number of equations in the public system is given by $m = 2su = 8$. However, the number of variables is no longer fixed to $n = m/2$, but we set $n = 5$ (i.e. we have a smaller blow up factor).

We choose the affine maps $\mathcal{S} : \mathbb{F}^8 \rightarrow \mathbb{F}^8$ and $\mathcal{T} : \mathbb{F}^5 \rightarrow \mathbb{F}^5$ as

$$\mathcal{S}(x_1, \dots, x_8) = \begin{pmatrix} \alpha^2 & \alpha & 0 & \alpha & 0 & 0 & \alpha^2 & \alpha^2 \\ 0 & 1 & \alpha & 1 & 0 & \alpha & \alpha^2 & \alpha \\ 0 & 1 & 1 & \alpha & \alpha & 0 & \alpha^2 & \alpha \\ \alpha^2 & 1 & 0 & \alpha^2 & \alpha & \alpha & \alpha^2 & 0 \\ \alpha^2 & \alpha & 0 & \alpha & \alpha^2 & \alpha^2 & 0 & \alpha^2 \\ 0 & \alpha^2 & \alpha^2 & 1 & \alpha^2 & \alpha & 0 & \alpha \\ \alpha & \alpha & \alpha & \alpha^2 & \alpha & \alpha & \alpha^2 & \alpha^2 \\ 1 & 1 & 0 & 1 & \alpha^2 & \alpha & \alpha & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \\ \alpha \\ \alpha^2 \\ 1 \\ \alpha^2 \\ 0 \end{pmatrix}.$$

and

$$\mathcal{T}(x_1, \dots, x_5) = \begin{pmatrix} \alpha^2 & 1 & \alpha & \alpha & \alpha \\ \alpha^2 & \alpha & \alpha^2 & \alpha & 1 \\ 1 & \alpha & \alpha^2 & \alpha^2 & 1 \\ 1 & 1 & \alpha & \alpha & 1 \\ \alpha & 1 & 0 & \alpha & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} + \begin{pmatrix} \alpha^2 \\ \alpha^2 \\ 1 \\ 0 \\ \alpha \end{pmatrix}.$$

The matrices A , B and $C \in \mathbb{F}^{2 \times 2}$ are chosen as

$$\begin{aligned} A &= \begin{pmatrix} x_1 + \alpha x_2 + x_3 + \alpha x_4 & \alpha x_1 + \alpha^2 x_3 + \alpha^2 x_4 \\ \alpha x_1 + x_2 + \alpha x_4 + \alpha x_5 & x_1 + x_2 \end{pmatrix}, \\ B &= \begin{pmatrix} \alpha x_1 + \alpha^2 x_2 + \alpha^2 x_3 + \alpha x_4 + \alpha^2 x_5 & \alpha x_1 + \alpha x_3 + x_4 + x_5 \\ \alpha x_1 + \alpha^2 x_2 + \alpha x_3 + x_4 + x_5 & \alpha x_1 + x_2 + \alpha x_3 + \alpha x_4 + \alpha^2 x_5 \end{pmatrix}, \\ C &= \begin{pmatrix} \alpha^2 x_1 + x_3 + \alpha^2 x_5 & \alpha^2 x_1 + \alpha^2 x_2 + \alpha^2 x_3 + \alpha^2 x_4 + x_5 \\ x_1 + \alpha x_2 + \alpha x_4 + \alpha^2 x_5 & x_1 + x_2 + \alpha^2 x_4 + x_5 \end{pmatrix}. \end{aligned}$$

We compute $E_1 = A \cdot B$, $E_2 = A \cdot C$, $\mathcal{F} = (E_{1,1,1}, E_{1,1,2}, E_{1,2,1}, E_{1,2,2}, E_{2,1,1}, E_{2,1,2}, E_{2,2,1}, E_{2,2,2})$ and $\mathcal{P} = (p^{(1)}, \dots, p^{(8)}) = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$. We get

$$\begin{aligned} p^{(1)} &= \alpha x_1^2 + \alpha^2 x_1 x_2 + x_1 x_3 + \alpha^2 x_1 x_5 + x_2^2 + x_2 x_4 + \alpha^2 x_2 x_5 \\ &\quad + \alpha x_3 x_4 + \alpha x_4 x_5 + x_5 + \alpha, \\ p^{(2)} &= \alpha^2 x_1 x_2 + \alpha x_1 + x_2 x_3 + \alpha^2 x_2 x_5 + \alpha^2 x_2 + \alpha x_3^2 + \alpha x_3 x_4 + x_3 x_5 \\ &\quad + x_3 + x_4^2 + \alpha^2 x_4 + \alpha x_5^2 + 1, \\ p^{(3)} &= x_1^2 + x_1 x_3 + x_1 x_4 + \alpha^2 x_1 + x_2^2 + x_2 x_3 + \alpha^2 x_2 x_4 + \alpha x_2 x_5 + x_2 + x_3 + \alpha x_4^2 \\ &\quad + x_5 + 1, \\ p^{(4)} &= \alpha^2 x_1^2 + \alpha x_1 x_2 + x_1 x_4 + x_2^2 + x_2 x_3 + \alpha x_2 x_4 + \alpha^2 x_2 + \alpha^2 x_3 x_4 \\ &\quad + \alpha^2 x_3 x_5 + \alpha^2 x_4^2 + \alpha x_4 x_5 + \alpha x_4 + \alpha x_5^2 + \alpha^2 x_5, \\ p^{(5)} &= x_1^2 + x_1 x_2 + \alpha^2 x_1 x_3 + x_1 x_4 + x_1 x_5 + \alpha^2 x_1 + \alpha^2 x_2^2 + \alpha x_2 x_3 + x_2 x_4 + x_2 x_5 \\ &\quad + \alpha x_3 x_4 + x_3 x_5 + \alpha x_3 + \alpha^2 x_4^2 + x_4 x_5 + \alpha x_4 + \alpha^2 x_5^2 + \alpha x_5 + 1, \\ p^{(6)} &= \alpha x_1^2 + x_1 x_2 + \alpha x_1 x_3 + \alpha^2 x_1 x_4 + \alpha^2 x_1 + \alpha^2 x_2 x_3 + \alpha x_2 x_5 + \alpha^2 x_2 + x_3 \\ &\quad + \alpha x_4 x_5 + x_4 + \alpha x_5^2 + \alpha^2 x_5 + \alpha, \\ p^{(7)} &= \alpha^2 x_1 x_2 + x_1 x_3 + \alpha^2 x_1 x_4 + \alpha^2 x_1 x_5 + \alpha^2 x_1 \alpha^2 x_2^2 + x_2 x_3 + \alpha^2 x_2 x_4 \\ &\quad + \alpha^2 x_3^2 + \alpha x_3 x_4 + \alpha^2 x_3 x_5 + \alpha^2 x_3 + x_4 x_5 + \alpha^2 x_5^2 + x_5 + 1, \\ p^{(8)} &= \alpha x_1^2 + \alpha^2 x_1 x_2 + \alpha^2 x_1 x_3 + \alpha^2 x_1 x_4 + \alpha^2 x_2^2 \alpha x_2 x_4 + \alpha^2 x_2 x_5 + \alpha^2 x_2 \\ &\quad + \alpha^2 x_3^2 + \alpha^2 x_3 x_4 + \alpha^2 x_3 x_5 + \alpha x_3 + \alpha x_4^2 + x_4 x_5 + x_4 + \alpha x_5^2 + \alpha x_5 \end{aligned}$$

We want to encrypt the message $\mathbf{z} = (\alpha, \alpha, \alpha^2, 1, \alpha^2) \in \mathbb{F}^5$. We get

$$\mathbf{w} = \mathcal{P}(\mathbf{z}) = (\alpha, 1, 1, \alpha^2, \alpha^2, 0, 1, \alpha^2) \in \mathbb{F}^8.$$

In order to decrypt the ciphertext $\mathbf{w} = (\alpha, 1, 1, \alpha^2, \alpha^2, 0, 1, \alpha^2) \in \mathbb{F}^8$, we compute

$$S^{-1} = \begin{pmatrix} \alpha & 1 & 0 & \alpha^2 & \alpha^2 & 0 & 1 & \alpha \\ \alpha & 1 & \alpha^2 & 0 & \alpha & 1 & 1 & \alpha \\ 0 & 1 & \alpha^2 & 1 & \alpha^2 & 1 & \alpha & \alpha \\ \alpha & \alpha^2 & 1 & 1 & 0 & \alpha^2 & 1 & 0 \\ 0 & \alpha^2 & \alpha^2 & \alpha & \alpha^2 & 0 & 1 & 1 \\ \alpha^2 & \alpha^2 & \alpha^2 & \alpha & \alpha & \alpha^2 & 0 & \alpha \\ 1 & 0 & 0 & 0 & \alpha^2 & \alpha^2 & 1 & \alpha^2 \\ 1 & 0 & 1 & 0 & 1 & \alpha & 0 & 1 \end{pmatrix}$$

and

$$\mathbf{x} = S^{-1}(\alpha - c_S) = (0, 1, 0, \alpha^2, \alpha^2, \alpha, 0, 0).$$

We define the matrices \bar{E}_1 and $\bar{E}_2 \in \mathbb{F}^{2 \times 2}$ as

$$\bar{E}_1 = \begin{pmatrix} 0 & 1 \\ 0 & \alpha^2 \end{pmatrix} \quad \text{and} \quad \bar{E}_2 = \begin{pmatrix} \alpha^2 & \alpha \\ 0 & 0 \end{pmatrix}$$

and consider the relations $W \cdot \bar{E}_1 = B$ and $W \cdot \bar{E}_2 = C$ (for an unknown matrix $W = \bar{A}^{-1}$). After eliminating the elements of the matrix W from these equations, we find 3 linearly independent equations in the 5 plaintext variables

$$\begin{aligned} y_2 + y_3 &= 0 \\ \alpha^2 y_1 + \alpha^2 y_2 + y_4 &= 0 \\ y_1 + \alpha y_2 + y_5 &= 0 \end{aligned} \tag{7.2}$$

After substituting these equations into the public key, we get

$$\begin{aligned} \text{from } p^{(1)} : 0 &= 0 \\ \text{from } p^{(2)} : \alpha y_2^2 &= 1 \\ \text{from } p^{(3)} : 0 &= 0 \\ \text{from } p^{(4)} : y_2^2 &= \alpha^2 \\ \text{from } p^{(5)} : \alpha^2 y_1 y_2 + \alpha y_2^2 &= \alpha^2 \end{aligned}$$

$$\text{from } p^{(6)} : \alpha y_1 y_2 + y_2^2 = \alpha$$

$$\text{from } p^{(7)} : \alpha^2 y_1^2 + y_1 y_2 + \alpha y_2^2 = 0$$

$$\text{from } p^{(8)} : \alpha y_1^2 + \alpha^2 y_1 y_2 + y_2^2 = 0$$

We interpret y_1^2 , $y_1 y_2$ and y_2^2 as new variables and solve the resulting linear system by Gaussian elimination. We get

$$(y_1^2, y_1 y_2, y_2^2, y_1, y_2) = (\alpha^2, \alpha^2, \alpha^2, \alpha, \alpha)$$

which corresponds to $(y_1, y_2) = (\alpha, \alpha)$. By substituting this result into (7.2), we get $(y_3, y_4, y_5) = (\alpha, 0, 1)$. Altogether, we find

$$\mathbf{y} = \mathcal{F}^{-1}(\mathbf{x}) = (\alpha, \alpha, \alpha, 0, 1).$$

Finally, we have to invert the second affine map \mathcal{T} . We get

$$T^{-1} = \begin{pmatrix} 1 & 0 & 1 & \alpha^2 & 0 \\ 1 & \alpha & \alpha^2 & \alpha^2 & 1 \\ 1 & 0 & 0 & \alpha & \alpha^2 \\ \alpha & 1 & \alpha^2 & 1 & 0 \\ 1 & 0 & \alpha^2 & 0 & 0 \end{pmatrix}$$

and

$$\mathbf{z} = T^{-1}(\mathbf{y} - c_T) = (\alpha, \alpha, \alpha^2, 1, \alpha^2) \in \mathbb{F}^5.$$

7.3 Attacks on SimpleMatrix

In order to find secure parameters for the SimpleMatrix encryption scheme, we basically have to consider two different attacks:

- the direct attack and
- rank based attacks

7.3.1 Direct Attack

The most straightforward way to attack a multivariate encryption scheme such as SimpleMatrix is the direct attack. For this, an attacker considers the public system

$$\mathcal{P}(z_1, \dots, z_n) = \mathbf{w}$$

as an instance of the MQ Problem and tries to solve it using an algorithm like XL, Mutant XL or F_4 .

Experiments [4, 5] have shown that the public systems of the SimpleMatrix scheme and its variants can be solved faster than random systems. The reason for this is the relatively low degree of regularity of these systems. A conservative assertion is that the degree of regularity of a direct attack against SimpleMatrix is at least $r + 1$.

This assertion can be supported by the fact that even the holder of the private key in some sense needs to deal with polynomials of this degree due to the involvement of the inverse of matrix \bar{A} .

In order to defend the scheme against direct attacks, we therefore need significantly more equations than for the public systems like UOV and Rainbow. This is also due to the fact that we deal here with overdetermined systems to ensure the public key \mathcal{P} to be injective.

7.3.2 Rank Attacks

In general, there are two different flavors of the rank attack.

The first one is called the MinRank attack or LowRank attack as proposed by Goubin et al. in [2]. The other one is called the HighRank Attack [1]. In this section we analyze the complexity of these two attacks against the SimpleMatrix encryption scheme.

In the case of the SimpleMatrix schemes, the components of the central map \mathcal{F} and the public key \mathcal{P} are homogeneous quadratic polynomials in $\mathbb{F}[x_1, \dots, x_n]$. Let Q_i and \bar{Q}_i ($i = 1, \dots, m$) be the symmetric matrices associated to the i -th component of \mathcal{F} and \mathcal{P} respectively (see Sect. 5.2). Note that, for underlying fields of characteristic 2, the diagonal elements of these matrices are 0. In the case of the SimpleMatrix scheme, the rank of the matrices Q_i is obviously bounded by $2r$.

The goal of the MinRank attack is to find a vector $\mathbf{t} = (t_1, t_2, \dots, t_m) \in \mathbb{F}^m$ such that the rank of the matrix $\tilde{Q} = \sum_{i=1}^m t_i \bar{Q}_i$ is less or equal to $2r$. Such a matrix \tilde{Q} corresponds to a central polynomial. By finding m of these matrices of low rank the attacker can therefore recover the affine map S and therefore the private key of the scheme.

In order to find such a matrix \tilde{Q} of low rank, the attacker can apply any of the MinRank algorithms discussed in Sect. 4.2.

The complexity of the MinRank attack against the SimpleMatrix scheme can be estimated by

$$\text{Complexity}_{\text{MinRank SimpleMatrix}} = \mathcal{O}(q^{\lceil \frac{m}{n} \rceil 2r m^3}) = \mathcal{O}(q^{4r r^6}). \quad (7.3)$$

For the HighRank Attack, we form an arbitrary linear combination

$$\tilde{Q} = \sum_{i=1}^m \alpha_i \bar{Q}_i \text{ and find } V = \text{Ker}(\tilde{Q}).$$

If \tilde{Q} has a nontrivial kernel, we set $(\sum_{i=1}^m \lambda_i \tilde{Q}_i) \cdot V = 0$ and check if the solution set \hat{V} for the λ_i has dimension $n - 2r$. If this is true, V is, with certain probability, a subspace of $\mathcal{T}^{-1}(\mathcal{O})$ where $\mathcal{O} = \{\mathbf{x} = (x_1, \dots, x_n) : x_1 = \dots = x_{n-2r} = 0\}$.

We can therefore use the attack to recover the secret linear transformation \mathcal{T} and with it the private key of our scheme. The complexity of the HighRank attack can be estimated by

$$\text{Complexity}_{\text{HighRank SimpleMatrix}} = \mathcal{O}(n^6 q^{2r}) = \mathcal{O}(r^{12} q^{2r}). \quad (7.4)$$

Therefore, for carefully chosen parameters, Rank attacks against the Simple Matrix scheme are highly impractical.

We also would like to point out that, as stated in [5], linearization and higher order linearization equations (HOLE) attacks do not apply to SimpleMatrix schemes. Recently there are publications claiming polynomial time attacks on all SM schemes, but our analysis of the paper shows that they are not correct and this is further confirmed by the fact that none of the papers have any practical implementation of the attacks.

7.3.3 Practical Parameters

In this section we propose practical parameters for the (rectangular) Simple Matrix scheme leading to schemes which are secure against the attacks discussed above. For this, we choose $\text{GF}(2^{16})$ for the underlying field.

For the rectangular SimpleMat-rix encryption scheme we furthermore choose the parameters in such a way that the probability of a decryption failure is less than 2^{-40} . Thus we get $s = r + 2$. In order to enable efficient decryption, we choose $u = r + 2$. Tables 7.1 and 7.2 show the resulting key and ciphertext sizes.

Table 7.1 Parameters and key sizes of the SimpleMatrix encryption scheme

Security category	Parameters (q, r, m, n)	Public key size (MB)	Private key size (MB)	Plaintext size (byte)	Ciphertext size (byte)	Probability of decr. failure
I	$(2^{16}, 12, 288, 144)$	5.8	0.28	288	576	2^{-16}
II	$(2^{16}, 16, 512, 256)$	32.4	0.90	512	1024	2^{-16}

Table 7.2 Parameters and key sizes of the rectangular SimpleMatrix encryption scheme

Security category	Parameters (q, r, s, u, m, n)	Public key size (MB)	Private key size (MB)	Plaintext size (byte)	Ciphertext size (byte)	Probability of decr. failure
I	(2^{16} , 12, 14, 14, 392, 196)	14.6	0.38	392	784	2^{-48}
II	(2^{16} , 16, 18, 18, 648, 324)	65.5	1.03	648	1296	2^{-48}

References

1. D. Coppersmith, J. Stern, S. Vaudenay, Attacks on the birational signature scheme, in *Advances in Cryptology — CRYPTO 1994*. Lecture Notes in Computer Science, vol. 773 (Springer, Berlin, 1994), pp. 435–443

2. L. Goubin, N. Courtois, Cryptanalysis of the TTM cryptosystem, in *Advances in Cryptology — ASIACRYPT 2000*. Lecture Notes in Computer Science, vol. 1976 (Springer, Berlin, 2000), pp. 44–57

3. A. Kipnis, A. Shamir, Cryptanalysis of the HFE public key cryptosystem by relinearization, in *Advances in Cryptology — CRYPTO 1999*. Lecture Notes in Computer Science, vol. 1666 (Springer, Berlin, 1999), pp. 19–30

4. C. Tao, A. Diene, S. Tang, J. Ding, Simple matrix scheme for encryption, in *Advances in Cryptology — PQCrypto 2013*. Lecture Notes in Computer Science, vol. 7932 (Springer, Berlin, 2013), pp. 231–242

5. C. Tao, H. Xiang, A. Petzoldt, J. Ding, Simple matrix—a multivariate public key cryptosystem (MPKC) for encryption. *Finite Fields Th. App.* **35**, 352–368 (2015)

Chapter 8

Solving Polynomial Systems



Abstract This chapter considers the known techniques to solve (systems of) nonlinear polynomial equations. After giving a historical overview of the topic, we describe algorithms to solve univariate polynomials of high degree. The remainder of the chapter deals with algorithms to solve systems of nonlinear multivariate polynomials. We describe the XL algorithm, give a short introduction into the theory of Gröbner bases and present the most important algorithms to compute these bases. After analyzing the complexity of these algorithms against various types of multivariate polynomial systems, we end this chapter by giving an overview of the known algorithms used to solve over and underdetermined systems of multivariate quadratic equations.

In this chapter we discuss various methods to solve (systems of) polynomial equations. As already discussed in previous chapters of this book, this task is not only important in multivariate cryptography, but also in many other areas. In fact, nearly every cryptosystem can be written using systems of multivariate quadratic equations and therefore can be attacked by the methods described in this chapter. Especially in the cryptanalysis of symmetric schemes such as block and stream ciphers, this so called algebraic cryptanalysis plays an important role. But also in many fields outside cryptography such as coding and representation theory, as well as in the analysis of biological and chemical models, the methods discussed in this chapter play an important role.

After giving a quick overview of the long history of this field, we describe two algorithms to solve univariate polynomials of high degree over finite fields. These algorithms, Berlekamp's algorithm [4] and the Cantor–Zassenhaus algorithm [7], are used during the inversion of the central map of HFE and its variants (see Chap. 4).

The second part of this chapter deals with algorithms to solve systems of multivariate quadratic polynomials over finite fields. These systems appear in direct attacks against multivariate schemes, in which one tries to solve the equation $\mathcal{P}(\mathbf{z}) = \mathbf{w}$ directly as an instance of the MQ Problem as well as the UOV Reconciliation and the Rainbow-Band-Separation attacks (see Chap. 5). However, the discussed

algorithms are also used in many other fields such as the algebraic cryptanalysis of symmetric ciphers and membership and equality checks in the theory of ideals. A central notion in this area is that of Gröbner bases, which can be seen as a standard basis of the ideal generated by a polynomial system. After having computed a Gröbner basis of the corresponding ideal in lexicographic order, the solution of the nonlinear system can be found easily. In this chapter, we give a short introduction into the theory of Gröbner bases and describe the most important algorithms to compute these bases.

We finish the chapter by describing various algorithms to solve overdetermined ($m \gg n$) and underdetermined ($m \ll n$) quadratic systems. As we will see, these systems can be solved much faster than systems with $m \sim n$.

The chapter is organized as follows: In Sect. 8.1 we give a short overview of the history of solving (systems of) polynomial equations. In Sect. 8.2 we discuss two algorithms for finding roots of univariate polynomials of high degree, which can be used to invert the central map of HFE and its variants. Section 8.3 describes the XL-Algorithm for solving multivariate quadratic systems and discusses some of its variations. In Sect. 8.4 we introduce the concept of Gröbner Bases, while Sect. 8.5 discusses the most important algorithms used to compute these bases. In Sect. 8.6 we deal with the complexity of these algorithms against random systems and systems of the HFE type. Finally, Sect. 8.7 discusses techniques to solve over- and underdetermined systems of multivariate quadratic equations.

8.1 History of Solving Polynomial Equations

In this section we give a short overview of the history of solving (systems of) polynomial equations, see Table 8.1. While the research in the univariate case was basically finished at around 1820, many of the results in the multivariate case are quite new and have been found only in the second half of the last century.

Table 8.1 History of solving (systems of) polynomial equations

Degree D	Univariate case	Multivariate case
1		China (~ 1 AD)
	Trivial	Gaussian elimination (1810)
2	Babylonia (~ 800 BC)	XL-Algorithm
3	Cardano-Tartaglia (1545)	
4	Cardano-Ferrari (1545)	
≥ 5	No explicit formula	Gröbner Basis techniques
	(Theorem of Abel-Ruffino, Galois Theory ~ 1820)	(Buchberger’s algorithm (1965), F_4 (1999), F_5 (2002))
	Berlekamp’s algorithm (1967)	(exponential running time)
	Cantor–Zassenhaus (1981) (exponential in d)	

8.1.1 Solving Nonlinear Univariate Polynomial Equations

While solving a linear polynomial equation in one variable is a trivial task, solving nonlinear polynomial equations requires some mathematical understanding. In the quadratic case, this problem was first solved in ancient Babylon at around 800 BC with a method called quadratic extension.

Let $f(x) = x^2 + ax + b$ be a monic quadratic polynomial. In order to solve the equation $f(x) = 0$, we extend the left side of the equation to a real square, i.e.

$$\begin{aligned} x^2 + ax + b &= 0 \\ x^2 + ax + \frac{a^2}{4} &= -b + \frac{a^2}{4} \\ \left(x + \frac{a}{2}\right)^2 &= -b + \frac{a^2}{4} \\ x_{1,2} &= -\frac{a}{2} \pm \sqrt{\frac{a^2}{4} - b}, \end{aligned}$$

which leads to the famous solution formula for quadratic equations. This formula is an example for solving a polynomial equation by **radicals**, i.e. an algorithm consisting of the operations addition, subtraction, multiplication, division and root taking.

Starting from about 1450, many mathematicians tried to extend this idea to polynomials of higher degree. The first result was obtained by the Italian mathematician Tartaglia, who succeeded in reducing the cubic polynomial equation to the quadratic case. Although Tartaglia wanted to keep his method secret, it was leaked to Cardano who published it in his book *Ars Magna* of 1545 in a more general form.

Theorem 8.1 (Cardano-Tartaglia) Let $f(x) = x^3 + ax^2 + bx + c$. Set

$$p = b - \frac{a^2}{3} \quad \text{and} \quad q = \frac{2}{27}a^3 - \frac{1}{3}ab + c.$$

The three solutions of the equation $f(x) = 0$ are given by

$$x_1 = u + v - \frac{a}{3}, \quad x_2 = \zeta_3 u + \zeta_3^2 v - \frac{a}{3}, \quad x_3 = \zeta_3^2 u + \zeta_3 v - \frac{a}{3},$$

where

$$u = \sqrt[3]{-\frac{q}{2} + \sqrt{\frac{q^2}{4} + \frac{p^3}{9}}}, \quad v = \sqrt[3]{-\frac{q}{2} - \sqrt{\frac{q^2}{4} + \frac{p^3}{9}}} \quad (8.1)$$

and $\zeta_3 = -\frac{1}{2} + \frac{i}{2}\sqrt{3}$ is the third root of unity.

Proof Set $x = z - \frac{a}{3}$ so that $f(x) = 0$ becomes

$$z^3 + pz + q = 0.$$

Note that $(u + v)^3 - 3uv(u + v) - u^3 - v^3 = 0$. Set $z = u + v$ and compare coefficients in the two cubic equations in order to obtain

$$p = -3uv, \quad q = -u^3 - v^3.$$

This leads to the quadratic equation $u^6 + qu^3 - \frac{p^3}{27} = 0$ for u^3 . It has the solutions $u^3 = -\frac{q}{2} \pm \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}$. Also v^3 has the same solutions. Choose opposite signs for u and v and take the cubic roots in order to arrive at the above formulas.

With the discriminant $D = q^2/4 + p^3/27$ one sees that $f(x) = 0$ has three real roots when $D < 0$, but has only one real root for $D > 0$. For $D < 0$ the method works well, despite the fact that it requires the use of complex numbers. When $D > 0$ the formulas as given may not provide the correct answer. The reason is that x_1 should be the real solution. When $D < 0$ this is the case since then u and v as given in (8.1) are conjugate to each other. When $D > 0$ this is not guaranteed. Instead one has to select \bar{u} from $u, \zeta_3 u, \zeta_3^2 u$ and \bar{v} from $v, \zeta_3 v, \zeta_3^2 v$ so that $x_1 = \bar{u} + \bar{v} - a/3$ is real. \square

In his book, Cardano also published a solution method for quartic polynomial equations, which was found by his student Ferrari. Similar to Tartaglia's strategy for the cubic case (reducing a cubic polynomial to a quadratic one), this formula was obtained by reducing the quartic to the cubic case.

The first proof that this method can not be extended to polynomials of degree ≥ 5 was given by Abel and Ruffino in 1799/1824.¹

However, further insight into the problem was given by the work of Évariste Galois. In order to understand his result, we need a number of definitions.

Definition 8.2 If \mathbb{E}/\mathbb{F} is a field extension, the **Galois group** of \mathbb{E}/\mathbb{F} , denoted by $\text{Gal}(\mathbb{E}/\mathbb{F})$, is the set of \mathbb{F} -automorphisms of \mathbb{E} .

Definition 8.3 Let $f(x)$ be a univariate polynomial over $\mathbb{F}[x]$. The **splitting field** \mathbb{S}_f is the smallest extension field of \mathbb{F} in which $f(x)$ can be written as a product of linear factors.

¹The proof given by Ruffino in 1799 was incomplete and corrected by Abel in 1824.

Definition 8.4 The group G is said to be **solvable** if there exists a series

$$1 = G^{(r)} \trianglelefteq G^{(r-1)} \trianglelefteq \dots \trianglelefteq G^{(0)} = G,$$

where each of the $G^{(i)}$ is a normal subgroup of its successor.

Thus we get

Theorem 8.5 (Galois) *Let $f(x)$ be a polynomial over a field of characteristic 0. If f is solvable by radicals, then the Galois group of the field extension \mathbb{S}_f/\mathbb{F} is a solvable group.*

Since the alternating group A_5 is not solvable, this theorem shows that general polynomial equations of degree 5 and higher can not be solved by radicals.

In order to find the roots of polynomials of degree ≥ 5 , one therefore has to use other techniques. Over fields of characteristic 0, we can use for example Newton's method to find approximate solutions. Over finite fields, there exist with Berlekamp's algorithm and the Cantor–Zassenhaus algorithm two efficient algorithms to solve this problem. Since these two algorithms are important for the inversion of the central map of the HFE cryptosystem (see Chap. 4), we take a closer look on these algorithms in Sect. 8.2.

8.1.2 Solving Systems of Multivariate Polynomial Equations

The problem of solving a system of linear equations first appeared at around 1 AD in ancient China. However, a general method to solve this problem was not found until the work of Gauss and Lagrange at about 1810. For us it is only important that a system of linear equations can be solved efficiently in polynomial time (in time $O(n^3)$ by Gaussian elimination or $O(2^{2.376})$ by the Coppersmith/Winograd method).

On the other hand, solving a system of nonlinear multivariate polynomial equations is a much harder task. In fact, that problem or the MQ Problem (see Sect. 2.3) can be proven to be NP-hard. Algorithms to solve this problem did not appear until 1965, when Buchberger published his thesis. Since these algorithms for solving systems of nonlinear polynomial equations are important for estimating the security of multivariate schemes, we take a closer look on them in Sects. 8.3–8.7.

8.2 Solving Univariate Polynomials of High Degree

In order to invert the central map of HFE and its variants (see Chap. 4), one has to find a root of the univariate polynomial $\mathcal{F}(Y) = X$ over the extension field \mathbb{E} . Since this polynomial can be of very high degree, we can not do this with the methods described in the previous section. In this section, we describe two efficient algorithms to solve univariate polynomials of high degree over finite fields: The Berlekamp and the Cantor–Zassenhaus algorithm.

Since both of these algorithms require as input a square-free polynomial, we first have to deal with the question of how to turn a general polynomial into a square-free one. We find

Theorem 8.6 *A monic polynomial $f \in \mathbb{F}[X]$ is square-free (no repeated factors), if and only if*

$$\gcd(f(X), f'(X)) = 1$$

holds.

Proof “ \Rightarrow ”: Let $f(X) \in \mathbb{F}[X]$ be a monic square-free polynomial of degree d . So, in the splitting field of $f(X)$, we have

$$f(X) = \prod_{i=1}^d (X - \alpha_i)$$

with pairwise distinct linear factors. Therefore,

$$f'(X) = \sum_{i=1}^d \left(\prod_{j \neq i} (X - \alpha_j) \right).$$

Since the linear factors are pairwise distinct, we have $(X - \alpha_i) \nmid \prod_{j \neq i} (X - \alpha_j)$. Moreover, $(X - \alpha_i)$ divides all but one of the summands of $f'(X)$ which implies $\gcd(f(X), f'(X)) = 1$.

“ \Leftarrow ”: If $f(X)$ is not square-free, we can write $f(X) = g(X)^2 h(X)$ with two monic polynomials $g(X)$ and $h(X)$ of lower degree. We get

$$f'(X) = 2g(X)g'(X)h(X) + g(X)^2 h'(X)$$

which implies that $g(X) \mid f'(X)$ and $g(X) \mid \gcd(f(X), f'(X))$. Therefore, $\gcd(f(X), f'(X)) \neq 1$, which proves the claim by contradiction. \square

Moreover, we have

Theorem 8.7 Let $\gcd(f(X), f'(X)) = 0$ for some non constant polynomial $f(X) \in \mathbb{F}_{p^e}[X]$. Then $f(X) = g(X)^p$ for some $g(X) \in \mathbb{F}_{p^e}[X]$.

Proof Let $f(X) = \sum_{i=0}^d \alpha_i X^i$ with $\alpha_d \neq 0$. Thus, $f'(X) = \sum_{i=1}^d i \alpha_i X^{i-1}$. Since $f(X) \neq 0$, $\gcd(f(X), f'(X)) = 0$ implies $f'(X) = 0$ and therefore $i \alpha_i = 0 \forall i \in \{1, \dots, d\}$.

Assume that, for a fixed i , $\alpha_i \neq 0$ holds. We then have $i = 0 \pmod p$, which implies that all non zero terms of $f(X)$ have exponents divisible by p . Therefore, we have $f(X) = g(X)^p$, which, due to the Frobenius mapping property, implies $f(X) = g(X)^p$. \square

We therefore get the following algorithm to make a univariate monic polynomial square-free.

Algorithm 8.1 Square-free algorithm

Input: A monic non-zero polynomial $f \in \mathbb{F}_{p^e}[X]$, $f(X) = \prod_i f_i(X)^{e_i}$

Output: A square-free polynomial $\tilde{f} \in \mathbb{F}_{p^e}[X]$, $\tilde{f}(X) = \prod_i f_i(X)$

```

1:  $g(X) \leftarrow \gcd(f(X), f'(X))$ 
2: if  $g(X) \neq 0$  then //  $f(X) = h(X)^p$  for some polynomial  $h$ 
3:    $h(X) \leftarrow f(X)^{1/p}$ 
4:   run the algorithm again with  $h(X)$ 
5: else if  $g(X) = 1$  then //  $f(X)$  is already square free
6:   return  $f(X)$ 
7: else
8:   run the algorithm again with  $\frac{f(X)}{g(X)}$ .
9: end if
```

8.2.1 Berlekamp's Algorithm

The Berlekamp algorithm was developed in 1967 by Elwyn Berlekamp at Bell laboratories [4].

An important ingredient of the algorithm is the so called Berlekamp matrix.

Definition 8.8 For a polynomial $f(X) \in \mathbb{F}_q[X]$ of degree d , we define the **Berlekamp matrix** Q as

$$Q_f = \begin{pmatrix} q_{0,0} & q_{0,1} & \cdots & q_{0,d-1} \\ q_{1,0} & q_{1,1} & \cdots & q_{1,d-1} \\ \vdots & & \ddots & \vdots \\ q_{d-1,0} & q_{d-1,1} & \cdots & q_{d-1,d-1} \end{pmatrix} \in \mathbb{F}_q^{d \times d}.$$

Here, the element $q_{i,j}$ is the coefficient of X^j of the degree $(d-1)$ polynomial $X^{iq} \bmod f(X)$.

Theorem 8.9 *Berlekamp's algorithm gives the factorization of a monic square-free polynomial $f(X)$ into its irreducible factors.*

Proof See [1, Theorem 7.4.5]. □

Algorithm 8.2 Berlekamp's algorithm

Input: A monic square free polynomial $f(X) \in \mathbb{F}_q[X]$ of degree d

Output: List \mathcal{B} of irreducible factors $\hat{f}(X)$ such that $f = \prod_{\hat{f} \in \mathcal{B}} \hat{f}(X)$

- 1: Generate the Berlekamp matrix Q_f of $f(X)$ (see Definition 8.8).
 - 2: Compute a basis v_1, \dots, v_r of the left-kernel of the matrix $Q_f - I_d$. If $r = 1$, the polynomial $f(X)$ is irreducible and we are done. Otherwise, continue to the next step.
 - 3: **for** $k = 2$ to r **do**
 - 4: **for** $s \in F$ **do**
 - 5: Set $\hat{f} = \gcd(f(X), v_k(X) - s)$. If $\hat{f} \neq 1$, add \hat{f} to \mathcal{B} and set $f(X) = \frac{f(X)}{\hat{f}(X)}$. If $f(X) = 1$, then terminate.
 - 6: **end for**
 - 7: **end for**
 - 8: **return** \mathcal{B} .
-

Theorem 8.10 *Over a finite field F_q , the computational complexity of Berlekamp's algorithm for a degree n polynomial is*

$$O(n^\omega + n(\log(n) \log(\log(n)) \log q)),$$

where n^ω is the cost to multiply two $n \times n$ matrices, and $O(n(\log(n) \log(\log(n))))$ is the complexity to multiply two polynomials of degree n .

Proof See [25, Sect. 2]. □

8.2.2 Toy Example

We want to find the irreducible factors of the polynomial

$$f(X) = X^6 + X^5 + X^4 + X^3 + 1 \in \mathbb{F}_2[X].$$

First, we have to check if $f(X)$ is square-free. We get

$$\frac{f(X)}{f'(X)} = \frac{X^6 + X^5 + X^4 + X^3 + 1}{X^4 + X^2} = 1.$$

We therefore know that $f(X)$ is square-free and can continue with Berlekamp's algorithm. For this, we compute for $i \in \{0, \dots, 5\}$ the polynomials $X^{2^i} \bmod f(X)$, i.e.

$$\begin{aligned} X^{2^0} \bmod f(X) &= 1 \\ X^{2^1} \bmod f(X) &= X^2 \\ X^{2^2} \bmod f(X) &= X^4 \\ X^{2^3} \bmod f(X) &= X^5 + X^4 + X^3 + 1 \\ X^{2^4} \bmod f(X) &= X^4 + X^2 + X \\ X^{2^5} \bmod f(X) &= X^5 + 1 \end{aligned}$$

and put the result into the Berlekamp matrix Q_f . We obtain

$$Q_f = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

By subtracting the identity matrix I_6 from Q_f we get

$$Q_f - I_6 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

A basis of the left kernel of this matrix is given by $\mathbf{b}_1 = (1, 0, 0, 0, 0, 0) \equiv 1$ and $\mathbf{b}_2 = (0, 1, 0, 0, 1, 0) \equiv X + X^4$.

Next, we perform step 3 of the algorithm for the nontrivial basis element $u(X) = X + X^4$.

- For $s = 0$, we get $\gcd(f(X), u(X)) = \gcd(X^6 + X^5 + X^4 + X^3 + 1, X^4 + X) = 1$.
- For $s = 1$, we get $\gcd(f(X), u(X) - 1) = \gcd(X^6 + X^5 + X^4 + X^3 + 1, X^4 + X - 1) = X^4 + X + 1$.

So, $f_1(X) = X^4 + X + 1$ is an irreducible factor of $f(X)$ and we get

$$\frac{f(X)}{f_1(X)} = \frac{X^6 + X^5 + X^4 + X^3 + 1}{X^4 + X + 1} = X^2 + X + 1 =: f_2(X).$$

Since the algorithm terminates, we know that $f_2(X)$ is irreducible and we have

$$f(X) = f_1(X)f_2(X) = (X^4 + X + 1)(X^2 + X + 1)$$

.

8.2.3 The Cantor–Zassenhaus Algorithm

In [7], Cantor and Zassenhaus proposed a polynomial factoring algorithm which works over finite fields of odd characteristic. The algorithm factors a monic square-free polynomial $f(X) \in \mathbb{F}$ following two steps:

1. Factor the polynomial $f(X)$ into a product of polynomials $f(X) = \prod_d R_d(X)$, where each $R_d(X)$ is the product of all degree d irreducible factors of $f(X)$ (distinct degree factoring algorithm, DDF)
2. Factor each of the polynomials $R_d(X)$ into its irreducible components (fixed degree factoring algorithm, FDF).

Before we come to the description of the algorithms, we need two propositions.

Proposition 8.11 *In the polynomial ring $\mathbb{F}_q[X]$ we have*

$$X^q - X = \prod_{s \in \mathbb{F}} (X - s).$$

Proof The elements of \mathbb{F}_q^* form a multiplicative group of order $q - 1$, and therefore, for all elements $s \in \mathbb{F}_q^*$, we have $s^q - s = 0$. Thus, all elements of \mathbb{F}_q^* are roots of the polynomial $X^q - X$. Since $0 \in \mathbb{F}_q$ is clearly another root, the terms on the right hand side of the above equation are exactly the q linear factors of the degree q polynomial $X^q - X$. \square

Proposition 8.12 *The polynomial $X^{q^m} - X$ is the product of all irreducible polynomials over \mathbb{F}_q of degree $d|m$.*

Proof We prove the proposition by showing that each root of $X^{q^m} - X$ is a root of some irreducible polynomial in $\mathbb{F}_q[X]$ of degree $d|m$ and vice versa.

By Proposition 8.11, the roots of $X^{q^m} - X$ are exactly the elements of the field \mathbb{F}_{q^m} .

Consider an element $\alpha \in \mathbb{F}_{q^m}$ and the extension field $\mathbb{F}_q(\alpha)$ of \mathbb{F}_q . Then we get

$$[\mathbb{F}_{q^m} : \mathbb{F}_q] = [\mathbb{F}_{q^m} : \mathbb{F}_q(\alpha)] * [\mathbb{F}_q(\alpha) : \mathbb{F}_q].$$

We therefore get $[\mathbb{F}_q(\alpha) : \mathbb{F}_q] = d$ for some $d|m$ which implies that the degree of the minimal polynomial of α divides m .

On the other hand, let β be a root of some irreducible polynomial in $\mathbb{F}_q[X]$ of degree $d|m$. Then, $\mathbb{F}_q(\beta)$ is a subfield of \mathbb{F}_{q^m} which implies $\beta \in \mathbb{F}_{q^m}$. Therefore, β is a root of $X^{q^m} - X$. \square

With the help of these two propositions, we can now formulate the DDF and FDF algorithms as shown in Algorithms 8.3 and 8.4.

Algorithm 8.3 Distinct degree factoring algorithm (DDF)

Input: A monic squarefree polynomial $f(X) \in \mathbb{F}_q[X]$

Output: Polynomials $R_1(X), \dots, R_s(X) \in \mathbb{F}_q[X]$ such that $\prod_{i=1}^s R_i(X) = f(X)$. Each polynomial R_i is the product of all irreducible factors of $f(X)$ of degree i .

```

1:  $i \leftarrow 1$ 
2: repeat
3:    $R_i(X) \leftarrow \gcd(X^{q^i} - X, f(X))$ 
4:    $f(X) \leftarrow \frac{f(X)}{R_i(X)}$ 
5:    $i \leftarrow i + 1$ 
6: until  $f(X) = 1$ 
7: return  $R_1(X), \dots, R_i(X)$ 

```

Algorithm 8.4 Fixed degree factoring algorithm (FDF)

Input: A monic squarefree polynomial $R_d(X) \in \mathbb{F}_q[X]$ which can be written as product of irreducible polynomials f_1, \dots, f_r of degree d

Output: Set \mathcal{B} of irreducible factors of R_d

```

1:  $\mathcal{B} = \emptyset$ 
2: repeat
3:   Choose a random polynomial  $a(x)$  of degree  $\geq d$  and compute
      $g(X) = \gcd(a(X), R_d(X))$ 
4:   if  $g(X) \neq 1$  then //  $g(X)$  is a non-trivial factor of  $R_d(X)$ 
5:      $\mathcal{B} = \mathcal{B} \cup \{g(X)\}$ 
6:      $R_d(X) = R_d(X)/g(X)$ 
7:   else
8:      $g(X) = \gcd(a^{(q^d-1)/2} - 1, R_d(X))$ 
9:     if  $g(x) \neq 1$  then
10:       $\mathcal{B} = \mathcal{B} \cup \{g(x)\}$ 
11:       $R_d(X) = R_d(X)/g(X)$ 
12:     end if
13:   end if
14: until  $\deg(R_d(X)) = d$ 
15:  $\mathcal{B} = \mathcal{B} \cup \{R_d\}$ 
16: return  $\mathcal{B}$ 

```

By composing the DDF and the FDF algorithms, the Cantor–Zassenhaus algorithm now finds all irreducible factors of a polynomial $f \in \mathbb{F}_q[X]$.

Theorem 8.13 *The Cantor–Zassenhaus algorithm factors a monic squarefree polynomial over a finite field of odd characteristic into its irreducible factors.*

Algorithm 8.5 Cantor–Zassenhaus algorithm**Input:** A monic squarefree polynomial $f(X) \in \mathbb{F}_q[X]$ **Output:** Set \mathcal{B} of irreducible polynomials $\tilde{f}(X)$ such that $f(X) = \prod_{\tilde{f} \in \mathcal{B}} \tilde{f}(X)$.

- 1: $\mathcal{B} = \emptyset$
- 2: Use the DDF algorithm to factor f into its distinct degree components R_1, \dots, R_s ; each $R_d(X)$ is the product of all irreducible factors of $f(X)$ of degree d
- 3: **for** $i = 1$ to s **do**
- 4: $\mathcal{B} = \mathcal{B} \cup \text{FDF}(R_i(X))$
- 5: **end for**
- 6: **return** \mathcal{B}

Proof Due to Proposition 8.12, $X^{q^i} - X$ is the product of all irreducible polynomials in $\mathbb{F}_q[X]$ of degree $d|i$. Since, in step 4 of the DDF algorithm, we divided out lower degree factors, $R_i(X) = \gcd(X^{q^i} - X, f(X))$ is the product of all irreducible factors of $f(X)$ of degree i , which shows the correctness of the DDF algorithm.

So, it remains to show that the FDF algorithm factors a polynomial $R_d(X)$ into irreducible polynomials $g_1(X), \dots, g_r(X)$, where all g_1, \dots, g_r are of degree d . Since $\gcd(g_i, g_j) = 1$ for $i \neq j$, the Chinese Remainder theorem yields a ring isomorphism

$$\chi : \mathbb{F}[X]/\langle f \rangle \rightarrow \mathbb{F}[X]/\langle g_1 \rangle \times \cdots \times \mathbb{F}[X]/\langle g_r \rangle.$$

Let $S \subsetneq \{1, \dots, r\}$. If we can find a polynomial $s(X) \in \mathbb{F}[X]$ such that $s(X) \bmod g_i(X) = 0$ for $i \in S$, but $s(X) \bmod g_i(X) \neq 0$ for $i \notin S$ (a so called splitting polynomial), then $\gcd(R_d(X), s(X))$ is a non-trivial factor of $R_d(X)$.

Let $a \in \mathbb{F}[X]$ be a polynomial of low degree with $\gcd(a(X), R_d(X)) = 1$. We therefore have $a \bmod g_i \neq 0 \forall i = 1, \dots, r$. Since each of the rings $\mathbb{F}[X]/\langle g_i \rangle$ is a finite group of order q^d , we have $a^{q^d-1} = 1 \bmod g_i(x) \forall i$. Since q^d is odd, we have $a^{(q^d-1)/2} \bmod g_i(X) = \pm 1$ with equal probability. In other words, we have $a^{(q^d-1)/2} - 1 \bmod g_i = 0$ with probability exactly $1/2$. Therefore, $a^{(q^d-1)/2} - 1$ is a splitting polynomial for $R_d(x)$ with probability $1 - 2^{1-r}$, which yields an irreducible factor of degree d of $R_d(X)$. By repeating this step, the algorithm therefore outputs all irreducible factors of $R_d(x)$. \square

Theorem 8.14 *The computational complexity of the algorithm is*

$$O(n^2 \log n \log(\log n)(\log q + \log n))$$

over a finite field F_q .

Proof see [25, Sect. 2]. \square

Remark 8.15 The Cantor–Zassenhaus algorithm as described above works only for finite fields $\mathbb{F}_q[X]$ of odd characteristic. However, by using the polynomial $b(X) = \sum_{i=0}^{q^d} X^i$ instead of $a(X)$, it can be directly extended to fields of even characteristic.

Remark 8.16 In the case of the HFE scheme, it is not necessary to factor the polynomial $\mathcal{F}(X) - Y$ completely. Instead of this, it is sufficient to find one root of this polynomial. To do this, we run the DDF algorithm to find the polynomial $R_1(X)$ which is the product of all linear factors of $\mathcal{F}(X) - Y$. We then use the FDF algorithm to find a root of $R_1(X)$. We can abort the algorithm after having found one root.

8.2.4 Toy Example

Consider the polynomial $f(X) = X^7 + 2X^5 + X^3 + 2X \in \mathbb{F}_3[X]$.

First, we have to check if $f(X)$ is square free. Since

$$\gcd(f(X), f'(X)) = \gcd(X^7 + 2X^5 + X^3 + 2X, X^6 + X^4 + 2) = 1$$

this is the case. Next, we apply the DDF algorithm

- $i = 1 \Rightarrow \gcd(X^3 - X, f(X)) = X^3 + 2X =: f_1(X)$ and $g(X) = \frac{f(X)}{f_1(X)} = X^4 + 1$.
- $i = 2 \Rightarrow \gcd(X^9 - X, g(X)) = X^4 + 1 =: f_2(X)$.

Therefore we have $f(X) = f_1(X)f_2(X)$, where $f_1(X)$ is the product of all linear factors of $f(X)$ and $f_2(X)$ is the product of all irreducible quadratic factors of $f(X)$.

Next, we factor $f_1(X)$ into its linear factors. For this, we choose random polynomials $r(X)$ of degree 1. We have $e = (p^1 - 1)/2 = 1$.

- $a(X) = X + 2 \Rightarrow \gcd(X, f_1(X)) = X$. We set $f_1(X) = f_1(X)/X = X^2 + 2$.
- $a(X) = X + 1 \Rightarrow \gcd(X + 2, f_1(X)) = X + 1$. We now have $f_1(X) = f_1(X)/(X + 1) = X + 2$, which is itself an irreducible linear factor. So we are done.

We have $f(X) = X(X + 1)(X + 2)f_2(X)$.

In the last step, we have to factor $f_2(X) = X^4 + 1$ into its irreducible quadratic factors. For this, we choose random polynomials $r(x)$ of degree 2. We have $e = (p^2 - 1)/2 = 4$.

- $a(X) = x^2 \Rightarrow \gcd(X^8 + 1, f_2(X)) = 1$. So, we have to choose another polynomial $r(X)$.
- $a(X) = X^2 + X + 2 \Rightarrow r(X)^4 + 1 = X^8 + X^7 + 2X^6 + X^5 + X^4 + 2X^3 + 2X^2 + 2X + 2$.
 $\Rightarrow \gcd(f_2(X), r(X)^4 + 1) = 2X^2 + 2X + 2$. We set $f_2(X) = f_2(X)/(2X^2 + 2X + 2) = X^2 + X + 2$. Since this is also quadratic, we are done.

Altogether, we have

$$f(X) = X(X + 1)(X + 2)(2X^2 + 2X + 2)(X^2 + X + 2).$$

8.3 The XL-Algorithm

The XL-Algorithm (“eXtended Linearization”) is a general method to find a solution of a system \mathcal{P} of nonlinear multivariate equations over a finite field \mathbb{F} proposed by Courtois et al. in [8]. It is most effective for overdetermined systems having exactly one solution. The algorithm consists of two steps.

1. Extend the system $\mathcal{P} \subset \mathbb{F}[x_1, \dots, x_n]$ by multiplying the polynomials $p \in \mathcal{P}$ by all monomials $h \in \mathbb{F}[x_1, \dots, x_n]$ up to a given degree.
2. Perform Gaussian elimination on the extended system to generate a univariate polynomial $\hat{p}(x_i)$. Solve this polynomial by e.g. Berlekamp’s algorithm to find the value of x_i , substitute it into the polynomials of \mathcal{P} and continue with step 1 on the simplified system.

This procedure is formalized in Algorithm 8.6. If the value of D was chosen too small, the Gaussian elimination performed in step 4 of the algorithm will not produce a univariate polynomial. In this case one has to choose a larger value of D and try again.

Algorithm 8.6 XL-Algorithm

Input: System $\mathcal{P} = (p^{(1)}, \dots, p^{(m)})$ of multivariate quadratic polynomials in the variables x_1, \dots, x_n

Output: vector $\bar{\mathbf{x}} = (\bar{x}_1, \dots, \bar{x}_n)$ such that $p^{(1)}(\bar{\mathbf{x}}) = \dots = p^{(m)}(\bar{\mathbf{x}}) = 0$

- 1: **for** $i = 1$ to n **do**
 - 2: Fix an integer $D > 2$.
 - 3: Let T^{D-2} be the set of all monomials up to degree $D - 2$ and define $\tilde{\mathcal{P}}_D$ to be the set of all polynomials of the form $hp^{(j)}$ with $h \in T^{D-2}$ and $j = 1, \dots, m$.
 - 4: Let $>_\sigma$ be a monomial ordering according to which the univariate polynomials in x_i and the constant terms come last. Sort the monomials of $\tilde{\mathcal{P}}_D$ according to $>_\sigma$ and interpret each monomial as an independent variable. Perform Gaussian elimination on the resulting system. If this produces a univariate polynomial $\hat{p}(x_i)$, go to the next step. Otherwise, choose a larger value for D and try again.
 - 5: Use Berlekamp’s algorithm to find the value \bar{x}_i of x_i and substitute this value into the polynomials of \mathcal{P} .
 - 6: **end for**
 - 7: **return** $\bar{\mathbf{x}} = (\bar{x}_1, \dots, \bar{x}_n)$.
-

Let P_D be the vector space spanned by the polynomials in $\tilde{\mathcal{P}}_D$ (when interpreting each monomial as an independent variable). Then we have

Theorem 8.17 *The XL-Algorithm terminates for a fixed degree D , if $|T^D| - \dim(P_D) \leq D$ holds.*

Proof The XL-Algorithm terminates, if and only if Step 4 of the algorithm generates a univariate polynomial. Let us suppose that, in Step 4 of Algorithm 8.6, we want to find a univariate polynomial in the variable \hat{x} . We denote the space spanned by the $D + 1$ monomials in T^D containing only \hat{x} (these are the

monomials $\hat{x}^0, \dots, \hat{x}^D$ by $P_{\hat{x}}$. Under the assumption of the theorem, we have $\dim(P_D) + \dim(P_{\hat{x}}) > |T^D|$, i.e. $P_D \cap P_{\hat{x}} \neq \emptyset$. Therefore, Step 4 of the algorithm will find at least one univariate polynomial. \square

Unfortunately, Theorem 8.17 does not yield the smallest degree D for which the XL-Algorithm works. Analyzing this problem would go beyond the scope of this book. For the interested reader we refer to [9].

8.3.1 Toy Example

Let $\mathbb{F} = \text{GF}(7)$ and $(m, n) = (4, 3)$. We want to find a solution of the (overdetermined) system $\mathcal{P} = (p^{(1)}, p^{(2)}, p^{(3)}, p^{(4)})$ with

$$\begin{aligned} p^{(1)}(x_1, x_2, x_3) &= 5x_1^2 + 5x_1x_2 + 2x_1x_3 + 6x_1 + 4x_2 + 6x_3^2 + 2x_3 + 2, \\ p^{(2)}(x_1, x_2, x_3) &= 6x_1^2 + 5x_1x_2 + 3x_1x_3 + 3x_1 + 5x_2^2 + x_2x_3 + 3x_2 + 3x_3^2 + 5x_3, \\ p^{(3)}(x_1, x_2, x_3) &= 2x_1^2 + 5x_1x_2 + 5x_1x_3 + 2x_2^2 + 3x_2x_3 + 3x_2, \\ p^{(4)}(x_1, x_2, x_3) &= 4x_1^2 + 5x_1x_2 + x_1 + x_2^2 + x_2x_3 + 3x_2 + 4x_3^2 + 6x_3 + 6. \end{aligned}$$

We multiply the polynomials of the system \mathcal{P} by all monomials of degree ≤ 1 and obtain the coefficient matrix

$$M_1 = \begin{pmatrix} 0 & 0 & 0 & 5 & 0 & 0 & 5 & 0 & 2 & 6 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 6 & 2 & 2 \\ 0 & 0 & 0 & 6 & 0 & 0 & 5 & 0 & 3 & 3 & 0 & 0 & 5 & 0 & 1 & 3 & 0 & 3 & 5 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 5 & 0 & 5 & 0 & 0 & 0 & 2 & 0 & 3 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 5 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 3 & 0 & 4 & 6 & 6 \\ 5 & 5 & 2 & 6 & 0 & 0 & 4 & 6 & 2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & 5 & 3 & 3 & 5 & 1 & 3 & 3 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 5 & 5 & 0 & 2 & 3 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 5 & 0 & 1 & 1 & 1 & 3 & 4 & 6 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 5 & 2 & 6 & 0 & 0 & 0 & 0 & 0 & 4 & 6 & 2 & 2 & 0 & 0 & 0 & 0 \\ 0 & 6 & 0 & 0 & 5 & 3 & 3 & 0 & 0 & 0 & 5 & 1 & 3 & 3 & 5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 5 & 5 & 0 & 0 & 0 & 0 & 2 & 3 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 5 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 3 & 4 & 6 & 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 & 5 & 0 & 2 & 6 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 6 & 2 & 2 & 0 \\ 0 & 0 & 6 & 0 & 0 & 5 & 0 & 3 & 3 & 0 & 0 & 5 & 0 & 1 & 3 & 0 & 3 & 5 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 5 & 0 & 5 & 0 & 0 & 0 & 2 & 0 & 3 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 5 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 3 & 0 & 4 & 6 & 6 & 0 \end{pmatrix}.$$

Within this matrix, the polynomials of the extended system (rows) are sorted in the order

$$(p^{(1)}, \dots, p^{(4)}, x_1 p^{(1)}, \dots, x_1 p^{(4)}, x_2 p^{(1)}, \dots, x_2 p^{(4)}, x_3 p^{(1)}, \dots, x_3 p^{(4)}).$$

The monomials of degree ≤ 3 are sorted according to the monomial ordering $>_\sigma$ with $x_1^3 >_\sigma x_1^2 x_2 >_\sigma x_1^2 x_3 >_\sigma x_1^2 >_\sigma x_1 x_2^2 >_\sigma x_1 x_2 x_3 >_\sigma x_1 x_2 >_\sigma x_1 x_3^2 >_\sigma x_1 x_3 >_\sigma x_1 >_\sigma x_2^3 >_\sigma x_2^2 x_3 >_\sigma x_2^2 >_\sigma x_2 x_3^2 >_\sigma x_2 x_3 >_\sigma x_2 >_\sigma x_3^3 >_\sigma x_3^2 >_\sigma x_3 >_\sigma 1$.

Putting the matrix M_1 into echelon form yields

$$\tilde{M}_1 = \begin{pmatrix} 5 & 5 & 2 & 6 & 0 & 0 & 4 & 6 & 2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 6 & 2 & 0 & 5 & 1 & 1 & 0 & 4 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 6 & 6 & 3 & 6 & 3 & 6 & 0 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 5 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 3 & 0 & 4 & 6 & 6 \\ 0 & 0 & 0 & 0 & 3 & 3 & 2 & 3 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 1 & 1 & 3 & 5 & 5 & 1 & 6 & 3 & 1 & 2 & 0 & 5 & 4 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 6 & 0 & 5 & 3 & 0 & 0 & 5 & 0 & 6 & 5 & 0 & 5 & 4 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 4 & 6 & 3 & 0 & 1 & 0 & 5 & 4 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 5 & 0 & 2 & 0 & 0 & 2 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 4 & 6 & 5 & 0 & 0 & 4 & 6 & 6 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 3 & 2 & 3 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 2 & 3 & 4 & 6 & 4 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 5 & 1 & 0 & 5 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 4 & 2 & 4 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 4 & 5 & 4 & 6 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 6 & 3 & 3 \end{pmatrix}.$$

From the last row of the matrix \tilde{M}_1 we can derive the univariate polynomial

$$5x_3^3 + 6x_3 + 3 = 0.$$

Solving this polynomial using Berlekamp's algorithm yields $x_3 = 5$.

Note here that, as the toy example shows, the inversion of Theorem 8.17 is not true. Here we have $|T^3| = 20$ (number of columns of \tilde{M}_1) and $\dim(P_3) = 16$ (number of non-zero rows in \tilde{M}_1). It follows $|T^D| - \dim(P_D) = 4 > 3$. Despite of this, the XL-Algorithm terminates.

We substitute the solution $x_3 = 5$ into the polynomials of the system \mathcal{P} , obtaining

$$\begin{aligned} \hat{p}^{(1)}(x_1, x_2) &= 5x_1^2 + 5x_1x_2 + 2x_1 + 4x_2 + 1, \\ \hat{p}^{(2)}(x_1, x_2) &= 6x_1^2 + 5x_1x_2 + 4x_1 + 5x_2^2 + x_2 + 2, \\ \hat{p}^{(3)}(x_1, x_2) &= 2x_1^2 + 5x_1x_2 + 4x_1 + 2x_2^2 + 4x_2, \\ \hat{p}^{(4)}(x_1, x_2) &= 4x_1^2 + 5x_1x_2 + x_1 + x_2^2 + x_2 + 3. \end{aligned}$$

The system $\hat{\mathcal{P}}$ corresponds to the matrix

$$M_2 = \begin{pmatrix} 5 & 5 & 2 & 0 & 4 & 1 \\ 6 & 5 & 4 & 5 & 1 & 2 \\ 2 & 5 & 4 & 2 & 4 & 0 \\ 4 & 5 & 1 & 1 & 1 & 3 \end{pmatrix}.$$

Here we used the monomial ordering $>_\sigma: x_1^2 >_\sigma x_1x_2 >_\sigma x_1 >_\sigma x_2^2 >_\sigma x_2 >_\sigma 1$.

We do not have to extend the system $\hat{\mathcal{P}}$ anymore, but can directly compute the row echelon form of the matrix M_2 . We obtain

$$\tilde{M}_2 = \begin{pmatrix} 5 & 5 & 2 & 0 & 4 & 1 \\ 0 & 6 & 3 & 5 & 6 & 5 \\ 0 & 0 & 1 & 3 & 5 & 2 \\ 0 & 0 & 0 & 3 & 3 & 1 \end{pmatrix}.$$

From the last row of the matrix \tilde{M}_2 we can derive the univariate polynomial

$$3x_2^2 + 3x_2 + 1 = 0,$$

leading to $x_2 = 5$. Substituting this value into the polynomials of $\hat{\mathcal{P}}$ yields

$$\hat{p}^{(1)}(x_1) = 5x_1^2 + 6x_1,$$

$$\hat{p}^{(2)}(x_1) = 6x_1^2 + x_1 + 6,$$

$$\hat{p}^{(3)}(x_1) = 2x_1^2 + x_1,$$

$$\hat{p}^{(4)}(x_1) = 4x_1^2 + 5x_1 + 5,$$

yielding $x_1 = 3$. Therefore, we have found the solution $(x_1, x_2, x_3) = (3, 5, 5)$ of the original system $\mathcal{P}(x_1, x_2, x_3) = 0$. \square

After the proposal of the basic algorithm, several variants of XL were suggested. The most important of these are

- **FXL**: The idea of this variation is to fix the values of some of the variables before applying the XL-Algorithm. This often helps to find a solution of the system \mathcal{P} at a lower degree D . However, in case of a wrong guess, one has to run the XL-Algorithm several times.
- **XL2**: This XL-variant was proposed for the use over the field $\text{GF}(2)$ of 2 elements. The main idea is to make use of the field equations $x_i^2 - x_i = 0$ during the elimination process.
- **MutantXL**: During the elimination process of XL, there sometimes appear polynomials of lower degree than expected (so called Mutants). The basic idea of MutantXL is to privilege these mutants during the next extension process.

8.4 Gröbner Bases

Gröbner Basis was proposed by Bruno Buchberger in [6]. It allow us to find all solutions of a system of multivariate nonlinear polynomial equations by giving a simple representation of the variety of the solution space. Especially, if the Gröbner Basis was computed according to the lexicographic order of monomials, it is very easy to derive all the solutions of the system from it. In this sense, Gröbner Basis techniques can be viewed as an extension of the Gaussian elimination to the nonlinear case.

Definition 8.18 An **ideal** in $\mathbb{F}[x_1, \dots, x_n]$ is a subset $I \subset \mathbb{F}[x_1, \dots, x_n]$ such that for each element $a \in \mathbb{F}[x_1, \dots, x_n]$ and any element $b \in I$ we have $ab \in I$.

A subset $M \subset I$ is called **generating system** of the ideal I (we write $I = \langle M \rangle$)

$\Leftrightarrow \forall a \in I \exists m_1, \dots, m_s \in M$ and $\alpha_1, \dots, \alpha_s \in \mathbb{F}[x_1, \dots, x_n]$ such that

$$a = \sum_{i=1}^s \alpha_i m_i. \quad (8.2)$$

Theorem 8.19 For every ideal $I \subset \mathbb{F}[x_1, \dots, x_n]$ there exists a finite generating system.

Proof See [22, Theorem 1.8]. □

Let σ be an admissible order of monomials (see Definition 2.8).

Definition 8.20 Let $f = \sum_{i=1}^s c_i t_i$ be a polynomial in $\mathbb{F}[x_1, \dots, x_n]$. Without loss of generality we assume that we have $t_1 >_{\sigma} t_2 >_{\sigma} \dots >_{\sigma} t_s$. Then we call

- t_1 the **leading monomial** of f with respect to the monomial order σ . We denote it by $\text{LM}_{\sigma}(f)$ or $\text{LM}(f)$.
- c_1 the **leading coefficient** of f with respect to the monomial order σ . We denote it by $\text{LC}_{\sigma}(f)$ or $\text{LC}(f)$.
- $c_1 t_1$ the **leading term** of f with respect to σ . We denote it by $\text{LT}_{\sigma}(f)$ or $\text{LT}(f)$.

Note that we have

$$\text{LT}(f) = \text{LC}(f)\text{LM}(f).$$

Definition 8.21 A **Gröbner Basis** of an ideal I is a subset $G = \{g^{(1)}, \dots, g^{(s)}\} \subset I$ such that

$$I = \langle g^{(1)}, \dots, g^{(s)} \rangle \text{ and } \langle \text{LT}(I) \rangle = \langle \text{LT}(g^{(1)}), \dots, \text{LT}(g^{(s)}) \rangle. \quad (8.3)$$

Remark 8.22 Definition 8.21 shows that the result of a Gröbner Basis algorithm depends on the chosen monomial order. For example, a Gröbner Basis with respect to the lexicographical order is in general not a Gröbner Basis with respect to the graded reverse lexicographical order.

8.4.1 Reduction of Polynomials

Let $g \in \mathbb{F}[x_1, \dots, x_n]$ and $F = \{f^{(1)}, \dots, f^{(s)}\} \subset \mathbb{F}[x_1, \dots, x_n]$. The polynomial g can be reduced modulo F to a polynomial $h \in \mathbb{F}[x_1, \dots, x_n]$ (we write $g \mapsto_F h$), if $\exists p^{(i)} \in \mathbb{F}[x_1, \dots, x_n]$ ($i = 1, \dots, s$) such that

$$h = g - \sum_{i=1}^s p^{(i)} f^{(i)}. \quad (8.4)$$

Definition 8.23 The polynomial $g \in \mathbb{F}[x_1, \dots, x_n]$ is called **completely reduced** with respect to $F = \{f^{(1)}, \dots, f^{(s)}\}$, if no term of g is divisible by any $\text{LT}(f^{(i)})$ for all $f^{(i)} \in F$.

Theorem 8.24 Let $F = \{f^{(1)}, \dots, f^{(s)}\}$ be an ordered set of polynomials in $\mathbb{F}[x_1, \dots, x_n]$. Then, for any $g \in \mathbb{F}[x_1, \dots, x_n]$ there exist polynomials $p^{(1)}, \dots, p^{(s)}$ such that

$$g = \sum_{i=1}^s p^{(i)} f^{(i)} + r,$$

with a polynomial $r \in \mathbb{F}[x_1, \dots, x_n]$ being completely reduced with respect to F .

Proof See [22, Theorem 4.6]. □

Definition 8.25 The polynomial r of Theorem 8.24 is denoted as the **normal form** of g with respect to F .

For $F = \{f^{(1)}, \dots, f^{(s)}\}$ being a random subset of $\mathbb{F}[x_1, \dots, x_n]$, the normal form r of a polynomial $g \in \mathbb{F}[x_1, \dots, x_n]$ is not uniquely determined and depends on the order of the polynomials $f^{(i)} \in F$. However, we get

Theorem 8.26 If $G = \{g^{(1)}, \dots, g^{(s)}\}$ is a Gröbner Basis, the normal form r of a polynomial $h \in \mathbb{F}[x_1, \dots, x_n]$ with respect to G is uniquely determined and independent of the order of the polynomials $g^{(i)} \in G$.

Proof See [22, Theorem 4.13]. □

8.5 Buchberger's Algorithm and F_4

In this section we introduce the most important algorithms to compute Gröbner Bases. We start with a description of Buchberger's algorithm, which was the first general method to compute Gröbner Bases. After that, we discuss several improvements of this algorithm which finally led to Faugère's F_4 and F_5 algorithms which are currently considered as the fastest general methods to solve systems of nonlinear multivariate polynomial equations.

8.5.1 Buchberger's Algorithm

In his thesis [6], Buchberger developed an efficient algorithm to compute a Gröbner Basis of the ideal generated by a set $F = \{f^{(1)}, \dots, f^{(s)}\} \subset \mathbb{F}[x_1, \dots, x_n]$. The most important notion in his algorithm is the so called S-polynomial.

Definition 8.27 Let $f, g \in \mathbb{F}[x_1, \dots, x_n]$. The **S-polynomial** of f and g is defined by

$$\text{Spoly}(f, g) = \frac{\text{LCM}(\text{LT}(f), \text{LT}(g))}{\text{LT}(g)}g - \frac{\text{LCM}(\text{LT}(f), \text{LT}(g))}{\text{LT}(f)}f. \quad (8.5)$$

The following example shows that the notion of the S-polynomial depends on the monomial order: Let $\mathbb{F} = GF(7)$ and $f, g \in \mathbb{F}[x_1, x_2, x_3]$ with

$$\begin{aligned} f(x_1, x_2, x_3) &= 3x_1 + 5x_2^2 + 6x_2x_3, \\ g(x_1, x_2, x_3) &= 6x_1x_3 + 2x_2^2. \end{aligned}$$

- in the lexicographical order (lex) we have $\text{LT}(f) = 3x_1$, $\text{LT}(g) = 6x_1x_3$ and therefore

$$\begin{aligned} \text{Spoly}_{\text{lex}}(f, g) &= \frac{x_1x_3}{6x_1x_3}g - \frac{x_1x_3}{3x_1}f \\ &= 3x_2^2x_3 + 5x_2^2 + 5x_2x_3^2. \end{aligned}$$

- in the graded lexicographical order (glex) we have $\text{LT}(f) = 5x_2^2$ and $\text{LT}(g) = 6x_1x_3$ and therefore

$$\begin{aligned} \text{Spoly}_{\text{glex}}(f, g) &= \frac{x_1x_2^2x_3}{6x_1x_3}g - \frac{x_1x_2^2x_3}{5x_2^2}f \\ &= 3x_1x_2x_3^2 + 5x_2^4 + 5x_1^2x_3. \end{aligned}$$

- in the graded reverse lexicographical order (grevlex) we have $\text{LT}(f) = 5x_2^2$ and $\text{LT}(g) = 2x_2^2$ and therefore

$$\begin{aligned}\text{Spoly}_{\text{grevlex}}(f, g) &= \frac{x_2^2}{2x_2^2}g - \frac{x_2^2}{5x_2^2}f \\ &= 3x_1x_3 + 3x_2x_3 + 5x_1.\end{aligned}$$

□

Buchberger discovered the following criterion on a set G for being a Gröbner Basis:

Theorem 8.28 *Let $I \subset \mathbb{F}[x_1, \dots, x_n]$ be an ideal in the polynomial ring $\mathbb{F}[x_1, \dots, x_n]$. A subset $G \subset I$ with $I = \langle G \rangle$ is a Gröbner Basis of I if and only if*

$$\text{NormalForm}(\text{Spoly}(p, q)) = 0 \quad \forall p, q \in G.$$

Proof See [22, Theorem 4.18].

□

We can use this criterion to construct an algorithm for finding a Gröbner Basis of an ideal $I = \langle F \rangle \subset \mathbb{F}[x_1, \dots, x_n]$ (see Algorithm 8.7).

Algorithm 8.7 Buchberger's algorithm

Input: $F = \{f^{(1)}, \dots, f^{(s)}\}$, monomial order σ

Output: Gröbner Basis $G = \{g^{(1)}, \dots, g^{(s)}\}$ of $I = \langle f^{(1)}, \dots, f^{(m)} \rangle$

```

1:  $G \leftarrow F$ 
2: repeat
3:    $G' \leftarrow G$ 
4:   for each pair  $\{p, q\}, p \neq q \in G'$  do
5:      $S \leftarrow \text{NormalForm}(\text{Spoly}(p, q), G)$ 
6:     if  $S \neq 0$  then
7:        $G \leftarrow G \cup \{S\}$ 
8:     end if
9:   end for
10: until  $G = G'$ 
11: return  $G$ 
```

Theorem 8.29 *Buchberger's algorithm outputs a Gröbner Basis of the ideal I with respect to the monomial order σ after finitely many steps.*

Proof See [22, Theorem 4.19].

□

8.5.2 Improvements of Buchberger's Algorithm

Buchberger's algorithm as presented above has two main disadvantages:

- (1) The output of the algorithm depends on the order of the polynomials $f^{(i)} \in F$ and
- (2) many reductions of S-polynomials lead to zero and cause unnecessary work.

The first problem can be easily solved by introducing the term *reduced Gröbner Basis*.

Definition 8.30 A Gröbner Basis $G = \{g^{(1)}, \dots, g^{(s)}\}$ is said to be **reduced**, if all polynomials $g^{(i)}$ are monic and $\text{LM}(g^{(i)})$ does not divide $\text{LM}(g^{(j)})$ for all $i \neq j$, $1 \leq i < j \leq s$.

Theorem 8.31 If G and H are reduced Gröbner Bases generating the same ideal, then $G = H$.

Proof See [22, Theorem 4.21]. □

Thus, if we reduce the set G each time we enlarge it (i.e. after line 7 of Algorithm 8.7), the output of the algorithm will be unique.

In his thesis [6], Buchberger discovered two criteria to avoid reductions to zero.

- If $\text{LT}(g^{(i)})$ and $\text{LT}(g^{(j)})$ are relatively prime, then $\text{Spoly}(g^{(i)}, g^{(j)})$ reduces to zero and can be ignored.
- If there exists $g^{(k)} \in G$ such that $\text{LT}(g^{(k)})$ divides $\text{LCM}(\text{LT}(g^{(i)}), \text{LT}(g^{(j)}))$, and if $\text{Spoly}(g^{(i)}, g^{(k)})$ and $\text{Spoly}(g^{(j)}, g^{(k)})$ have already been considered, $\text{Spoly}(g^{(i)}, g^{(j)})$ reduces to zero and can be ignored.

8.5.3 Faugère's F_4 -Algorithm

As a further improvement, Faugère suggested for his F_4 [16] algorithm to compute many of the normal forms in one go by using the Macaulay matrix of the system and fast linear algebra. Let B be the set of all pairs (p, q) with $p \neq q \in G$.

Definition 8.32 Let $(f^{(i)}, f^{(j)})$ be a pair of polynomials from the set B . The weight D_{ij} of the pair $(f^{(i)}, f^{(j)})$ is defined as

$$D_{ij} = \deg(\text{LCM}(\text{LM}(f^{(i)}), \text{LM}(f^{(j)}))).$$

We set

$$d = \min_{(f^{(i)}, f^{(j)}) \in B} D_{ij}$$

and define the selection function $\text{Select}(B)$ by

$$\text{Select}(B) = \{(f^{(i)}, f^{(j)}) \in B : D_{ij} = d\}. \quad (8.6)$$

Faugère's idea was now to reduce all pairs $(f^{(i)}, f^{(j)}) \in \text{Select}(B)$ at the same time. This can be done as follows.

Let $\mathcal{F} = (f^{(1)}, \dots, f^{(s)})$ be a set of multivariate polynomials. Let $S = (t_1, \dots, t_r)$ be the support of the system \mathcal{F} (see Definition 2.2) and M_F be its Macaulay matrix (see Definition 2.10). As in Chap. 2 we can then write

$$\mathcal{F}(x_1, \dots, x_n) = M_F(t_1, \dots, t_r)^T.$$

Let \tilde{M}_F be the unique reduced row echelon form of the matrix M_F . By

$$\tilde{\mathcal{F}}(x_1, \dots, x_n) := \tilde{M}_F(t_1, \dots, t_r)^T$$

we therefore get a somewhat simplified system of polynomials. However, the system $\tilde{\mathcal{F}}$ may contain leading monomials which did not appear in the leading monomials of \mathcal{F} . We define

$$\tilde{\mathcal{F}}^+ = \{f \in \tilde{\mathcal{F}} : \text{LM}(f) \notin \text{LM}(\mathcal{F})\}.$$

Definition 8.33 Let $\mathcal{F} = \{f^{(1)}, \dots, f^{(s)}\} \subset \mathbb{F}[x_1, \dots, x_n]$ and let $p = (f^{(i)}, f^{(j)})$ be a pair of polynomials with $i \neq j$. Let t_i and t_j be the two monomials such that

$$t_i \text{LM}(f^{(i)}) = t_j \text{LM}(f^{(j)}) = \text{LCM}(\text{LM}(f^{(i)}), \text{LM}(f^{(j)})).$$

Define $\text{Left}(p) = (t_i, f^{(i)})$ and $\text{Right}(p) = (t_j, f^{(j)})$; therefore, $\text{Left}(p)$ and $\text{Right}(p)$ are both pairs consisting of a monomial and a polynomial.

If $C \subset B$ is a set of pairs, then we extend the definition as follows:

$$\text{Left}(C) = \cup_{p \in C} \text{Left}(p) \quad \text{and} \quad \text{Right}(C) = \cup_{p \in C} \text{Right}(p).$$

We now describe Faugère's F_4 algorithm. The algorithm takes as input a set $\mathcal{F} \subset \mathbb{F}[x_1, \dots, x_n]$ of multivariate polynomials and a selection function $\text{Select}(B)$. The selection function $\text{Select}(B)$ returns a subset of pairs from the list B of pairs. Besides choosing $\text{Select}(B)$ as indicated by Eq. (8.6), it is also possible that $\text{Select}(B)$ returns just one pair as in Buchberger's algorithm. The F_4 algorithm makes use of Algorithm 8.9 as a subroutine, which reduces a subset of $\mathbb{F}[x_1, \dots, x_n]$ by the candidate basis G . This algorithm takes as input a set $L \subset T^n \times \mathbb{F}[x_1, \dots, x_n]$ and G , and returns a reduced set of polynomials whose leading monomials have not yet appeared as leading monomials in G . In the procedure, tf is simply the multiplication of a monomial by a polynomial.

Faugère calls the part before the Gaussian elimination "Symbolic Preprocessing," as it can be done quickly in a strictly symbolic manner, and also in a time that

Algorithm 8.8 F_4 -algorithm**Input:** set of polynomials $\mathcal{F} = \{f^{(1)}, \dots, f^{(s)}\} \subset \mathbb{F}[x_1, \dots, x_n]$, selection function**Output:** reduced Gröbner basis G of $I = \langle \mathcal{F} \rangle$

```

1:  $G = \tilde{\mathcal{F}}$  // Echelon form of  $\mathcal{F}$ 
2:  $B = \{(f^{(i)}, f^{(j)}) \mid f^{(i)} \neq f^{(j)} \in G\}$ 
3:  $d = 0$ 
4: while  $B \neq \emptyset$  do
5:    $d = d + 1$ 
6:    $C_d = \text{Select}(B)$ 
7:    $B = B \setminus C_d$ 
8:    $L_d = \text{Left}(C_d) \cup \text{Right}(C_d)$ 
9:    $\tilde{\mathcal{F}}^+ = \text{Reduction}(L_d, G)$ 
10:  for  $h \in \tilde{\mathcal{F}}^+$  do
11:     $B = B \cup \{(h, g) \mid g \in G\}$ 
12:     $G = G \cup \{h\}$ 
13:  end for
14: end while
15: return  $G$ 

```

Algorithm 8.9 Reduction algorithm of F_4 **Input:** set L of (monomial / polynomial) pairs, candidate basis G **Output:** set $\tilde{\mathcal{F}}^+$ of polynomials whose leading monomials did not appear in G

```

1:  $\mathcal{F} = \{tf \mid (t, f) \in L\}$ 
2:  $D = LM(\mathcal{F})$ 
3: while  $D \neq \text{Supp}(\mathcal{F})$  do
4:   Choose  $m \in \text{Supp}(\mathcal{F}) \setminus D$ 
5:    $D = D \cup \{m\}$ 
6:   if  $\exists g \in G$  such that  $m = m'LM(g)$  then
7:      $\mathcal{F} = \mathcal{F} \cup \{m'g\}$ 
8:   end if
9: end while
10: use Gaussian elimination on  $M_F$  to compute  $\tilde{\mathcal{F}}$ 
11:  $\tilde{\mathcal{F}}^+ = \{f \in \tilde{\mathcal{F}} \mid LM(f) \notin LM(\mathcal{F})\}$ 
12: return  $\tilde{\mathcal{F}}^+$ 

```

is linear in the size of the input. The Gaussian elimination is the most time consuming step, since $\text{Supp}(\mathcal{F}_d)$ can grow exponentially in n as the degree of the terms increases. Unfortunately for the Gröbner Bases algorithms, intermediate polynomials can have a very high degree, even if the degrees of the final basis elements are moderate.

8.5.4 Toy Example

We perform the first round of the F_4 algorithm for the system $F = (f^{(1)}, f^{(2)}, f^{(3)}) : \mathbb{F}_5^3 \rightarrow \mathbb{F}_5^3$ with

$$f^{(1)}(x_1, x_2, x_3) = 3x_1^2 + 4x_3^2,$$

$$f^{(2)}(x_1, x_2, x_3) = 3x_1x_2 + x_1x_3,$$

$$f^{(3)}(x_1, x_2, x_3) = 4x_2^2 + 2x_2x_3 + 2.$$

The monomials in $f^{(1)}$, $f^{(2)}$ and $f^{(3)}$ are pairwise distinct. We therefore have $G = \tilde{\mathcal{F}} = \mathcal{F}$. We therefore define $g^{(i)} = f^{(i)}$ ($i = 1, \dots, 3$). We have $B = \{(f^{(1)}, f^{(2)}), (f^{(1)}, f^{(3)}), (f^{(2)}, f^{(3)})\}$ and

$$D_{12} = \deg(\text{LCM}(x_1^2, x_1x_2)) = \deg(x_1^2x_2) = 3,$$

$$D_{13} = \deg(\text{LCM}(x_1^2, x_2^2)) = \deg(x_1^2x_2^2) = 4,$$

$$D_{23} = \deg(\text{LCM}(x_1x_2, x_2^2)) = \deg(x_1x_2^2) = 3.$$

We therefore set

$$C_0 = \text{Select}(B) = \{(f^{(1)}, f^{(2)}), (f^{(2)}, f^{(3)})\}.$$

When we look at the first pair $(f^{(1)}, f^{(2)})$, we find

$$x_2\text{LM}(f^{(1)}) = x_1(\text{LM}(f^{(2)})) = \text{LCM}(\text{LM}(f^{(1)}), \text{LM}(f^{(2)})).$$

Therefore, for this pair we have

$$\text{Left}(f^{(1)}, f^{(2)}) = (x_2, f^{(1)}) \text{ and } \text{Right}(f^{(1)}, f^{(2)}) = (x_1, f^{(2)}).$$

Analogously, we find

$$\text{Left}(f^{(2)}, f^{(3)}) = (x_2, f^{(2)}) \text{ and } \text{Right}(f^{(2)}, f^{(3)}) = (x_1, f^{(3)}).$$

We therefore enter **Reduction** with the set

$$L_0 = \{(x_2, f^{(1)}), (x_1, f^{(2)}), (x_2, f^{(2)}), (x_1, f^{(3)})\}.$$

We set

$$\mathcal{F} = \{3x_1^2x_2 + 4x_2x_3^2, 3x_1^2x_2 + x_1^2x_3, 3x_1x_2^2 + x_1x_2x_3, 4x_1x_2^2 + 2x_1x_2x_3 + 2x_1\}.$$

The leading monomials of \mathcal{F} are $D = \{x_1^2x_2, x_1x_2^2\}$ and its support is given by

$$\text{Supp}(\mathcal{F}) = \{x_1^2x_2, x_1x_2^2, x_1^2x_3, x_2x_3^2, x_1x_2x_3, x_1\}.$$

The only monomial $m \in \text{Supp}(\mathcal{F}) \setminus D$ which can be extended to a polynomial from G is x_1 . We have

$$x_1x_1 = x_1^2 = LM(g^{(1)}).$$

We therefore have to add $f^{(5)} = x_1g^{(1)} = x_1^3 + 4x_1x_3^2$ to \mathcal{F} . By doing so, we get a new leading monomial x_1^3 and two new elements for the support of \mathcal{F} .

The monomial x_1 can also be extended to $LM(g^{(2)})$ (by multiplying it with x_2). However, the polynomial $x_2g^{(2)}$ is already contained in \mathcal{F} . Therefore, the Macaulay matrix of \mathcal{F} has the form

$$M_F = \begin{array}{c|ccccccc} & x_1^3 & x_1^2x_2 & x_1x_2^2 & x_1^2x_3 & x_1x_2x_3 & x_1x_3^2 & x_2x_3^2 & x_1 \\ \hline f^{(1)} & 0 & 3 & 0 & 0 & 0 & 0 & 4 & 0 \\ f^{(2)} & 0 & 3 & 0 & 1 & 0 & 0 & 0 & 0 \\ f^{(3)} & 0 & 0 & 3 & 0 & 1 & 0 & 0 & 0 \\ f^{(4)} & 0 & 0 & 4 & 0 & 2 & 0 & 0 & 2 \\ f^{(5)} & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}$$

By putting this matrix into echelon form we get

$$\tilde{M}_F = \begin{array}{c|ccccccc} & x_1^3 & x_1^2x_2 & x_1x_2^2 & x_1^2x_3 & x_1x_2x_3 & x_1x_3^2 & x_2x_3^2 & x_1 \\ \hline f^{(1)} & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ f^{(2)} & 0 & 1 & 0 & 0 & 0 & 0 & 3 & 0 \\ f^{(3)} & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 4 \\ f^{(4)} & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ f^{(5)} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 3 \end{array}$$

We therefore get two new leading monomials $x_1^2x_3$ and $x_1x_2x_3$, i.e.

$$\tilde{\mathcal{F}}^+ = \{x_1^2x_3 + x_2x_3^2, x_1x_2x_3 + 3x_1\}.$$

When denoting these monomials by $\tilde{f}^{(1)}$ and $\tilde{f}^{(2)}$, our new B has the form

$$B = \{(f^{(1)}, f^{(3)}), (\tilde{f}^{(1)}, g^{(1)}), (\tilde{f}^{(1)}, g^{(2)}), (\tilde{f}^{(1)}, g^{(3)}), (\tilde{f}^{(2)}, g^{(1)}), (\tilde{f}^{(2)}, g^{(2)}), (\tilde{f}^{(2)}, g^{(3)})\}.$$

The new candidate basis looks like

$$\begin{aligned} G &= (g^{(1)}, g^{(2)}, g^{(3)}, \tilde{f}^{(1)}, \tilde{f}^{(2)}) \\ &= \{3x_1^2 + 4x_3^2, 3x_1x_2 + x_1x_3, 4x_2^2 + 2x - 2x_3 + 2, x_1^2x_3 + x_2x_3^2, x_1x_2x_3 + 3x_1\}. \end{aligned}$$

As can be seen, the set B of pairs as well as the degree of the polynomials increases during the algorithm. Therefore, even for such a small example, performing the complete F_4 algorithm by hand is (nearly) impossible. \square

The F_5 algorithm is another algorithm proposed by Faugère [17]. The fundamental idea of F_5 is simple, namely one tries to avoid redundancy of computation in the linear algebra steps of F_4 , where there may be many polynomials being reduced

to zero. F_5 claims that it has a very practical and efficient way to do so. In the subsequent years, Faugère has claimed many surprisingly good results in practical attack using his implementation of F_5 . However, by now, no one else could ever reproduce such good results or anything even near in practical implementations. Since Faugère kept his implementation private, from the point of view of scientific research, we think it is better not to take it too seriously until people can publicly verify the results claimed.

The complexity of Gröbner basis algorithms, such as F_4 and F_5 , is mainly determined by the size of the largest matrix appearing during the reduction process. The number of columns in such a matrix is equal to the size of the support of \mathcal{F} . Before the first degree fall occurs, $\text{Supp}(\mathcal{F})$ contains more or less all monomials $t \in T^d$, where d is the degree of the polynomials in \mathcal{F} . After the first degree fall, $\text{Supp}(\mathcal{F})$ might be significantly smaller than T^d , which leads to a smaller matrix and therefore reduces the complexity of the reduction step. Therefore, the complexity of the whole algorithm is mainly determined by the so called *degree of regularity* d_{reg} .

Definition 8.34 The **degree of regularity** is the largest degree of the polynomials computed by a Gröbner basis algorithm before the first degree fall occurs.

The complexity of the F_4 algorithm can be estimated as

$$\text{Complexity}_{F_4} = 3 \binom{n + d_{\text{reg}}}{d_{\text{reg}}} \binom{n}{2}.$$

If the system \mathcal{F} to be solved is underdetermined (the number n of variables is larger than the number m of equations), we can estimate the number of solutions of the system \mathcal{F} by q^{n-m} , where q is the cardinality of the underlying field \mathcal{F} . In this case, the variety of the ideal $I = \langle \mathcal{F} \rangle$ can have a very complicated structure. Since Gröbner basis algorithms such as F_4 do not work well in this case, it is a good strategy to fix $n - m$ of the variables in order to get a determined system \mathcal{F}' , before applying the Gröbner basis algorithm on \mathcal{F}' .

One can expect that the system \mathcal{F}' has exactly one solution. Therefore, the variety of the ideal $I = \langle \mathcal{F}' \rangle$ has a simple structure and the Gröbner basis algorithm will work well. However, if the system is highly underdetermined ($n > 2m$), we can do even better than fixing variables. We discuss this case in Sect. 8.7.

Furthermore, in some cases, it helps to guess some variables in a determined system \mathcal{F} to create an overdetermined system \mathcal{F}' (Hybrid Approach [5]). In this case, one can not expect that the system \mathcal{F}' has a solution. Indeed, after having guessed k variables, the probability that the system \mathcal{F}' is solvable is q^{-k} . In order to find a solution of the original system \mathcal{F} , one therefore has to create about q^k overdetermined systems \mathcal{F}' (by guessing different values for the k variables to be guessed) and run the Gröbner basis algorithm on each of these systems.

Although this strategy implies running the algorithm q^k times, it often reduces the overall running time. We can estimate the complexity of solving a determined system \mathcal{F} of m equations in m variables using the Hybrid approach by

$$\text{Complexity}_{\text{Hybrid}} = \min_k q^k \left(3 \binom{m-k+d_{\text{reg}}(m,k)}{d_{\text{reg}}(m,k)} \binom{m-k}{2} \right).$$

8.6 Estimating the Degree of Regularity

As shown in the previous section, the complexity of a Gröbner Basis algorithm such as Buchberger's algorithm and F_4/F_5 mainly depends on the degree of regularity d_{reg} . In order to estimate the complexity of these algorithms, we therefore have to study the behavior of the degree of regularity.

8.6.1 Semi-Regular Systems

The idea of semi-regular sequences is very much inspired by the intention of describing a polynomial system of given size that is hardest to solve. The key parameters here are the number of polynomials m , the number of variables n and the degree of the polynomials. From our previous sections, we know that, in order to solve a system of non-homogeneous polynomials, we must somehow find non-trivial relations among the highest degree part of the polynomials of the (extended) system.

Intuitively, the semi-regular sequences can be seen as sequences of homogeneous polynomials which have the smallest possible number of these relations. The idea of semi-regular sequences was introduced in [3] and leads to good theoretical estimates of the complexity of Gröbner basis algorithms for such hard cases. These estimates are strongly supported by a large number of experiments, which show that randomly generated sequences in general are semi-regular. However, we know that most polynomial systems in cryptography, in particular the polynomial systems coming from the HFE cryptosystems, are not at all semi-regular. We will consider these systems in the next subsection.

Due to the fact that semi-regular sequences are important concepts in the theory of solving polynomial systems, but not so important for the complexity analysis of direct attacks on multivariate public key cryptosystems, we will only briefly explain the key definitions and some basic results and refer to [3, 10, 19] for more details.

Currently, the definition of semi-regular sequences is only applicable to the case of $q = 2$, and it is not clear yet how it can be exactly extended to other finite fields. To simplify the exposition and to avoid too much technical details, we will follow a more mathematical definition of semi-regular sequences coming from the work [10, 19] and will explain everything in the context of the finite field \mathbb{F}_2 .

Let $\{f^{(1)}(x_1, \dots, x_n), \dots, f^{(m)}(x_1, \dots, x_n)\}$ be a system of polynomials. Note here that all the polynomials $f^{(i)}$ are treated as functions over \mathbb{F}_2 , where x_i^2 is treated automatically as x_i .

Let d_i be the degree of the polynomial $f^{(i)}$. We define $f_h^{(i)}(x_1, \dots, x_n)$ to be the homogeneous part of the polynomial $f^{(i)}$ of degree d_i .

Next, we will shift our polynomials $f_h^{(i)}(x_1, \dots, x_n)$ to a new setting, namely we will look at them in a new algebra H_n , which we define as

$$H_n = \mathbb{F}_2[x_1, \dots, x_n] / \langle x_1^2, \dots, x_n^2 \rangle.$$

We define H_n^k to be the subspace of homogeneous polynomials of degree exactly k inside H_n . It is clear that

$$H_n = \bigoplus_0^n H_n^i,$$

H_n is a strongly graded \mathbb{F} -algebra with the grading given by the degrees of the polynomials.

We will abuse the terminology by treating, for now, the polynomials $f_h^{(i)}$ and x_i as elements of H_n .

Definition 8.35 A **graded ring** is a ring R , which can be written as the direct sum of Abelian groups R^i such that $R^i R^j \subset R^{i+j}$. The groups R^i are denoted as **homogeneous components** of R . We define the **index** $\text{Ind}(R)$ to be the grade of the highest non-trivial homogeneous component of R .

Therefore we have

$$R = \bigoplus_0^{\text{Ind}(R)} R^i.$$

Definition 8.36 Let $g^{(1)}, \dots, g^{(m)}$ be a set of homogeneous elements in R and $I = \langle g^{(1)}, \dots, g^{(m)} \rangle$ be the ideal generated by $g^{(1)}, \dots, g^{(m)}$. We define

$$\text{Ind}(g^{(1)}, \dots, g^{(m)}) = \text{Ind}(R/I).$$

In the case that R is strongly graded, which means that $R^i R^j = R^{i+j}$, we also have that

$$\text{Ind}(R/I) = \min\{i \geq 0 \mid I \cap R^i = R^i\}.$$

Definition 8.37 Let $g^{(1)}, \dots, g^{(m)}$ be a sequence of non-trivial homogeneous elements of H_n . We call the sequence $g^{(1)}, \dots, g^{(m)}$ **D-semi-regular** if for all $i = 1, 2, \dots, m$ the following condition holds.

If h is a homogeneous element in R such that $hg^{(i)} \in \langle g^{(1)}, \dots, g^{(i-1)} \rangle$ and $\deg(h) + \deg(g^{(i)}) < D$ then $h \in \langle g^{(1)}, \dots, g^{(i)} \rangle$.

A sequence of homogeneous polynomials $g^{(1)}, \dots, g^{(m)}$ is called **semi-regular** if it is D-semi-regular for $D = \text{Ind}(g^{(1)}, \dots, g^{(m)})$.

Definition 8.38 For a graded ring R , the **Hilbert series** HS_R is defined as the function

$$HS_R(z) = \sum_0^{\text{Ind}(R)} (\text{dimension}(R^k))z^k.$$

For a graded ideal I of R , the Hilbert series HS_I is defined as $HS_I = HS_{R/I}$ and

$$HS_I(z) = HS_{R/I}(z).$$

Definition 8.39 For a series $S(z)$, we define the **index** of $S(z)$ to be the lowest degree d such that the coefficient of z^d is non-positive. Otherwise we define the Index of $S(z)$ to be ∞ . For a series $S(z)$, we define the **truncated series** $[S(z)]_I$ to be the series $\sum_0^I s_i z^i$, and $[S(z)]$ to be $[S(z)]_{\text{Ind}(S(z))}$.

Now let us move back to H_n , namely we set $R = H_n$ and look at the polynomials $f_h^{(1)}, \dots, f_h^{(m)}$ as elements in H_n . Then we have

Theorem 8.40 A sequence $f_h^{(1)}, \dots, f_h^{(n)}$ is semi-regular if and only if

$$HS_{(f_h^{(1)}, \dots, f_h^{(m)})}(z) = \frac{(1+z)^n}{\prod_i^m (1+z^{d_i})},$$

where d_i is the degree of $f_h^{(i)}$.

Definition 8.41 In the setting of the above theorem, we define the **degree of regularity** of the ideal $I = \langle f_h^{(1)}, \dots, f_h^{(m)} \rangle$ to be $\text{Ind}(H_n / \langle f_h^{(1)}, \dots, f_h^{(m)} \rangle)$.

In order to study the complexity of the polynomial solving algorithms for non-homogeneous systems $f^{(1)}, \dots, f^{(m)}$, we essentially ignore the lower degree terms and reuse the results from above on $(f_h^{(1)}, \dots, f_h^{(m)})$. In particular, we set the degree of regularity of $(f^{(1)}, \dots, f^{(m)})$ to be the degree of regularity of $f_h^{(1)}, \dots, f_h^{(m)}$. For a multivariate quadratic system \mathcal{F} , we essentially ignore the linear part of $f^{(1)}, \dots, f^{(m)}$, since, in the case of semi-regular sequences, this part has only a minimal impact on the complexity of solving the system.

As we discussed before, the degree of regularity is essentially the highest degree of the polynomials appearing in the process of Gröbner basis algorithms such as F_4 and F_5 . Therefore, the degree of regularity determines the complexity of these algorithms. In [3] it is claimed that the complexity of the polynomial solving algorithms like F_4 can be estimated as $\mathcal{O}(\binom{n+d}{d}^\omega)$, where ω comes from the exponent in the complexity of Gaussian elimination of a square system. It is reasonable to take ω to be 2.39, which is the value typically quoted by the community.

We can give reasonable asymptotic estimates for the degree of regularity of semi-regular multivariate quadratic systems using modern analysis tools. For example, for

the case of $m = 2n$, namely the number of equations is two times the number of variables, the degree of regularity is asymptotically about $0.0858n$. However, the practical meaning of such an estimate is not clear at the moment.

Surely all the formulas presented in this section work only if the sequences are semi-regular. In [2], there are conjectures about the ubiquitousness of semi-regular sequences, but the results in [14, 19] show that the situation is actually very subtle and there remain many interesting and important mathematical questions in the area.

As mentioned earlier in this section, the above discussion is more or less theoretical, and the results on the degree of regularity obtained in this section can not be used to estimate the complexity of a direct attack on a multivariate public key cryptosystems. In order to estimate this complexity, one has to run computer experiments to study the degree of regularity of the public systems produced by the given multivariate scheme.

While, for some multivariate schemes such as UOV and Rainbow (see Chap. 5), one finds that the public systems of the schemes behave very similar to semi-regular sequences, this is not the case for other multivariate schemes. For example, for the public systems of HFE and its variants (see Chap. 4) and SimpleMatrix (see Chap. 7), the degree of regularity is much smaller than indicated above. The reason for this is the large algebraic structure of these schemes. In the next section we show how, in the case of HFE, we can use this algebraic structure to find an upper bound on the degree of regularity of the public systems.

8.6.2 HFE and Variants

Let us first recall the HFE cryptosystem of Chap. 4. The HFE cryptosystem uses a degree n extension field \mathbb{E} of \mathbb{F} as well as an isomorphism $\phi : \mathbb{F}^n \rightarrow \mathbb{E}$. In its basic form, the HFE cryptosystem can be used both as an encryption and signature scheme.

The central map of the HFE cryptosystem is a univariate polynomial map $\mathcal{F} : \mathbb{E} \rightarrow \mathbb{E}$ of the form

$$\mathcal{F}(X) = \sum_{i,j=0}^{q^i+q^j \leq D} \alpha_{ij} X^{q^i+q^j} + \sum_{i=0}^{q^i \leq D} \beta_i X^{q^i} + \gamma$$

with coefficients α_{ij} , β_i and γ randomly chosen from \mathbb{E} . Due to the special form of \mathcal{F} , the map $\tilde{\mathcal{F}} = \phi^{-1} \circ \mathcal{F} \circ \phi$ is a quadratic map over the vector space \mathbb{F}^n . In order to hide the structure of \mathcal{F} in the public key, $\tilde{\mathcal{F}}$ is composed with two affine maps \mathcal{S} and $\mathcal{T} : \mathbb{F}^n \rightarrow \mathbb{F}^n$. Therefore, the public key \mathcal{P} of the scheme is given as

$$\mathcal{P} = \mathcal{S} \circ \tilde{\mathcal{F}} \circ \mathcal{T}$$

and is a quadratic map from \mathbb{F}^n to \mathbb{F}^n .

The *private key* of the scheme consists of the three maps \mathcal{S}, \mathcal{F} and \mathcal{T} (and possibly the isomorphism ϕ).

The *public key* is the multivariate quadratic map $\mathcal{P} : \mathbb{F}^n \rightarrow \mathbb{F}^n$.

As we explained earlier, the systems derived from HFE are no longer semi-regular. Therefore, in order to study the complexity of direct attacks against schemes of the HFE family, we have to extend the definition of the degree of regularity to the non-semi-regular case (see [15]).

Let

$$A = \mathbb{F}[x_1, \dots, x_n] / (x_1^q - x_1, \dots, x_n^q - x_n) ,$$

the algebra of functions from \mathbb{F}^n to \mathbb{F} . Let $p^{(1)}, \dots, p^{(n)}$ be quadratic elements of A . Denote by A_k the subspace of A consisting of functions representable by a polynomial of degree less than or equal to k . For all j we have a natural map $\psi_j : A_j^n \rightarrow \sum_i A_j p^{(i)}$ given by

$$\psi_j(a_1, \dots, a_n) = \sum_i a_i p^{(i)} .$$

The key idea is to look for the so called “degree falls”; a degree fall occurs when the a_i have degree j , but $\sum_i a_i p^{(i)}$ has degree less than degree $j + 2$. Obviously there are trivial degree falls of the form $p^{(j)} p^{(i)} + (-p^{(i)}) p^{(j)}$ or $((p^{(i)})^{q-1} - 1) p^{(i)}$.

The *degree of regularity* of the set $\{p^{(1)}, \dots, p^{(n)}\}$ is now defined as the smallest degree (measured as $\deg a_j + \deg p^{(i)}$) at which a non-trivial degree fall occurs. Note that this idea is very much inspired by the original definition, namely for semi-regular sequences, this definition is the same as the original one.

Here we actually look only at the highest degree terms in the a_i , modulo terms of lower degrees. Mathematically, we actually work again in the associated graded ring $H_n = \mathbb{F}[x_1, \dots, x_n] / (x_1^q, \dots, x_n^q)$. The degree of regularity of the system $\{p^{(1)}, \dots, p^{(n)}\}$ in A will be the smallest degree at which we find non-trivial relations among the leading components $p_h^{(1)}, \dots, p_h^{(n)}$ (considered as elements of H_n).

Denote by H_n^k the subspace of H_n consisting of the homogeneous elements of degree k . Consider an arbitrary set of homogeneous quadratic elements $\{\lambda^{(1)}, \dots, \lambda^{(n)}\} \in H_n^2$. For all j , we have a natural map $\psi_j : (H_n^j)^n \rightarrow H_n^{j+2}$ given by

$$\psi_j(b_1, \dots, b_n) = \sum_i b_i \lambda^{(i)} .$$

Let $Z_j(\lambda^{(1)}, \dots, \lambda^{(n)}) = \ker \psi_j$; this is the subspace of relations of the form $\sum_i b_i \lambda^{(i)} = 0$. Inside $Z_j(\lambda^{(1)}, \dots, \lambda^{(n)})$, we have the subspace $T_j(\lambda^{(1)}, \dots, \lambda^{(n)})$ of trivial relations, which is generated by elements of the form:

1. $b(0, \dots, 0, \lambda^{(j)}, 0, \dots, 0, -\lambda^{(i)}, 0, \dots, 0)$ for $1 \leq i < j \leq n$ and $b \in H_n^{j-2}$; where $\lambda^{(j)}$ is in the i -th position and $-\lambda^{(i)}$ is in the j -th position;
2. $b(0, \dots, 0, (\lambda^{(i)})^{q-1} - 1, 0, \dots, 0)$ for $1 \leq i \leq n$ and $b \in H_n^{j-2(q-1)}$; where $(\lambda^{(i)})^{q-1}$ is in the i -th position;

The space of non-trivial relations is the quotient space

$$Z_j(\lambda^{(1)}, \dots, \lambda^{(n)})/T_j(\lambda^{(1)}, \dots, \lambda^{(n)}).$$

Definition 8.42 The **degree of regularity** of the system $\{\lambda^{(1)}, \dots, \lambda^{(n)}\}$ is defined by

$$d_{\text{reg}}(\{\lambda^{(1)}, \dots, \lambda^{(n)}\}) = \min\{j \mid Z_{j-2}(\{\lambda^{(1)}, \dots, \lambda^{(n)}\})/T_{j-2}(\{\lambda^{(1)}, \dots, \lambda^{(n)}\}) \neq 0\}.$$

Here, the degree of regularity depends only on the subspace generated by the $\lambda^{(i)}$ and not on the order of the elements. Therefore we denote the space generated by the $\lambda^{(i)}$ by V and write $d_{\text{reg}}(V)$ for $d_{\text{reg}}(\{\lambda^{(1)}, \dots, \lambda^{(n)}\})$.

There are two very important properties of the degree of regularity observed in [15]. First, the degree of regularity of a space is less than or equal to the degree of regularity of a subspace.

Proposition 8.43 (Property I) *Let V' be a subspace of V . Then*

$$d_{\text{reg}}(V) \leq d_{\text{reg}}(V').$$

Second, the degree of regularity is invariant under field extension. Let \mathbb{E} be an extension of \mathbb{F} . Define $H_{\mathbb{E}} = \mathbb{E}[x_1, \dots, x_n]/\langle x_1^q, \dots, x_n^q \rangle$ and denote by $V_{\mathbb{E}}$ the \mathbb{E} -subspace of $H_{\mathbb{E}}$ generated by the $\lambda^{(i)}$.

Proposition 8.44 (Property II) *Let \mathbb{E} be an extension of \mathbb{F} . Then*

$$d_{\text{reg}}(V_{\mathbb{E}}) = d_{\text{reg}}(V).$$

Let us now look at \mathcal{P} , a quadratic map with component functions $p^{(1)}, \dots, p^{(n)} \in A$. Let V and V^h be the vector spaces generated by the $p^{(1)}, \dots, p^{(n)}$ and their leading homogeneous components $p_h^{(1)}, \dots, p_h^{(n)}$ which are treated as elements of H_n . It is clear that the goal is actually to find a good bound for $d_{\text{reg}}(V^h)$.

The first step is to extend the base field to \mathbb{E} . When we extend the base field in A , we shift from functions from \mathbb{F}^n to \mathbb{F} to functions from \mathbb{F}^n to \mathbb{E} . Via the linear isomorphism $\phi^{-1}: \mathbb{E} \rightarrow \mathbb{F}^n$, this algebra is isomorphic to the algebra of functions from \mathbb{E} to \mathbb{E} which is simply $\mathbb{E}[X]/\langle X^{q^n} - X \rangle$.

From elementary Galois theory, we know that the space $V_{\mathbb{E}}$ corresponds under this identification to the space generated by $\mathcal{F}, \mathcal{F}^q, \dots, \mathcal{F}^{q^{n-1}}$. If we filter the algebra $\mathbb{E}[X]/\langle X^{q^n} - X \rangle$ by the degree of functions over \mathbb{F} , then the linear component is spanned by $X, X^q, \dots, X^{q^{n-1}}$. The associated graded ring will then be

the algebra $H_{\mathbb{E}} = \mathbb{E}[X_0, \dots, X_{n-1}]/\langle X_1^q, \dots, X_n^q \rangle$ where X_i corresponds to X^{q^i} . This is naturally isomorphic to the algebra H_n with coefficients extended to \mathbb{E} [10], which implies that the process of extending the base field and the process taking the associated graded ring commute.

Let P_i denote the leading component of \mathcal{F}^{q^i} in $H_{\mathbb{E}}$. Thus for instance, if \mathcal{F} is defined as above, then

$$P_0 = \sum_{i,j=0}^{n-1} \alpha_{ij} X_i X_j .$$

The space generated by the P_i is exactly isomorphic to $V_{\mathbb{E}}^h$, the subspace of $H_{\mathbb{E}}$ generated by the $p_h^{(i)}$. By putting all the above together, we can easily reach the following important theorem, stated in [15].

Theorem 8.45

$$d_{\text{reg}}(\{p^{(1)}, \dots, p^{(n)}\}) = d_{\text{reg}}(\{p_h^{(1)}, \dots, p_h^{(n)}\}) = d_{\text{reg}}(\{P_0, \dots, P_{n-1}\}).$$

Using Property II, we get the following immediate corollary.

Corollary 8.46

$$d_{\text{reg}}(\{p^{(1)}, \dots, p^{(n)}\}) \leq \min\{d_{\text{reg}}(Q) \mid Q \in V_{\mathbb{E}}^h\}.$$

Before the work of [10], the bounds on the degree of regularity in [15, 18] and previous works were obtained by counting dimensions. The basic idea is going back to [3, 27]. This approach was refined in [18] by using Property I to reduce to subsets $\{P_0, \dots, P_s\}$ which, for HFE systems, involve significantly fewer variables. It was further refined in [15]. The disadvantage of this approach was that no general formula for the degree of regularity could be derived.

In [10], a completely different approach was developed, namely a purely algebraic approach. Instead of counting and comparing dimensions, the authors of [10] actually looked for specific non-trivial relations. Surprisingly, an important bound could be found by restricting to the case of a single polynomial. They first gave a bound on the degree of regularity of a single polynomial in terms of the rank of the related quadratic form and then applied this simple formula to P_0 , thus obtaining an elegant bound on the degree of regularity of an HFE system in terms of its degree.

The degree of regularity of a single polynomial over \mathbb{F}_2 and \mathbb{F}_3 has been studied in great detail in [13, 14]. However, we do not need the kind of exact information found in those papers, but rather merely need to show the existence of non-trivial relations, which can be done by explicitly using the classification of quadratic forms.

Recall that P_0 is a homogeneous quadratic polynomial in the algebra $\mathbb{E}[X_0, \dots, X_{n-1}]/\langle X_0^q, \dots, X_{n-1}^q \rangle$. Using the classification theorem of quadratic

forms over finite fields, we are able to explicitly construct nontrivial relations and hence derive a simple bound for the degree of regularity of P_0 in terms of its rank.

We now briefly review the classification of quadratic forms over a finite field. We begin with the case when q is odd. A quadratic form in n variables is a homogeneous quadratic polynomial in the polynomial ring $\mathbb{E}[X_1, \dots, X_n]$. Two quadratic forms P and Q are said to be equivalent (written $P \sim Q$), if there is an invertible linear change of variables L , which transforms P into Q :

$$P \circ L(X_1, \dots, X_n) = Q(X_1, \dots, X_n).$$

Pick an element $c \in \mathbb{E}$ that is not a square. Then a standard classical theorem tells us that a quadratic form is equivalent to one of the two types

1. $X_1^2 + \dots + X_{r-1}^2 + X_r^2$
2. $X_1^2 + \dots + X_{r-1}^2 + cX_r^2$

for some $r \leq n$. The same classification applies to quadratic elements of the quotient ring $\mathbb{E}[X_1, \dots, X_n]/\langle X_1^q, \dots, X_n^q \rangle$.

When q is even, the situation is more complicated due to the fact that X^2 is linear rather than quadratic for $q = 2$. It is well known that a quadratic polynomial in the polynomial algebra $\mathbb{E}[X_1, \dots, X_n]$ is equivalent to a polynomial of one of the following forms for some $r \leq n$:

1. $X_1X_2 + \dots + X_{r-1}X_r$
2. $X_1X_2 + \dots + X_{r-2}X_{r-1} + X_r^2$
3. $X_1X_2 + \dots + X_{r-1}X_r + X_{r-1}^2 + cX_r^2$ where $c \in \mathbb{E} \setminus \{0\}$ satisfies $\text{TR}_{\mathbb{E}}(c) = 1$.

For $q > 2$, this classification carries over to the quotient ring $\mathbb{E}[X_1, \dots, X_n]/\langle X_1^q, \dots, X_n^q \rangle$. When $q = 2$, all quadratic elements of the quotient ring are equivalent to an element of the first type. In all cases the number r is known as the rank of Q . Note that if $q = 2$, the rank of a quadratic element must be at least 2.

When $r = 1$ (in the case $q > 2$), Q is actually equal to aX_1^2 for some $a \in \mathbb{E}$. It is easily verified that the smallest non-trivial relation is $X^{q-2}(aX^2) = 0$ and hence that $d_{\text{reg}}(Q) = q$. More generally we have the following inequality.

Theorem 8.47 *Let Q be quadratic of rank r . If $r > 1$,*

$$d_{\text{reg}}(Q) \leq \frac{r(q-1)}{2} + 2.$$

Proof In the case of a single polynomial, the definition of the degree of regularity can be expressed in terms of non-trivial annihilators. Let Q be an arbitrary quadratic element of $H = \mathbb{E}[X_1, \dots, X_n]/\langle X_1^q, \dots, X_n^q \rangle$. The annihilators of Q are the elements of $\text{Ann}(Q) = \{f \in H \mid fQ = 0\}$. The trivial annihilators are the multiples of Q^{q-1} . The degree of regularity is the smallest k such that there is a non-trivial annihilator of Q of degree $k - 2$. The degree of regularity is invariant

under a linear change of variables, so it is sufficient to prove the result by exhibiting explicit non-trivial annihilators for each of the above types of quadratic elements.

Because of the different types of standard forms, we need to consider separately the cases when q is odd and even. We also need to divide these cases into the cases when r is odd or even.

- Case 1: q odd, r even

Set $s = r/2$. In this case Q is of the form

$$Q = X_1^2 + X_2^2 + \cdots + X_{2s-1}^2 + aX_{2s}^2$$

for some $a \in \mathbb{E}$. Let

$$K_i = X_{2i-1}^{q-1} - X_{2i}^2 X_{2i-1}^{q-3} + X_{2i}^4 X_{2i-1}^{q-5} + \cdots + (-1)^{(q-1)/2} X_{2i}^{q-1}$$

for $i = 1, \dots, s-1$; and

$$K_s = X_{2s-1}^{q-1} - aX_{2s}^2 X_{2s-1}^{q-3} + a^2 X_{2s}^4 X_{2s-1}^{q-5} + \cdots + (-a)^{(q-1)/2} X_{2s}^{q-1}$$

Set

$$K = K_1 K_2 \cdots K_s .$$

It is clear that

$$K_i (X_{2i-1}^2 + X_{2i}^2) = X_{2i-1}^{q+1} - (-1)^{(q+1)/2} X_{2i}^{q+1} = 0 ,$$

for $i = 1, \dots, s-1$; and

$$K_s (X_{2s-1}^2 + aX_{2s}^2) = X_{2s-1}^{q+1} - (-a)^{(q+1)/2} X_{2s}^{q+1} = 0 .$$

Hence $KQ = 0$, which implies $K \in \text{Ann}(Q) \cap H^{s(q-1)}$. We claim that $K \notin \langle Q^{q-1} \rangle$. Consider the quotient algebra

$$\bar{H} = H / \left\langle X_{2i-1}^2 + X_{2i}^2, i = 1, \dots, s-1; X_{2s-1}^2 + aX_{2s}^2 \right\rangle .$$

The algebra \bar{H} has a basis consisting of monomials with the powers of the variables X_2, X_4, \dots, X_{2s} at most 1. It is clear that the image of Q (and hence also Q^{q-1}) in \bar{H} is zero, whereas the image of K is

$$\prod_i^s X_{2i-1}^{q-1} \left(\frac{q+1}{2} \right)^s$$

which is non-zero. Therefore K is not in the ideal generated by Q^{q-1} . Hence $d_{\text{reg}}(Q) \leq r(q-1)/2 + 2$.

- Case 2: q odd, r odd

Set $s = (r-1)/2$. In this case Q is of the form

$$Q = X_1^2 + X_2^2 + \cdots + X_{2s-1}^2 + X_{2s}^2 + aX_{2s+1}^2$$

for some $a \in \mathbb{E}$. From the classification of quadratic forms, we have

$$X_{2s-1}^2 + X_{2s}^2 + aX_{2s+1}^2 \sim X_{2s-1}^2 - X_{2s}^2 - aX_{2s+1}^2 \sim X_{2s-1}X_{2s} - aX_{2s+1}^2$$

so Q can be taken to be of the form:

$$Q = X_1^2 + X_2^2 + \cdots + X_{2s-2}^2 + X_{2s-1}X_{2s} - aX_{2s+1}^2 .$$

Let

$$K_i = X_{2i-1}^{q-1} - X_{2i}^2 X_{2i-1}^{q-3} + X_{2i}^4 X_{2i-1}^{q-5} + \cdots + (-1)^{(q-1)/2} X_{2i}^{q-1}$$

for $i = 1, \dots, s-1$; and

$$K' = \frac{(X_{2s-1}X_{2s})^{(q+1)/2} - a^{(q+1)/2} X_{2s+1}^{(q+1)}}{X_{2s-1}X_{2s} - aX_{2s+1}^2} X_{2s-1}^{(q-1)/2} .$$

Note that

$$K'(X_{2s-1}X_{2s} - aX_{2s+1}^2) = (X_{2s-1}X_{2s})^{(q+1)/2} X_{2s-1}^{(q-1)/2} = 0 .$$

Set

$$K = K_1 K_2 \cdots K_{s-1} K' .$$

Note that the degree of K is $(s-1)(q-1) + 3(q-1)/2 = r(q-1)/2$. Again we see that $KQ = 0$ and therefore $K \in \text{Ann}(Q) \cap H^{r(q-1)/2}$. Consider the quotient algebra

$$\bar{H} = H / \left\langle X_{2i-1}^2 + X_{2i}^2, i = 1, \dots, s-1; X_{2s-1}X_{2s} - aX_{2s+1}^2 \right\rangle ,$$

Then \bar{H} has a basis consists of monomials in which the powers of the variables $X_2, X_4, \dots, X_{2(s-1)}, X_{2s+1}$ are at most one. The image of Q in \bar{H} is zero, but that of K is

$$\prod_{i=1}^{s-1} X_{2i-1}^{q-1} \left(\frac{q+1}{2} \right)^{s-1} X_{2s-1}^{q-1} X_{2s}^{(q-1)/2} \left(\frac{q+1}{2} \right)$$

which is non-zero. Hence K is not in the ideal generated by Q^{q-1} and $d_{\text{reg}}(Q) \leq r(q-1)/2 + 2$.

- Case 3: q even, r even (Q of type (1) or (3))

First suppose that Q is of the form $Q = X_1 X_2 + \cdots + X_{2s-1} X_{2s}$ where $r = 2s$. Set $G = X_1^{q-1} X_3^{q-1} \cdots X_{2s-1}^{q-1}$. Then it is easily seen that $H \in \text{Ann}(Q) \cap H^{s(q-1)}$. Consider the quotient algebra

$$\bar{H} = H / \langle X_1 - X_2, \dots, X_{2s-1} - X_{2s} \rangle.$$

The image of Q in \bar{H} is $\bar{Q} = X_1^2 + X_3^2 + \cdots + X_{2s-1}^2 = (X_1 + X_3 + \cdots + X_{2s-1})^2$, so the image of Q^{q-1} is $\bar{Q}^{q-1} = 0$. On the other hand, the image of G is $X_1^{q-1} X_3^{q-1} \cdots X_{2s-1}^{q-1}$ which is non-zero. Thus $G \notin \langle Q^{q-1} \rangle$. Hence $d_{\text{reg}}(Q) \leq r(q-1)/2 + 2$.

Next suppose that Q is of the form $Q = X_1 X_2 + \cdots + X_{2s-1} X_{2s} + X_{2s-1}^2 + \alpha X_{2s}^2$. Let \mathbb{L} be a finite extension field of \mathbb{E} in which the equation $1 + X + \alpha X^2$ has a root. In $\mathbb{L}[X_1, \dots, X_r]$, Q is equivalent to $X_1 X_2 + \cdots + X_{2s-1} X_{2s}$. Since the degree of regularity is invariant under extensions of the base field by Property II, it follows from the first part that $d_{\text{reg}} \leq r(q-1)/2 + 2$.

- Case 4: q even, r odd (Q of type (2))

Note that, in this case, we must have $q > 2$. We may assume that Q is of the form $Q = X_1 X_2 + \cdots + X_{2s-1} X_{2s} + X_{2s+1}^2$ where $r = 2s + 1$. Set

$$G = X_1^{q-1} X_3^{q-1} \cdots X_{2s-3}^{q-1} X_{2s-1}^{q/2} (X_{2s-1} X_{2s} + X_{2s+1}^2)^{(q-2)/2}.$$

Note that $\deg H = r(q-2)/2$ and

$$GQ = (X_{2s-1} X_{2s} + X_{2s+1}^2)^{q/2} X_{2s-1}^{q/2} = X_{2s-1}^q X_{2s}^{q/2} + X_{2s+1}^q X_{2s-1}^{q/2} = 0.$$

Consider the quotient algebra

$$\bar{H} = H / \langle X_1 - X_2, \dots, X_{2s-1} - X_{2s} \rangle.$$

The image of Q in \bar{H} is $\bar{Q} = X_1^2 + X_3^2 + \cdots + X_{2s-1}^2 + X_{2s+1}^2 = (X_1 + X_3 + \cdots + X_{2s-1} + X_{2s+1})^2$, so the image of Q^{q-1} is $\bar{Q}^{q-1} = 0$. On the other hand, the image of G is $X_1^{q-1} X_3^{q-1} \cdots X_{2s-3}^{q-1} X_{2s-1}^{q/2} (X_{2s-1} + X_{2s+1})^{(q-2)/2}$ which is non-zero. Thus $G \notin \langle Q^{q-1} \rangle$ and hence $d_{\text{reg}}(Q) \leq r(q-1)/2 + 2$. \square

Let us define the Q-Rank of a quadratic operator $P(X)$ to be the minimal rank of elements of the space $V_{\mathbb{E}}^h$ generated by P_0, \dots, P_{n-1} , i.e.

$$\text{Q-Rank } P = \min\{\text{Rank } Q \mid Q \in V_{\mathbb{E}}^h\}$$

Note in particular that $\text{Q-Rank}(P) \leq \text{Rank}(P_0)$.

Theorem 8.48 *Let P be a quadratic operator of degree D . If $\text{Q-Rank}(P) > 1$, the degree of regularity of the associated system is upper bounded by*

$$\frac{(q-1) \text{Q-Rank}(P)}{2} + 2 .$$

In particular, this is less than or equal to

$$\frac{(q-1)(\lfloor \log_q(D-1) \rfloor + 1)}{2} + 2 .$$

If $\text{Q-Rank}(P) = 1$, then the degree of regularity is less than or equal to q .

Proof The first assertion follows from Theorem 8.47 and Corollary 8.46. Suppose that

$$P(X) = \sum_{q^i + q^j \leq D} a_{ij} X^{q^i + q^j} + \sum_{q^i \leq D} b_i X^{q^i} + c .$$

Then

$$P_0 = \sum_{q^i + q^j \leq D} a_{ij} X_i X_j .$$

Let k be the largest subscript of a variable X_k that occurs non-trivially in P_0 (that is, $a_{ik} \neq 0$ for some i). The rank of P_0 is bounded by the number of variables involved in its expression which is at most $k+1$. On the other hand, by our assumption on D , $D \geq q^k + 1$ or equivalently, $k \leq \lfloor \log_q(D-1) \rfloor$. Thus the rank of P_0 is at most $\lfloor \log_q(D-1) \rfloor + 1$. \square

A very interesting case is the Matsumoto-Imai scheme, where $\mathcal{F}(X) = X^{1+2^\theta}$ over the field $GF(2)$. Then $P_0 = X_0 X_\theta$ has rank 2. So our theorem implies that the degree of regularity is less than or equal to three. This is precisely the statement that linearization equations exist in this case [23]. On the other hand if we consider a Matsumoto-Imai operator over a field of order $q = 2^m$, then the degree of regularity remains 3 but our bound is $2^m + 1$. Therefore our estimate formula needs to be improved when q is not a prime.

For fixed q , the degree of regularity of an HFE public key is $O(\log_q D)$. Consider now a Gröbner basis attack on an HFE system of degree D . We continue to make

the assumption that these algorithms will terminate at degree equal to the degree of regularity or shortly after this. The runtime of this algorithm will be $O(n^{\omega D_{\text{reg}}})$. Assuming that the security parameter is chosen in such a way that $D = O(n^\alpha)$, the runtime for the Gröbner basis attack on an HFE system over any base field will be $2^{O(\log(n)^2)}$; that is, it will be quasi-polynomial.

On the other hand, suppose that q itself is a component of the security parameter and is taken to be of scale $O(n)$ (this assumption is reasonable since it will only increase the computation complexity for HFE systems by the scale of $O(\log_2 n)$). If the bound above is asymptotically sharp, then the degree of regularity will be at least of the scale $O(n)$, and therefore inverting HFE systems will be exponential.

We do not expect or believe the bound obtained in Theorem 8.48 to be optimal in any degree of generality. If we compare the bound

$$(q-1)(\lfloor \log_q(D-1) \rfloor + 1)/2 + 2$$

with that obtained in [15] for a large number of values of n and D and prime q . We can see that as n becomes large relative to q , the two bounds appear to be getting closer, though the bound in [10] are frequently slightly higher. It seems possible that there may be a tighter upper bound of the form $cq \log_q(D)$ for some scalar c when q is a prime.

Later, this results were further extended to get upper bounds on the degree of regularity of HFE variants such as HFE-, HFEv- and HFEv- [11, 12].

Theorem 8.49 *The degree of regularity of the polynomial system derived from an HFEv- system is less than or equal to*

$$\begin{aligned} & \frac{(q-1)(r+v+a-1)}{2} + 2 \text{ if } q \text{ is even and } r+a \text{ is odd,} \\ & \frac{(q-1)(r+v+a)}{2} + 2 \text{ otherwise.} \end{aligned}$$

where q is the size of the base field, D is the degree of the HFE polynomial, and a and v are the numbers of Minus equations and Vinegar variables respectively. The “rank” parameter r is given by $r = \lfloor \log_q(D-1) \rfloor + 1$.

When $a = 0$, which gives us the HFEv scheme, the degree of regularity is upper bounded by

$$\begin{aligned} & \frac{(q-1)(r+v-1)}{2} + 2 \text{ if } q \text{ is even and } r \text{ is odd,} \\ & \frac{(q-1)(r+v)}{2} + 2 \text{ otherwise.} \end{aligned}$$

When $v = 0$, which gives us the HFE- scheme, the degree of regularity is less than or equal to

$$\frac{(q-1)(r+a-1)}{2} + 2 \text{ if } q \text{ is even and } r+a \text{ is odd,}$$

$$\frac{(q-1)(r+a)}{2} + 2 \text{ otherwise.}$$

Though these are nice formula, we know for general q , especially large q , that they are not so tight and need much improvement. However, for the case $q = 2$ and currently practical parameters, the above formulas seem to be very accurate bounds, which was verified by a large number of experiments [24]. These theoretical bounds are actually used to select the parameters in some the submissions to the NIST standardization process of post-quantum cryptosystems [21].

8.7 Algorithms for Solving Over- and Underdetermined Systems

As discussed in the previous sections, the algorithms used to solve multivariate quadratic systems require in general exponential time. However, if the systems considered are either highly overdetermined ($m \gg n$) or highly underdetermined ($n \gg m$), there exist special algorithms which run in polynomial time. In this section, we present two of these algorithms. Furthermore, we discuss here an algorithm which solves underdetermined systems of m equations in $n = \nu m$ variables by solving a determined system of $m - \lfloor \nu \rfloor + 1$ equations and variables. This algorithm plays an important role in the security analysis of the UOV scheme (see Chap. 5).

8.7.1 Solving Overdetermined Systems with $m \sim n^2$

If the number of equations in a multivariate quadratic system is quadratic in the number of variables, the system can be solved by the Relinearization method. In particular, this method works if we have

$$m \geq \frac{n(n+3)}{2}.$$

The algorithm consists of two steps.

1. Interpret each quadratic monomial $x_i x_j$ as a new variable x_{ij} . Therefore one obtains a system of m linear equations in the $\frac{n(n+1)}{2} + n = \frac{n(n+3)}{2}$ variables

$$\underbrace{x_{11}, x_{12}, \dots, x_{1n}, x_{22}, x_{23}, \dots, x_{nn}}_{\text{former quadratic monomials}}, \underbrace{x_1, x_2, \dots, x_n}_{\text{linear monomials}}.^2$$

2. Solve the linear system generated in the previous step by Gaussian elimination. If $m \geq \frac{n(n+3)}{2}$, the linear system will have (with high probability) exactly one solution, which corresponds to the solution of the original quadratic system.

The complexity of the algorithm is $O(n^{2\omega})$, where $2 < \omega \leq 3$ is the exponent of solving a linear system. Therefore, the algorithm solves the system in polynomial time.

Remember that we used the above technique in the decryption process of the (rectangular) SimpleMatrix scheme (see Chap. 7).

8.7.2 Solving Underdetermined Systems with $n \sim m^2$

In [20], Hashimoto et al. proposed an algorithm to solve an underdetermined multivariate quadratic system \mathcal{P} of m equations in $n \geq \frac{m(m+3)}{2}$ variables in polynomial time. The idea of the algorithm is to decompose the system \mathcal{P} into $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$, where \mathcal{S} and \mathcal{T} are linear transformations. From \mathcal{F} one extracts a number of expressions, which lead to $m(m+1)/2$ linear equations in $n-m$ variables. If this linear system has a solution, then this solution is substituted back into the transformed system \mathcal{F} resulting in an easily invertible quadratic map. Note that this idea is very similar to a structural attack against a multivariate public key cryptosystem. The process of solving the equation $\mathcal{P}(\mathbf{z}) = 0$ consists of three steps.

Recall that a general system \mathcal{P} of multivariate quadratic equations is given by

$$\mathcal{P} : \quad p^{(k)} = \sum_{1 \leq i, j \leq n} a_{ij}^{(k)} x_i x_j + \sum_{1 \leq i \leq n} b_i^{(k)} x_i + c^{(k)}, \text{ for } k = 1, \dots, m \quad (8.7)$$

The system \mathcal{P} can also be written in matrix notation, i.e.

$$\mathcal{P} : \quad p^{(k)} = \mathbf{x}^T A^{(k)} \mathbf{x} + \mathbf{b}^{(k)} \mathbf{x} + c^{(k)}$$

with $n \times n$ matrices $A^{(k)} = (a_{ij}^{(k)})$ containing the coefficients of the homogeneous quadratic terms, vectors $\mathbf{b}^{(k)} = (b_1^{(k)}, \dots, b_n^{(k)})$ containing the coefficients of the linear terms and field elements $c^{(k)}$ ($k = 1, \dots, m$).

The system \mathcal{P} is transformed iteratively into a system \mathcal{F} of form (8.8) where each of the m steps consists of two linear transformations. The two transformations depend only on the quadratic terms (or the matrices $A^{(k)}$), but will be applied also

²Note that, in the case of $\mathbb{F} = GF(2)$, we only need $m \geq \frac{n(n+1)}{2}$ equations, since the quadratic monomials x_i^2 ($i = 1, \dots, n$) don't exist.

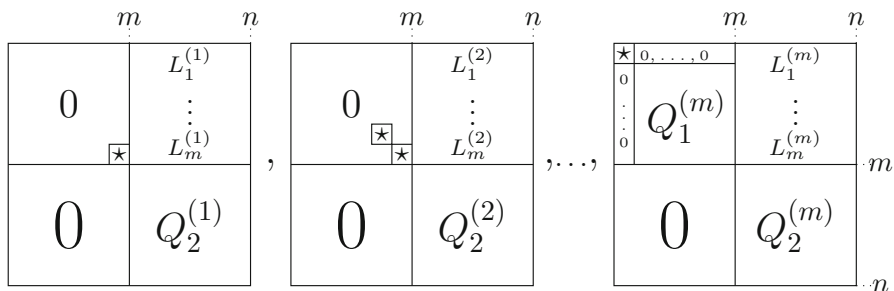


Fig. 8.1 Structure of the homogeneous part of the system \mathcal{F}

to the linear terms in order to keep everything consistent. When it is clear from the context we will write sometimes $a_m^{(k)}$ instead of $a_{m,m}^{(k)}$ to refer to the coefficient of x_m^2 of the polynomial $p^{(k)}$.

The idea of the algorithm is outlined below and described in more details later.

1. Decompose the system \mathcal{P} into $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$, where \mathcal{S} and \mathcal{T} are linear maps and \mathcal{F} is a set of quadratic equations of the form

$$\begin{aligned}
 a_m^{(1)} x_m^2 + \sum_{1 \leq i \leq m} x_i L_i^{(1)} + Q_2^{(1)} &= 0, \\
 a_{m-1}^{(2)} x_{m-1}^2 + Q_1^{(2)}(x_m) + \sum_{1 \leq i \leq m} x_i L_i^{(2)} + Q_2^{(2)} &= 0, \\
 a_{m-2}^{(3)} x_{m-2}^2 + Q_1^{(3)}(x_{m-1}, x_m) + \sum_{1 \leq i \leq m} x_i L_i^{(3)} + Q_2^{(3)} &= 0, \\
 &\vdots \\
 a_{m-\ell+1}^{(\ell)} x_{m-\ell+1}^2 + Q_1^{(\ell)}(x_{m-\ell+2}, \dots, x_m) + \sum_{1 \leq i \leq m} x_i L_i^{(\ell)} + Q_2^{(\ell)} &= 0, \\
 &\vdots \\
 a_1^{(m)} x_1^2 + Q_1^{(m)}(x_2, \dots, x_m) + \sum_{1 \leq i \leq m} x_i L_i^{(m)} + Q_2^{(m)} &= 0.
 \end{aligned} \tag{8.8}$$

For $i, j = 1, \dots, m$, the $L_j^{(i)} = L_j^{(i)}(x_{m+1}, \dots, x_n)$ are m^2 linear functions and each $Q_2^{(i)} = Q_2^{(i)}(x_{m+1}, \dots, x_n)$ is a quadratic polynomial (with linear and constant terms) in the given variables. The functions $Q_1^{(j)}$ are homogeneous quadratic functions in the listed variables. For example $Q_1^{(2)}(x_m) = a_m^{(2)} x_m^2$. The structure of the equations of \mathcal{F} is shown in Fig. 8.1.

2. Consider the linear equations

$$L_i^{(j)}(x_{m+1}, \dots, x_n) = 0 \quad \text{for } j = 1, \dots, m, \quad i = 1, \dots, m+1-j, \quad (8.9)$$

and solve this system of $m(m+1)/2$ equations for (x_{m+1}, \dots, x_n) . In order for a solution to exist we need at least as many unknowns as equations, i.e.

$$n - m \geq \frac{m(m+1)}{2} \quad \text{or} \quad n \geq \frac{m(m+3)}{2}.$$

3. After having found a solution for x_{m+1}, \dots, x_n , we substitute the values of x_{m+1}, \dots, x_n into the system \mathcal{F} to obtain a system \tilde{F} of the form

$$\begin{aligned} a_m^{(1)}x_m^2 + \overline{Q}_2^{(1)} &= 0, \\ a_{m-1}^{(2)}x_{m-1}^2 + Q_1^{(2)}(x_m) + x_m\overline{L}_m^{(2)} + \overline{Q}_2^{(2)} &= 0, \\ a_{m-2}^{(3)}x_{m-2} + Q_1^{(3)}(x_{m-1}, x_m) + x_{m-1}\overline{L}_{m-1}^{(3)} + x_m\overline{L}_m^{(3)} + \overline{Q}_2^{(3)} &= 0, \\ &\vdots \\ a_{m-\ell+1}^{(\ell)}x_{m-\ell+1}^2 + Q_1^{(\ell)}(x_{m-\ell+2}, \dots, x_m) + \sum_{m-\ell+2 \leq i \leq m} x_i\overline{L}_i^{(\ell)} + \overline{Q}_2^{(\ell)} &= 0, \\ &\vdots \\ a_1^{(m)}x_1^2 + Q_1^{(m)}(x_2, \dots, x_m) + \sum_{2 \leq i \leq m} x_i\overline{L}_i^{(m)} + \overline{Q}_2^{(m)} &= 0. \end{aligned} \quad (8.10)$$

Note that $\overline{L}_j^{(i)}$ and $\overline{Q}_2^{(i)}$ are the values of $L_j^{(i)}$ and $Q_2^{(i)}$ after being evaluated with the result from the previous step and therefore are constants. If a square root exists for the first equation in (8.10) we can solve for x_m . By substituting this solution into the second equation, we can hope to get a solution for x_{m-1} . We repeat this process until we get a value for x_1 .

8.7.2.1 Finding the Decomposition of \mathcal{P}

In the first iteration we ensure that the element $a_{11}^{(m)}$ is not zero, if necessary by interchanging $p^{(m)}$ with another equation. Then we set all coefficients of x_1^2 in $p^{(i)}$ ($i \in \{1, \dots, m-1\}$) to 0 by computing $\tilde{p}^{(i)} = p^{(i)} - \frac{a_{11}^{(i)}}{a_{11}^{(m)}}p^{(1)}$. This transformation corresponds to a linear transformation $\tilde{\mathcal{P}} = \mathcal{S}_m \circ \mathcal{P}$ where, for the general case, the matrix \mathcal{S}_ℓ is given by

$$S_\ell = \begin{pmatrix} 1 & 0 & \dots & 0 & s_{1,\ell} & 0 & \dots & 0 \\ 0 & 1 & 0 & \vdots & s_{2,\ell} & \vdots & & \vdots \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ 0 & \dots & 0 & 1 & s_{\ell-1,\ell} & \vdots & & \vdots \\ 0 & \dots & \dots & 0 & 1 & 0 & \dots & 0 \\ \vdots & & & \vdots & 0 & 1 & 0 & \dots & 0 \\ \vdots & & & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \dots & 0 & 0 & 0 & \dots & 0 & 1 \end{pmatrix}. \quad (8.11)$$

and $s_{i,\ell} = a_{jj}^{(i)} / a_{jj}^{(\ell)}$.

The second transformation at each iteration is a linear transformation of the variables. For this we introduce a $n \times n$ matrix T_ℓ ($\ell = 2, \dots, m$) of the form

$$T_\ell = \begin{pmatrix} 1 & 0 & \dots & 0 & t_{1,\ell} & 0 & \dots & 0 \\ 0 & 1 & 0 & \vdots & t_{2,\ell} & \vdots & & \vdots \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ 0 & \dots & 0 & 1 & t_{\ell-1,\ell} & \vdots & & \vdots \\ 0 & \dots & \dots & 0 & t_{\ell,\ell} & 0 & \dots & 0 \\ \vdots & & & \vdots & t_{\ell+1,\ell} & 1 & 0 & \dots & 0 \\ \vdots & & & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \dots & 0 & t_{n,\ell} & 0 & \dots & 0 & 1 \end{pmatrix}. \quad (8.12)$$

The matrix T_ℓ is the identity matrix with the ℓ -th column replaced by the unknowns $t_{j,\ell}$ ($j = 1, \dots, n$). The quadratic terms are then transformed into $\tilde{A}^{(k)} = T_\ell^T A^{(k)} T_\ell$. In the resulting matrix the terms in the ℓ -th column and ℓ -th row are dot products of the vector of unknowns and columns and/or rows of $A^{(k)}$. The exception is the term at the intersection of the ℓ -th row and the ℓ -th column, where the unknowns appear in a quadratic form, but the explicit form of that term is not needed. All other terms remain unchanged in this transformation.

For $j \neq \ell$, the transformed terms in column ℓ and row ℓ are given by

$$\begin{aligned} \tilde{a}_{j,\ell}^{(k)} &= \sum_{1 \leq i \leq n} a_{i,j}^{(k)} t_{i,\ell} \\ \tilde{a}_{\ell,j}^{(k)} &= \sum_{1 \leq i \leq n} a_{j,i}^{(k)} t_{i,\ell} \end{aligned}$$

We are only interested in the transformed terms with $1 \leq j < \ell$ and by setting

$$\tilde{a}_{j\ell}^{(k)} + \tilde{a}_{\ell j}^{(k)} = 0 \quad (j < \ell)$$

$$\left(\begin{array}{cccc|c} 0 & 0 & \star & \star & \cdots \\ 0 & \star & \star & \star & \cdots \\ \star & \star & \star & \star & \cdots \\ \star & \star & \star & \star & \cdots \\ \hline \vdots & \vdots & \vdots & \vdots & \ddots \end{array} \right) \quad
\left(\begin{array}{cccc|c} 0 & 0 & \star & \star & \cdots \\ 0 & \star & \star & \star & \cdots \\ \star & \star & \star & \star & \cdots \\ \star & \star & \star & \star & \cdots \\ \hline \vdots & \vdots & \vdots & \vdots & \ddots \end{array} \right) \quad
\left(\begin{array}{cccc|c} 0 & 0 & \star & \star & \cdots \\ 0 & \star & \star & \star & \cdots \\ \star & \star & \star & \star & \cdots \\ \star & \star & \star & \star & \cdots \\ \hline \vdots & \vdots & \vdots & \vdots & \ddots \end{array} \right) \quad
\left(\begin{array}{cccc|c} a_1 & 0 & \star & \star & \cdots \\ 0 & \star & \star & \star & \cdots \\ \star & \star & \star & \star & \cdots \\ \star & \star & \star & \star & \cdots \\ \hline \vdots & \vdots & \vdots & \vdots & \ddots \end{array} \right)$$

Fig. 8.2 Structure of the system \mathcal{P} after the first iteration of the algorithm ($m = 4$; a_1 stands for $a_{1,1}^{(4)}$)

the corresponding coefficients of $x_j x_\ell$ with $j < \ell$ and $1 \leq k \leq m$ can be fixed to zero. This results in the following system of homogeneous linear equations for the unknowns $t_{1,\ell}$ to $t_{n,\ell}$

$$\sum_{1 \leq i \leq n} (a_{j,i}^{(k)} + a_{i,j}^{(k)}) t_{i,\ell} = 0 \quad \text{for } j < \ell \quad (8.13)$$

For this system only a non-trivial solution with $t_{\ell,\ell} \neq 0$ is of interest.

In particular we use, in the first iteration, the matrix T_2 in order to eliminate all terms with $x_1 x_2$. As seen from the above expressions this results in a system of m homogeneous linear equations in the n unknowns $t_{1,2}, t_{2,2}, \dots, t_{n,2}$. The terms, which are eliminated from the polynomials $p^{(1)}, \dots, p^{(m)}$, are displayed in Fig. 8.2 (for the case of $m = 4$). By the transformation T_2 , the system \mathcal{P} is transformed to a system $\tilde{\mathcal{P}}$, but we stay with the same notation by setting $\mathcal{P} = \tilde{\mathcal{P}}$.

During the second iteration of the algorithm we check that the new term $a_{2,2}^{m-1}$ is not zero; if necessary we interchange $p^{(m-1)}$ with another $p^{(j)}$ where $1 \leq j < m - 1$. Next, we eliminate the terms with x_2^2 in the equations $p^{(1)}, \dots, p^{(m-2)}$, by subtracting the appropriate multiple of $p^{(m-1)}$ from these polynomials. This transformation can be represented by the matrix S_{m-1} .

For the other transformation in the second iteration of the algorithm we use the matrix T_3 . This time we eliminate the terms with $x_1 x_3$ and $x_2 x_3$ from $p^{(1)}$ to $p^{(m-1)}$, but only $x_1 x_3$ from $p^{(m)}$. From (8.13) we obtain $2m - 1$ homogeneous equations in the n unknowns $t_{1,3}, \dots, t_{n,3}$. Assuming that there exists a nontrivial solution, we can transform the system into a system of the form shown in Fig. 8.3.

Figure 8.4 shows the system after the third iteration. Note that this time the terms of $x_1 x_4$, $x_2 x_4$ and $x_3 x_4$ are eliminated from $p^{(1)}$ to $p^{(m-2)}$, but only $x_1 x_4$ and $x_2 x_4$ from $p^{(m-1)}$, and only $x_1 x_4$ from $p^{(m)}$. The figure also shows that, for $m = 4$, 3 iterations suffice to bring the system into the desired form of (8.8) and Fig. 8.1, as the next iteration would only check that $a_{m,m}^{(1)}$ is nonzero. For the general case with m equations we need analogously $m - 1$ iterations.

Algorithm 8.10 computes step by step the two transformations \mathcal{S} and \mathcal{T} and a map \mathcal{F} of the form of (8.8) such that $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$. In particular, it finds linear

$$\begin{pmatrix} 0 & 0 & 0 & \star & \cdots \\ 0 & 0 & 0 & \star & \cdots \\ 0 & 0 & \star & \star & \cdots \\ \star & \star & \star & \star & \cdots \\ \hline \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \quad
\begin{pmatrix} 0 & 0 & 0 & \star & \cdots \\ 0 & 0 & 0 & \star & \cdots \\ 0 & 0 & \star & \star & \cdots \\ \star & \star & \star & \star & \cdots \\ \hline \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \quad
\begin{pmatrix} 0 & 0 & 0 & \star & \cdots \\ 0 & a_2 & 0 & \star & \cdots \\ 0 & 0 & \star & \star & \cdots \\ \star & \star & \star & \star & \cdots \\ \hline \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \quad
\begin{pmatrix} a_1 & 0 & 0 & \star & \cdots \\ 0 & \star & \star & \star & \cdots \\ 0 & \star & \star & \star & \cdots \\ \star & \star & \star & \star & \cdots \\ \hline \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

Fig. 8.3 Structure of the system \mathcal{P} after the second iteration of the algorithm ($m = 4$; a_2 stands for $a_{2,2}^{(3)}$)

$$\begin{pmatrix} 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & \star & \cdots \\ \hline \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \quad
\begin{pmatrix} 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & a_3 & 0 & \cdots \\ 0 & 0 & 0 & \star & \cdots \\ \hline \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \quad
\begin{pmatrix} 0 & 0 & 0 & 0 & \cdots \\ 0 & a_2 & 0 & 0 & \cdots \\ 0 & 0 & \star & \star & \cdots \\ 0 & 0 & \star & \star & \cdots \\ \hline \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \quad
\begin{pmatrix} a_1 & 0 & 0 & 0 & \cdots \\ 0 & \star & \star & \star & \cdots \\ 0 & \star & \star & \star & \cdots \\ 0 & \star & \star & \star & \cdots \\ \hline \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

Fig. 8.4 Structure of the system \mathcal{P} after the third iteration of the algorithm ($m = 4$; a_3 stands for $a_{3,3}^{(2)}$)

transformations S_2, \dots, S_m and T_2, \dots, T_m such that

$$\mathcal{F} = \underbrace{S_2 \circ \dots \circ S_m}_{\mathcal{S}^{-1}} \circ \mathcal{P} \circ \underbrace{T_2 \circ \dots \circ T_m}_{\mathcal{T}^{-1}}.$$

The algorithm as stated below assumes that a non zero term $a_{k,k}^{(m-k)} \neq 0$ can always be found. This might not always be the case. If it happens, we use the identity matrix for the first transformation and compute the second transformation as before.

Although the algorithm is described with references to polynomials, it is much easier to implement it with matrices which represent these polynomials. Here one has the option to use either symmetric matrices or an upper triangular form. The transformation T_ℓ does not preserve the chosen form, and it appears to be beneficial to restore it after each iteration.

8.7.2.2 Solving the System

After having found the decomposition of \mathcal{P} (and therefore a system \mathcal{F} of form (8.8)), we solve the linear equations $L_i^{(j)}(x_{m+1}, \dots, x_n) = 0$ ($i \in \{1, \dots, m\}$, $j \in$

Algorithm 8.10 Transforming the system \mathcal{P} into a system \mathcal{F} of form (8.8)**Input:** \mathcal{P} , a multivariate quadratic system of m equations in $n \geq \frac{m(m+3)}{2}$ variables**Output:** linear transformations \mathcal{S} and \mathcal{T} , quadratic map \mathcal{F} of the form of (8.8) such that $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$.

```

1:  $\ell = 1$ .
2: while  $\ell \leq m$  do
3:   If necessary, permute the polynomials  $p^{(i)}$  ( $i = 1, \dots, m$ ) such that
     the coefficient  $a_{\ell, \ell}^{(m-\ell+1)}$  of  $x_\ell^2$  in  $p^{(m-\ell+1)}$  is non zero.
4:   For  $i = 1, \dots, m - \ell$  set  $\tilde{p}^{(i)} = p^{(i)} - \frac{a_{\ell, \ell}^{(i)}}{a_{\ell, \ell}^{(m-\ell+1)}} p^{(m-\ell+1)}$  and store
     the coefficients  $\frac{a_{\ell, \ell}^{(i)}}{a_{\ell, \ell}^{(m-\ell+1)}}$  in a matrix  $S_{m-\ell+1}$  of the form (8.11).
5:    $\ell = \ell + 1$ 
6:   Set  $\mathcal{P} = \tilde{\mathcal{P}}$ .
7:   Define a matrix  $T_\ell$  of the form (8.12) and compute  $\hat{\mathcal{P}} = \mathcal{P} \circ T_\ell$ .
8:   for  $i = 1$  to  $m + 1 - \ell$  do
9:     Set the coefficients of the terms  $x_j x_\ell$  in  $\hat{\mathcal{P}}$  to zero for  $1 \leq j < \ell$ .
10:  end for
11:  for  $i = m + 2 - \ell$  to  $m$  do
12:    Set the coefficients of the terms  $x_j x_\ell$  in  $\hat{\mathcal{P}}$  to zero ( $1 \leq j < m - i$ ).
13:  end for
14:  This gives a system of homogeneous linear equations in the unknown
     elements of  $T_\ell$ . Find a solution with  $t_{\ell, \ell} \neq 0$ .
15:  Compute  $\hat{\mathcal{P}} = \mathcal{P} \circ T_\ell$  and then set  $\mathcal{P} = \hat{\mathcal{P}}$ .
16: end while
17: Set  $\mathcal{F} = \mathcal{P}$ 
18: Set  $\mathcal{T} = (T_2 \cdots T_m)^{-1}$ .
19: Set  $\mathcal{S} = (S_{m-1} \cdots S_1)^{-1}$ 
20: return  $\mathcal{S}, \mathcal{F}, \mathcal{T}$ 

```

$\{1, \dots, m - i + 1\}$) using Gaussian elimination. We then substitute this solution into the system \mathcal{F} in order to convert it to the form of (8.10).

Under the assumption that $-\overline{Q}_2^{(1)}$ is a real square in \mathbb{F} , we can derive from the first equation of (8.10) the value of x_m . By substituting this value into the second equation of (8.10), we can hope to find a value for x_{m-1} . We continue this process until we have found a value for x_1 . In order to get a solution \mathbf{y} of $\mathcal{F}(\mathbf{x}) = 0$, we append the values of the variables x_{m+1}, \dots, x_n found by solving the linear equations $L_i^{(j)}$ (see above), i.e.

$$\mathbf{y} = (x_1, \dots, x_m, x_{m+1}, \dots, x_m).$$

Finally, to get a solution \mathbf{z} of the original system $\mathcal{P}(\mathbf{x}) = 0$, we compute $\mathbf{z} = \mathcal{T}^{-1}(\mathbf{y})$.

Our algorithm works for fields of odd and of even characteristic. However, in the process of solving the system \mathcal{F} , we have to take square roots from field elements. Since an element of a field of odd characteristic is a square with probability $1/2$, the success probability of the algorithm is $(1/2)^m$. We therefore have to run the

algorithm 2^m times to find a solution of the original system \mathcal{P} . So, the actual complexity of solving the system \mathcal{P} over a field of odd characteristic is $O(2^m n^\omega m)$.

For fields of even characteristic, we can find the square roots of all field elements, so that the first equation in (8.10) does not cause any difficulty. But further down the line we may have to solve a quadratic equation with a linear term and it can happen that the quadratic equation is irreducible.

It is possible that the decomposition gives $a_{m,m}^{(1)} = 0$. Then most likely $\overline{Q}^{(1)} \neq 0$ and a solution for x_m can not be found. If $\overline{Q}^{(1)} = 0$ holds, there is a chance that the equations of (8.10) lead to a solution. In [20] it is mentioned that in this case a linear equation may be encountered, which allows the process of solving the system to be continued.

The algorithm assumes that a solution of the full system also satisfies the linear system selected in (8.9). This is not guaranteed and it manifests itself when solving (8.10) breaks down. Since the variables were subdivided into the two groups (x_1, \dots, x_m) and (x_{m+1}, \dots, x_n) it is possible to overcome this difficulty by interchanging some of the variables in the two groups. Another possibility is to use a more general linear transformation.

Creating the toy example below indicated to us that there is a substantial chance that the algorithm will fail when the underlying field is small, since the chance that $a_{m,m}^{(1)} = 0$ is fairly high. For larger fields this should be less problematic.

8.7.3 Analysis of the Algorithm

The algorithm works, if and only if all linear systems appearing during the process of transforming and solving the system have a solution. In particular, we require that all systems contain at least as many variables as equations. The linear system we have to solve in the ℓ -th iteration of the loop of Algorithm 8.10 (line 14) contains

$$(m - \ell + 1)(\ell - 1) + \sum_{m+2-\ell}^m (m + 1 - i) = \frac{m(m+1)}{2} - 1$$

equations in the n variables of T_ℓ . Therefore, choosing

$$n \geq \frac{m(m+1)}{2}$$

enables us to find the desired decomposition of \mathcal{P} into $\mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$.

In the process of solving the system \mathcal{F} , we have to solve the linear equations $L_i^{(j)} = 0$ with $(i = 1, \dots, m, j = 1, \dots, m - i + 1)$ in the $n - m$ variables x_{m+1}, \dots, x_n . Altogether, there are

$$\sum_{i=1}^m (m - i + 1) = \frac{m(m+1)}{2}$$

equations. We therefore need

$$n - m \geq \frac{1}{2}m(m+1)$$

or

$$n \geq \frac{m(m+3)}{2}$$

variables in the original quadratic system \mathcal{P} to run the algorithm.

Assuming that the algorithm works on the first try, then the complexity of the algorithm is $O(n^\omega m)$, where $2 < \omega \leq 3$ is the linear algebra constant of solving a linear system.

8.7.4 Toy Example

We want to solve the system $\mathcal{P} = (p^{(1)}, p^{(2)}, p^{(3)})$ in the variables x_1, \dots, x_9 , where

$$\begin{aligned} p^{(1)} = & x_1^2 + \alpha^2 x_1 x_4 + x_1 x_5 + \alpha x_1 x_6 + \alpha x_1 x_9 + x_1 + x_2^2 + \alpha^2 x_2 x_3 + \alpha^2 x_2 x_4 \\ & + x_2 x_5 + x_2 x_6 + \alpha^2 x_2 x_7 + x_2 x_8 + \alpha x_2 x_9 + \alpha x_3^2 + x_3 x_4 + x_3 x_6 + \alpha^2 x_3 x_9 \\ & + \alpha^2 x_4^2 + x_4 x_5 + \alpha x_4 x_6 + x_4 x_9 + x_4 + \alpha^2 x_5^2 + \alpha x_5 x_6 + \alpha x_5 x_7 \\ & + x_5 x_8 + x_5 x_9 + \alpha^2 x_5^2 + \alpha^2 x_6^2 + x_6 x_8 + \alpha x_6 x_9 + \alpha^2 x_7^2 + x_7 x_8 + \alpha^2 x_7 x_9 \\ & + \alpha^2 x_7 + \alpha^2 x_8^2 + \alpha^2 x_8 x_9 + \alpha x_9^2 + \alpha x_9 + \alpha^2, \end{aligned}$$

$$\begin{aligned} p^{(2)} = & \alpha^2 x_1 x_2 + \alpha^2 x_1 x_3 + \alpha x_1 x_5 + x_1 x_6 + \alpha x_1 x_8 + \alpha x_1 x_9 + x_1 + x_2 x_3 \\ & + \alpha x_2 x_4 + \alpha^2 x_2 x_5 + \alpha x_2 x_6 + \alpha x_2 x_7 + x_2 x_8 + \alpha x_2 x_9 + x_2 + \alpha x_3^2 + x_3 x_4 \\ & + \alpha x_3 x_5 + x_3 x_6 + \alpha^2 x_3 x_8 + \alpha^2 x_3 x_9 + \alpha x_3 + x_4^2 + \alpha x_4 x_5 + \alpha x_4 x_6 + x_4 x_7 \\ & + \alpha x_4 x_8 + x_4 x_9 + x_4 + \alpha x_5^2 + \alpha x_5 x_7 + \alpha x_5 x_8 + \alpha x_5 x_9 + x_5 + \alpha x_6^2 \\ & + \alpha^2 x_6 x_7 + \alpha x_6 x_9 + \alpha^2 x_6 + \alpha x_7^2 + x_7 x_8 + \alpha^2 x_7 x_9 + \alpha^2 x_8^2 + \alpha x_8 x_9 \\ & + \alpha^2 x_8 + \alpha x_9^2 + \alpha^2 x_9 + 1, \end{aligned}$$

$$\begin{aligned} p^{(3)} = & x_1^2 + \alpha x_1 x_3 + \alpha^2 x_1 x_4 + \alpha x_1 x_5 + x_1 x_6 + \alpha x_1 + x_2^2 + \alpha x_2 x_3 + \alpha^2 x_2 x_4 \\ & + \alpha^2 x_2 x_6 + \alpha^2 x_2 x_7 + \alpha^2 x_2 x_8 + x_2 x_9 + \alpha x_3^2 + \alpha x_3 x_4 + \alpha^2 x_3 x_5 + \alpha^2 x_3 x_7 \end{aligned}$$

$$\begin{aligned}
& + x_3x_8 + \alpha x_4^2 + \alpha x_4x_7 + x_4x_8 + \alpha^2x_4x_9 + \alpha x_4 + \alpha^2x_5^2 + x_5x_6 + x_5x_7 \\
& + \alpha x_5x_8 + \alpha x_5x_9 + x_5 + \alpha^2x_6^2 + \alpha^2x_6x_8 + \alpha^2x_6x_9 + \alpha x_6 + \alpha x_7^2 + x_7x_8 \\
& + \alpha x_7 + \alpha^2x_8^2 + \alpha x_8x_9 + x_8 + \alpha^2x_9^2 + \alpha.
\end{aligned}$$

8.7.4.1 Finding the Decomposition of \mathcal{P}

Iteration 1a In part 1 of the first iteration we have to set the coefficients of x_1^2 in the polynomials $p^{(1)}$ and $p^{(2)}$ to 0. We have

$$\frac{p_{11}^{(1)}}{p_{11}^{(3)}} = 1 \quad \text{and} \quad \frac{p_{11}^{(2)}}{p_{11}^{(3)}} = 0.$$

Such we get

$$S_3 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

After this transformation, the components $\tilde{p}^{(1)}$, $\tilde{p}^{(2)}$ and $\tilde{p}^{(3)}$ of the map $\tilde{\mathcal{P}} = S_3 \circ \mathcal{P}$ have the form

$$\begin{aligned}
\tilde{p}^{(1)} &= \alpha x_1x_3 + \alpha^2x_1x_5 + \alpha^2x_1x_6 + \alpha x_1x_9 + \alpha^2x_1 + x_2x_3 + x_2x_5 + \alpha x_2x_6 \\
&+ \alpha x_2x_8 + \alpha^2x_2x_9 + \alpha^2x_3x_4 + \alpha^2x_3x_5 + x_3x_6 + \alpha^2x_3x_7 + x_3x_8 \\
&+ \alpha^2x_3x_9 + x_4^2 + x_4x_5 + \alpha x_4x_6 + \alpha x_4x_7 + x_4x_8 + \alpha x_4x_9 + \alpha^2x_4 \\
&+ \alpha^2x_5x_6 + \alpha^2x_5x_7 + \alpha^2x_5x_8 + \alpha^2x_5x_9 + \alpha x_5 + \alpha x_6x_8 + x_6x_9 + \alpha x_6 \\
&+ x_7^2 + \alpha^2x_7x_9 + x_7 + x_8x_9 + x_8 + x_9^2 + \alpha x_9 + 1, \\
\tilde{p}^{(2)} &= \alpha^2x_1x_2 + \alpha^2x_1x_3 + \alpha x_1x_5 + x_1x_6 + \alpha x_1x_8 + \alpha x_1x_9 + x_1 + x_2x_3 \\
&+ \alpha x_2x_4 + \alpha^2x_2x_5 + \alpha x_2x_6 + \alpha x_2x_7 + x_2x_8 + \alpha x_2x_9 + x_2 + \alpha x_3^2 \\
&+ x_3x_4 + \alpha x_3x_5 + x_3x_6 + \alpha^2x_3x_8 + \alpha^2x_3x_9 + \alpha x_3 + x_4^2 + \alpha x_4x_5 \\
&+ \alpha x_4x_6 + x_4x_7 + \alpha x_4x_8 + x_4x_9 + x_4 + \alpha x_5^2 + \alpha x_5x_7 + \alpha x_5x_8 \\
&+ \alpha x_5x_9 + x_5 + \alpha x_6^2 + \alpha^2x_6x_7 + \alpha x_6x_9 + \alpha^2x_6 + \alpha x_7^2 + x_7x_8 + \alpha^2x_7x_9 \\
&+ \alpha^2x_8^2 + \alpha x_8x_9 + \alpha^2x_8 + \alpha x_9^2 + \alpha^2x_9 + 1, \\
\tilde{p}^{(3)} &= x_1^2 + \alpha x_1x_3 + \alpha^2x_1x_4 + \alpha x_1x_5 + x_1x_6 + \alpha x_1 + x_2^2 + \alpha x_2x_3 + \alpha^2x_2x_4 \\
&+ \alpha^2x_2x_6 + \alpha^2x_2x_7 + \alpha^2x_2x_8 + x_2x_9 + \alpha x_3^2 + \alpha x_3x_4 + \alpha^2x_3x_5
\end{aligned}$$

$$\begin{aligned}
& + \alpha^2 x_3 x_7 + x_3 x_8 + \alpha x_4^2 + \alpha x_4 x_7 + x_4 x_8 + \alpha^2 x_4 x_9 + \alpha x_4 + \alpha^2 x_5^2 + x_5 x_6 \\
& + x_5 x_7 + \alpha x_5 x_8 + \alpha x_5 x_9 + x_5 + \alpha^2 x_6^2 + \alpha^2 x_6 x_8 + \alpha^2 x_6 x_9 + \alpha x_6 + \alpha x_7^2 \\
& + x_7 x_8 + \alpha x_7 + \alpha^2 x_8^2 + \alpha x_8 x_9 + x_8 + \alpha^2 x_9^2 + \alpha.
\end{aligned}$$

We set $\mathcal{P} = \tilde{\mathcal{P}}$.

Iteration 1b Next, we need to find a linear transformation T_2 of the variables, which turns the coefficients of $x_1 x_2$ in $p^{(i)}$ to zero ($i = 1, 2, 3$). We set

$$T_2 = \begin{pmatrix} 1 & t_{1,2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & t_{2,2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & t_{3,2} & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & t_{4,2} & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & t_{5,2} & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & t_{6,2} & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & t_{7,2} & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & t_{8,2} & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & t_{9,2} & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

and compute $\hat{\mathcal{P}} = \mathcal{P} \circ T_2$. Setting the coefficients of $x_1 x_2$ to zero leads to three linear homogeneous equations for the $t_{i,2}$ with $i = 1, \dots, 9$. The coefficient matrix for these equations is

$$\begin{pmatrix} 0 & 0 & \alpha & 0 & \alpha^2 & \alpha^2 & 0 & 0 & \alpha \\ 0 & \alpha^2 & \alpha^2 & 0 & \alpha & 1 & 0 & \alpha & \alpha \\ 0 & 0 & \alpha & \alpha^2 & \alpha & 1 & 0 & 0 & 0 \end{pmatrix}.$$

The nullspace of this system is six dimensional, but only three dimensional if we require that $t_{2,2} \neq 0$. We select

$$(t_{1,2}, t_{2,2}, t_{3,2}, t_{4,2}, t_{5,2}, t_{6,2}, t_{7,2}, t_{8,2}, t_{9,2}) = (0, 1, \alpha, \alpha, 1, 0, 0, 0, 0).$$

By substituting this solution into the matrix T_2 and computing $\tilde{\mathcal{P}} = \mathcal{P} \circ T_2$, we get the following system after the first iteration of the transformation process

$$\begin{aligned}
\tilde{p}^{(1)} = & \alpha x_1 x_3 + \alpha^2 x_1 x_5 + \alpha^2 x_1 x_6 + \alpha x_1 x_9 + \alpha^2 x_1 + x_2^2 + \alpha^2 x_2 x_3 + \alpha x_2 x_5 \\
& + x_2 x_7 + x_2 x_8 + \alpha x_2 x_9 + \alpha^2 x_2 + \alpha^2 x_3 x_4 + \alpha^2 x_3 x_5 + x_3 x_6 + \alpha^2 x_3 x_7 \\
& + x_3 x_8 + \alpha^2 x_3 x_9 + x_4^2 + x_4 x_5 + \alpha x_4 x_6 + \alpha x_4 x_7 + x_4 x_8 + \alpha x_4 x_9 + \alpha^2 x_4 \\
& + \alpha^2 x_5 x_6 + \alpha^2 x_5 x_7 + \alpha^2 x_5 x_8 + \alpha^2 x_5 x_9 + \alpha x_5 + \alpha x_6 x_8 + x_6 x_9 + \alpha x_6 \\
& + x_7^2 + \alpha^2 x_7 x_9 + x_7 + x_8 x_9 + x_8 + x_9^2 + \alpha x_9 + 1,
\end{aligned}$$

$$\begin{aligned}
\tilde{p}^{(2)} = & \alpha^2 x_1 x_3 + \alpha x_1 x_5 + x_1 x_6 + \alpha x_1 x_8 + \alpha x_1 x_9 + x_1 + x_2^2 + x_2 x_3 + \alpha x_2 x_4 \\
& + \alpha^2 x_2 x_5 + \alpha^2 x_2 x_6 + \alpha x_2 x_7 + x_2 x_8 + \alpha^2 x_2 x_9 + x_2 + \alpha x_3^2 + x_3 x_4 \\
& + \alpha x_3 x_5 + x_3 x_6 + \alpha^2 x_3 x_8 + \alpha^2 x_3 x_9 + \alpha x_3 + x_4^2 + \alpha x_4 x_5 + \alpha x_4 x_6 \\
& + x_4 x_7 + \alpha x_4 x_8 + x_4 x_9 + x_4 + \alpha x_5^2 + \alpha x_5 x_7 + \alpha x_5 x_8 + \alpha x_5 x_9 + x_5 \\
& + \alpha x_6^2 + \alpha^2 x_6 x_7 + \alpha x_6 x_9 + \alpha^2 x_6 + \alpha x_7^2 + x_7 x_8 + \alpha^2 x_7 x_9 + \alpha^2 x_8^2 \\
& + \alpha x_8 x_9 + \alpha^2 x_8 + \alpha x_9^2 + \alpha^2 x_9 + 1, \\
\tilde{p}^{(3)} = & x_1^2 + \alpha x_1 x_3 + \alpha^2 x_1 x_4 + \alpha x_1 x_5 + x_1 x_6 + \alpha x_1 + \alpha x_2 x_3 + x_2 x_5 + \alpha x_2 x_6 \\
& + x_2 x_8 + \alpha x_2 x_9 + \alpha x_2 + \alpha x_3^2 + \alpha x_3 x_4 + \alpha^2 x_3 x_5 + \alpha^2 x_3 x_7 + x_3 x_8 \\
& + \alpha x_4^2 + \alpha x_4 x_7 + x_4 x_8 + \alpha^2 x_4 x_9 + \alpha x_4 + \alpha^2 x_5^2 + x_5 x_6 + x_5 x_7 + \alpha x_5 x_8 \\
& + \alpha x_5 x_9 + x_5 + \alpha^2 x_6^2 + \alpha^2 x_6 x_8 + \alpha^2 x_6 x_9 + \alpha x_6 + \alpha x_7^2 + x_7 x_8 + \alpha x_7 \\
& + \alpha^2 x_8^2 + \alpha x_8 x_9 + x_8 + \alpha^2 x_9^2 + \alpha
\end{aligned}$$

Iteration 2 Next, we have to find a linear transformation S_2 such that the coefficient of x_2^2 of the first polynomial becomes zero. This is accomplished with the matrix

$$S_2 = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

We then have to turn the coefficients of $x_1 x_3$ and $x_2 x_3$ of the polynomials $p^{(1)}$ and $p^{(2)}$ as well as the coefficient of $x_1 x_3$ of the polynomial $p^{(3)}$ to zero. For this we define a matrix T_3 as specified in (8.12).

We compute $\hat{p}^{(i)} = p^{(i)} \circ T_3$ ($i = 1, \dots, 3$) and obtain the following coefficient matrix for the five homogeneous linear equations

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 1 & \alpha & 0 & \alpha & 0 \\ 0 & 0 & \alpha & \alpha & 1 & \alpha^2 & \alpha^2 & 0 & 1 \\ 0 & 0 & \alpha^2 & 0 & \alpha & 1 & 0 & \alpha & \alpha \\ 0 & 0 & 1 & \alpha & \alpha^2 & \alpha^2 & \alpha & 1 & \alpha^2 \\ 0 & 0 & \alpha & \alpha^2 & \alpha & 1 & 0 & 0 & 0 \end{pmatrix}.$$

The dimension of the nullspace is five, but only one vector meets the requirement that $t_{3,3} \neq 0$:

$$(t_{1,3}, t_{2,3}, t_{3,3}, t_{4,3}, t_{5,3}, t_{6,3}, t_{7,3}, t_{8,3}, t_{9,3}) = (0, 0, \alpha, 0, \alpha, 0, 0, 0, 1).$$

By substituting this solution into the matrix T_3 and computing $\tilde{\mathcal{P}} = \mathcal{P} \circ T_3$, we get a system \mathcal{F} of the form (8.8)

$$\begin{aligned}
 f^{(1)} &= x_1x_5 + \alpha x_1x_6 + \alpha x_1x_8 + \alpha x_1 + \alpha x_2x_4 + x_2x_5 + \alpha^2x_2x_6 + \alpha^2x_2x_7 \\
 &\quad + x_2x_9 + \alpha x_2 + \alpha x_3^2 + x_3x_4 + \alpha^2x_3x_5 + \alpha x_3x_6 + \alpha^2x_3x_7 + \alpha x_3x_8 \\
 &\quad + \alpha x_3x_9 + \alpha^2x_3 + \alpha^2x_4x_5 + \alpha^2x_4x_7 + \alpha^2x_4x_8 + \alpha^2x_4x_9 + \alpha x_4 + \alpha x_5^2 \\
 &\quad + \alpha^2x_5x_6 + x_5x_7 + x_5x_8 + x_5x_9 + \alpha^2x_5 + \alpha x_6^2 + \alpha^2x_6x_7 + \alpha x_6x_8 \\
 &\quad + \alpha^2x_6x_9 + x_6 + \alpha^2x_7^2 + x_7x_8 + x_7 + \alpha^2x_8^2 + \alpha^2x_8x_9 + \alpha x_8 + \alpha^2x_9^2 + x_9, \\
 f^{(2)} &= \alpha x_1x_5 + x_1x_6 + \alpha x_1x_8 + \alpha x_1x_9 + x_1 + x_2^2 + \alpha x_2x_4 + \alpha^2x_2x_5 \\
 &\quad + \alpha^2x_2x_6 + \alpha x_2x_7 + x_2x_8 + \alpha^2x_2x_9 + x_2 + x_3^2 + x_3x_5 + \alpha x_3x_9 + \alpha x_3 \\
 &\quad + x_4^2 + \alpha x_4x_5 + \alpha x_4x_6 + x_4x_7 + \alpha x_4x_8 + x_4x_9 + x_4 + \alpha x_5^2 + \alpha x_5x_7 \\
 &\quad + \alpha x_5x_8 + \alpha x_5x_9 + x_5 + \alpha x_6^2 + \alpha^2x_6x_7 + \alpha x_6x_9 + \alpha^2x_6 + \alpha x_7^2 + x_7x_8 \\
 &\quad + \alpha^2x_7x_9 + \alpha^2x_8^2 + \alpha x_8x_9 + \alpha^2x_8 + \alpha x_9^2 + \alpha^2x_9 + 1, \\
 f^{(3)} &= x_1^2 + \alpha^2x_1x_4 + \alpha x_1x_5 + x_1x_6 + \alpha x_1 + \alpha^2x_2x_3 + x_2x_5 + \alpha x_2x_6 \\
 &\quad + x_2x_8 + \alpha x_2x_9 + \alpha x_2 + x_3^2 + \alpha^2x_3x_5 + x_3x_6 + \alpha^2x_3x_7 + \alpha^2x_3x_8 \\
 &\quad + \alpha^2x_3x_9 + \alpha x_3 + \alpha x_4^2 + \alpha x_4x_7 + x_4x_8 + \alpha^2x_4x_9 + \alpha x_4 + \alpha^2x_5^2 + x_5x_6 \\
 &\quad + x_5x_7 + \alpha x_5x_8 + \alpha x_5x_9 + x_5 + \alpha^2x_6^2 + \alpha^2x_6x_8 + \alpha^2x_6x_9 + \alpha x_6 + \alpha x_7^2 \\
 &\quad + x_7x_8 + \alpha x_7 + \alpha^2x_8^2 + \alpha x_8x_9 + x_8 + \alpha^2x_9^2 + \alpha.
 \end{aligned}$$

The linear transformations \mathcal{S} and \mathcal{T} are given by the matrices

$$S^{-1} = S_2 \circ S_3 = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad T^{-1} = T_2 \circ T_3 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \alpha & \alpha & 0 & 0 & 0 & 0 & 0 \\ 0 & \alpha & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & \alpha & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

8.7.4.2 Solving the System

We want to find a vector $\mathbf{z} \in \mathbb{F}^9$ such that $\mathcal{P}(\mathbf{z}) = 0$. For that we consider the linear maps L_i^j ($i \in \{1, \dots, m\}$, $j \in \{1, \dots, m - i + 1\}$):

$$\begin{aligned} L_1^{(1)} &: x_5 + \alpha x_6 + \alpha x_8 + 0x_9 + \alpha, \\ L_1^{(2)} &: \alpha x_4 + x_5 + \alpha^2 x_6 + \alpha^2 x_7 + x_9 + \alpha, \\ L_1^{(3)} &: x_4 + \alpha^2 x_5 + \alpha x_6 + \alpha^2 x_7 + \alpha x_8 + \alpha x_9 + \alpha^2, \\ L_2^{(1)} &: \alpha x_5 + x_6 + \alpha x_8 + \alpha x_9 + 1, \\ L_2^{(2)} &: \alpha x_4 + \alpha^2 x_5 + \alpha^2 x_6 + \alpha x_7 + x_8 + \alpha^2 x_9 + 1, \\ L_3^{(1)} &: \alpha^2 x_4 + \alpha x_5 + x_6 + \alpha. \end{aligned}$$

By setting these equations to zero and solving for x_4, \dots, x_9 , we get

$$(x_4, \dots, x_9) = (1, \alpha^2, 0, 0, \alpha^2, \alpha^2). \quad (8.14)$$

We substitute these values into the components of \mathcal{F} and obtain

$$\begin{aligned} \tilde{f}^{(1)} &: \alpha x_3^2 + \alpha, \\ \tilde{f}^{(2)} &: x_2^2 + x_3^2 + \alpha^2, \\ \tilde{f}^{(3)} &: x_1^2 + \alpha^2 x_2 x_3 + \alpha^2 x_2 + x_3^2 + 1. \end{aligned}$$

We solve the system $\tilde{\mathcal{F}}(\mathbf{y}) = 0$ from top to bottom. $\tilde{f}^{(1)} = 0$ yields the double root $x_3 = 1$. Substituting this into $\tilde{f}^{(2)}$ yields $x_2^2 + \alpha = 0$, which has the double root $x_2 = \alpha^2$. Substituting all of this into $\tilde{f}^{(3)}$ yields $x_1^2 = 0$ or $x_1 = 0$.

Altogether, we obtain

$$\mathbf{y} = \mathcal{F}^{-1}(0) = (0, \alpha^2, 1, 1, \alpha^2, 0, 0, \alpha^2, \alpha^2)$$

Finally, we invert the second linear map \mathcal{T} to obtain

$$\mathbf{z} = \mathcal{T}^{-1}(\mathbf{y}) = (0, \alpha^2, \alpha^2, 0, \alpha, 0, 0, \alpha^2, \alpha).$$

By substituting \mathbf{z} into the system \mathcal{P} , we can check that it is a correct solution of the equation $\mathcal{P}(\mathbf{z}) = 0$.

If we want to solve $\mathcal{P}(\mathbf{z}) = \mathbf{w}$ for an arbitrary \mathbf{w} , say $\mathbf{w} = (1, \alpha^2, \alpha)$ then we have to invert the first linear map to get

$$\mathbf{x} = \mathcal{S}^{-1}(\mathbf{w}) = (0, \alpha^2, \alpha).$$

Since the linear system (8.14) does not depend on the constant terms of the system \mathcal{P} , we can simply set $\tilde{\mathcal{F}} = \tilde{\mathcal{F}} - \mathbf{x}$ and obtain

$$\begin{aligned}\tilde{f}^{(1)} &: \alpha x_3^2 + \alpha, \\ \tilde{f}^{(2)} &: x_2^2 + x_3^2, \\ \tilde{f}^{(3)} &: x_1^2 + \alpha^2 x_2 x_3 + \alpha^2 x_2 + x_3^2 + \alpha^2.\end{aligned}$$

Solving these equations from top to bottom we get $(x_1, x_2, x_3) = (\alpha^2, 1, 1)$ and with this

$$\mathbf{y} = \mathcal{F}^{-1}(\mathbf{x}) = (\alpha^2, 1, 1, 1, \alpha^2, 0, 0, \alpha^2, \alpha^2).$$

The solution to $\mathcal{P}(\mathbf{z}) = \mathbf{w}$ is then

$$\mathbf{z} = \mathcal{T}^{-1}(\mathbf{y}) = (\alpha^2, 1, 0, \alpha^2, 0, 0, 0, \alpha^2, \alpha),$$

as can be verified directly.

8.7.5 Solving Underdetermined Systems with $n = \nu m$

As we have seen in the previous section, an underdetermined multivariate quadratic system with m equations and $n \geq \frac{m(m+3)}{2}$ variables can be solved in polynomial time. When the number of variables is below this bound, we need exponential time to solve the system.

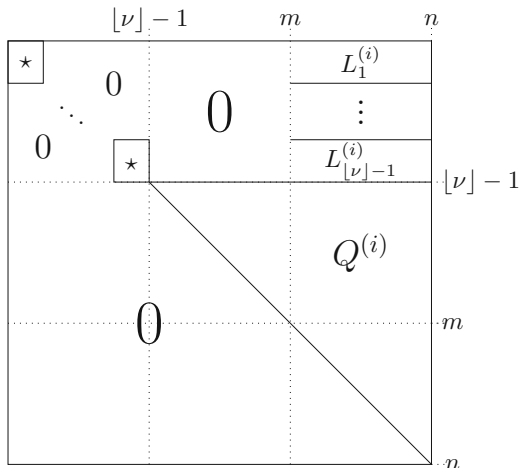
However as proposed in [26], when the number of variables is given by $n = \nu m$ for $\nu \geq 2$, we can use the additional variables to solve the system faster. Hereby we assume that ν is strictly less than m , because otherwise we could solve the system in polynomial time using the techniques presented in the previous section. In particular we get

Theorem 8.50 ([26]) *A multivariate quadratic system of m equations in $n = \nu m$ variables can be solved in about the same time as a multivariate quadratic system of $m - \lfloor \nu \rfloor + 1$ equations in $m - \lfloor \nu \rfloor + 1$ variables.³*

Let \mathcal{P} be a multivariate quadratic system of m equations in $n = \nu m$ variables. In the following we describe how to solve the system $\mathcal{P}(x_1, \dots, x_{\nu m}) = 0$ by solving a determined system $\hat{\mathcal{P}}(y_1, \dots, y_{m-\lfloor \nu \rfloor+1}) = 0$ of $m - \lfloor \nu \rfloor + 1$ quadratic equations and performing a number of linear algebra operations. The process consists of two main steps, which are summarized in Algorithms 8.11 and 8.12.

³The term “about” here means that we do not consider polynomial terms in the complexity (e.g. Gaussian elimination).

Fig. 8.5 Structure of the matrices representing the homogeneous quadratic parts of the polynomials $f^{(1)}, \dots, f^{(m)}$ generated by Algorithm 8.11



1. Find a linear map $\mathcal{T} : \mathbb{F}^n \rightarrow \mathbb{F}^n$ which transforms the system \mathcal{P} into a system $\mathcal{F} = \mathcal{P} \circ \mathcal{T}^{-1}$, where \mathcal{F} is a system of m equations in n variables. The components $f^{(k)}$ ($i = 1, \dots, m$) of the map \mathcal{F} are of the form (8.15) and are illustrated by Fig. 8.5.

$$f^{(k)} = \sum_{i=1}^{\lfloor v \rfloor - 1} a_{ii}^{(k)} x_i^2 + \sum_{i=1}^{\lfloor v \rfloor - 1} x_i L_i^{(k)}(x_m, \dots, x_n) + Q^{(k)}(x_{\lfloor v \rfloor}, \dots, x_n) \quad (8.15)$$

2. Invert recursively the maps \mathcal{F} and \mathcal{T} . This part includes the transformation of the system \mathcal{F} into a determined system $\hat{\mathcal{F}}$ of $m - \lfloor v \rfloor + 1$ quadratic equations and the solution of $\hat{\mathcal{F}}(\mathbf{x}) = 0$ using a Gröbner basis technique.

Algorithm 8.11 Transforming the system \mathcal{P} into a system \mathcal{F} of the form of (8.15)

Input: Multivariate quadratic system \mathcal{P} of m equations in $n = vm$ variables ($v > 2$)

Output: Maps \mathcal{F} and \mathcal{T} such that $\mathcal{F} = \mathcal{P} \circ \mathcal{T}^{-1}$, where \mathcal{T} is a linear map and \mathcal{F} is a quadratic map, whose components are of the form (8.15).

- 1: **for** $\ell = 2$ to m **do**
 - 2: Define an $n \times n$ matrix T_ℓ of the form of (8.12) and compute $\hat{\mathcal{P}} = \mathcal{P} \circ T_\ell$.
 - 3: Set, for $k = \min\{\lfloor v \rfloor - 1, \ell - 1\}$ and $i = 1, \dots, m$ the coefficients $\hat{a}_{1,\ell}^{(i)}, \dots, \hat{a}_{k,\ell}^{(i)}$ to zero. This gives a linear system in the unknown elements of the matrix T_ℓ , which can be solved for $t_{1,\ell}, \dots, t_{n,\ell}$.
 - 4: Compute $\hat{\mathcal{P}} = \mathcal{P} \circ T_\ell$ and then set $\mathcal{P} = \hat{\mathcal{P}}$.
 - 5: **end for**
 - 6: Set $\mathcal{F} = \mathcal{P}$
 - 7: Set $\mathcal{T} = (T_2 \cdots T_m)^{-1}$.
 - 8: **return** \mathcal{F}, \mathcal{T}
-

In order to find a solution of $\mathcal{F}(\mathbf{x}) = 0$, Algorithm 8.12 first solves the linear system given by

$$\begin{aligned} L_1^{(1)}(x_{m+1}, \dots, x_n) &= 0 \\ &\vdots \\ L_{\lfloor v \rfloor - 1}^{(1)}(x_{m+1}, \dots, x_n) &= 0 \\ L_1^{(2)}(x_{m+1}, \dots, x_n) &= 0 \end{aligned} \tag{8.16}$$

$$\begin{aligned} &\vdots \\ L_{\lfloor v \rfloor - 1}^{(m)}(x_{m+1}, \dots, x_n) &= 0 \end{aligned} \tag{8.17}$$

In the case of $n = \nu m$, this system consists of $m(\lfloor \nu \rfloor - 1)$ linear equations in $n - m = m(\nu - 1)$ variables. Therefore, with high probability, this system has a solution.

By substituting this solution into the polynomials $f^{(1)}, \dots, f^{(m)}$, we obtain a system $\tilde{\mathcal{F}}$ of the form shown in Fig. 8.6. Note that the components of the map $\tilde{\mathcal{F}}$ can be written as

$$\tilde{f}^{(i)} = \alpha_1^{(i)} x_1^2 + \dots + \alpha_{\lfloor \nu \rfloor - 1}^{(i)} x_{\lfloor \nu \rfloor - 1}^2 + \tilde{Q}^{(i)}(x_{\lfloor \nu \rfloor}, \dots, x_m), \tag{8.18}$$

where $\tilde{Q}^{(1)}, \dots, \tilde{Q}^{(m)}$ are quadratic functions in the variables $x_{\lfloor \nu \rfloor}, \dots, x_m$. By performing Gaussian elimination we can derive, from the first $\lfloor \nu \rfloor - 1$ components of $\tilde{\mathcal{F}}(\mathbf{x}) = 0$, $\lfloor \nu \rfloor - 1$ equations of the form

$$\begin{aligned} x_1^2 &= \hat{Q}^{(1)}(x_{\lfloor \nu \rfloor}, \dots, x_m) \\ &\vdots \\ x_{\lfloor \nu \rfloor - 1}^2 &= \hat{Q}^{(\lfloor \nu \rfloor - 1)}(x_{\lfloor \nu \rfloor}, \dots, x_m). \end{aligned} \tag{8.19}$$

By substituting these representations of $x_1^2, \dots, x_{\lfloor \nu \rfloor - 1}^2$ into the last $m - (\lfloor \nu \rfloor - 1)$ components of $\tilde{\mathcal{F}}$, we obtain a multivariate quadratic system $\hat{\mathcal{F}}$ of $m - \lfloor \nu \rfloor + 1$ equations in the $m - \lfloor \nu \rfloor + 1$ variables $x_{\lfloor \nu \rfloor}, \dots, x_m$. We solve the equation $\hat{\mathcal{F}}(\mathbf{x}) = 0$ by e.g. a Gröbner basis technique. We substitute the so obtained values of $x_{\lfloor \nu \rfloor}, \dots, x_m$ back into (8.19) to get the values of $x_1^2, \dots, x_{\lfloor \nu \rfloor - 1}^2$. By taking the square roots, we therefore find the values of $x_1, \dots, x_{\lfloor \nu \rfloor - 1}$. Finally, we append the values of x_{m+1}, \dots, x_n found by solving the equations $L_j^{(i)} = 0$ to obtain a solution $\mathbf{y} = (y_1, \dots, y_n)$ of $\mathcal{F}(\mathbf{x}) = 0$.

In order to find a solution \mathbf{z} of $\mathcal{P}(\mathbf{x}) = 0$, we finally set $\mathbf{z} = \mathcal{T}^{-1}(\mathbf{y})$.

Algorithm 8.12 Solving the system \mathcal{F}

Input: Multivariate quadratic system \mathcal{F} of m equations in $n = \nu m$ variables, whose components are of the form of Fig. 8.5.

Output: vector $\mathbf{y} \in \mathbb{F}^n$ such that $\mathcal{F}(\mathbf{y}) = 0$.

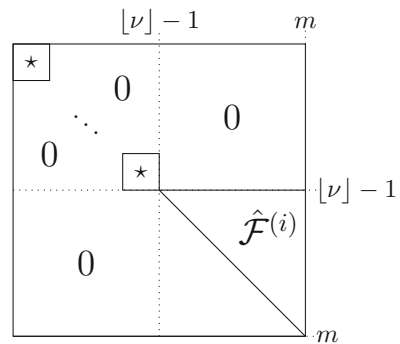
- 1: Set the linear equations $L_1^{(i)}, \dots, L_{\lfloor \nu \rfloor - 1}^{(i)}$ ($i = 1, \dots, m$) of (8.17) to zero and solve the resulting linear system for x_{m+1}, \dots, x_n .
- 2: Substitute the values of x_{m+1}, \dots, x_n into the polynomials $f^{(1)}, \dots, f^{(m)}$. The result is a system $\tilde{\mathcal{F}}$, whose components are of form of Fig. 8.6.
- 3: Perform Gaussian elimination on the polynomials $\tilde{f}^{(1)}, \dots, \tilde{f}^{(\lfloor \nu \rfloor - 1)}$ to find equations $x_1^2 = \hat{Q}^{(1)}(x_{\lfloor \nu \rfloor}, \dots, x_m), \dots, x_{\lfloor \nu \rfloor - 1}^2 = \hat{Q}^{(\lfloor \nu \rfloor - 1)}(x_{\lfloor \nu \rfloor}, \dots, x_m)$.
- 4: Substitute the representations of $x_1^2, \dots, x_{\lfloor \nu \rfloor - 1}^2$ found in the previous step into the polynomials $\tilde{f}^{(\lfloor \nu \rfloor)}, \dots, \tilde{f}^{(m)}$. The result is a determined multivariate quadratic system $\hat{\mathcal{F}}$ of $m - \lfloor \nu \rfloor + 1$ equations (corresponding to the lower right part of Fig. 8.6).
- 5: Solve the system $\hat{\mathcal{F}}(x_{\lfloor \nu \rfloor}, \dots, x_m) = 0$ using XL or a Gröbner basis method.
- 6: Substitute the values of $x_{\lfloor \nu \rfloor}, \dots, x_m$ into the quadratic equations found in step 4 in order to find the values of $x_1^2, \dots, x_{\lfloor \nu \rfloor - 1}^2$ and derive from this the values of $x_1, \dots, x_{\lfloor \nu \rfloor - 1}$ by taking the square roots. (If \mathbb{F} is of odd characteristic and a square root does not exist, return to step 3 and change your selection.)
- 7: Set $\mathbf{y} = (x_1, \dots, x_{\lfloor \nu \rfloor - 1}, x_{\lfloor \nu \rfloor}, \dots, x_m, x_{m+1}, \dots, x_n)$. Here, x_{m+1}, \dots, x_n are the solutions of the linear system of step 1.
- 8: **return** \mathbf{y} .

8.7.6 Toy Example

For our toy example we choose $q = 4$, $m = 3$ and $n = 2m = 6$. We want to solve the system $\mathcal{P}(\mathbf{x}) = (0)$ with

$$\begin{aligned}
 p^{(1)} = & \alpha^2 x_1^2 + \alpha^2 x_1 x_2 + \alpha^2 x_1 x_4 + \alpha^2 x_1 + \alpha x_2^2 + x_2 x_3 + \alpha^2 x_2 x_4 + x_2 x_5 \\
 & + x_2 x_6 + \alpha^2 x_2 + \alpha x_3^2 + \alpha x_3 x_4 + \alpha x_3 x_5 + x_3 + \alpha x_4^2 + \alpha^2 x_4 x_5 + \alpha^2 x_4 x_6 \\
 & + \alpha x_4 + \alpha^2 x_5^2 + \alpha x_5 x_6 + \alpha^2 x_5 + \alpha x_6^2 + x_6 + 1,
 \end{aligned}$$

Fig. 8.6 Structure of the matrices representing the homogeneous quadratic parts of the polynomials $\tilde{f}^{(1)}, \dots, \tilde{f}^{(m)}$ (after step 2 of Algorithm 8.12). Note that the values of the variables x_{m+1}, \dots, x_m have already been determined. Therefore, $\tilde{\mathcal{F}}$ is a system in the m variables x_1, \dots, x_m



$$\begin{aligned}
p^{(2)} &= \alpha^2 x_1 x_2 + \alpha^2 x_1 x_3 + x_1 x_5 + x_1 x_6 + \alpha x_1 + \alpha^2 x_2^2 + \alpha x_2 x_3 + x_2 x_4 + x_2 x_5 \\
&\quad + x_2 + \alpha^2 x_3 x_4 + \alpha^2 x_3 x_6 + \alpha^2 x_3 + \alpha^2 x_4^2 + \alpha x_4 x_6 + x_5^2 + x_5 x_6 + \alpha^2 x_5 + \alpha x_6^2, \\
p^{(3)} &= \alpha^2 x_1^2 + \alpha x_1 x_3 + \alpha x_1 x_4 + x_1 x_5 + \alpha x_1 x_6 + x_1 + x_2^2 + x_2 x_3 + \alpha^2 x_2 x_4 \\
&\quad + \alpha^2 x_2 x_5 + \alpha x_2 x_6 + x_2 + x_3^2 + x_3 x_4 + \alpha x_3 x_5 + \alpha x_3 x_6 + \alpha x_3 + \alpha x_4 x_5 \\
&\quad + \alpha^2 x_4 x_6 + \alpha^2 x_5^2 + x_5 x_6 + \alpha^2 x_6 + \alpha^2.
\end{aligned}$$

8.7.6.1 Step 1: Finding the Decomposition of \mathcal{P}

We define a matrix T_2 of the form

$$T_2 = \begin{pmatrix} 1 & t_{1,2} & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & t_{3,2} & 1 & 0 & 0 & 0 \\ 0 & t_{4,2} & 0 & 1 & 0 & 0 \\ 0 & t_{5,2} & 0 & 0 & 1 & 0 \\ 0 & t_{6,2} & 0 & 0 & 0 & 1 \end{pmatrix}.$$

and compute the map

$$\hat{\mathcal{P}} = \mathcal{P} \circ T_2.$$

By setting the coefficient of $x_1 x_2$ in the polynomials $\hat{p}^{(1)}$, $\hat{p}^{(2)}$ and $\hat{p}^{(3)}$ to zero, we obtain the linear equations

$$\begin{aligned}
\alpha^2 t_{4,2} + \alpha^2 &= 0, \\
\alpha^2 t_{3,2} + x_{5,2} + t_{6,2} + \alpha^2 &= 0, \\
\alpha t_{3,2} + \alpha t_{4,2} + t_{5,2} + \alpha t_{6,2} &= 0.
\end{aligned}$$

Solving these equations by Gaussian elimination yields

$$(t_{1,2}, t_{3,2}, \dots, t_{6,2}) = (0, 1, 1, 0, 0).$$

We substitute this solution into the matrix T_2 and compute, for $i = 1, \dots, 3$,

$$\tilde{\mathcal{P}}^{(i)} = \mathcal{P} \circ T_2.$$

After this, we set $\mathcal{P} = \tilde{\mathcal{P}}$. In the second step we define a matrix T_3 of the form

$$T_3 = \begin{pmatrix} 1 & 0 & t_{1,3} & 0 & 0 & 0 \\ 0 & 1 & t_{2,3} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & t_{4,3} & 1 & 0 & 0 \\ 0 & 0 & t_{5,3} & 0 & 1 & 0 \\ 0 & 0 & t_{6,3} & 0 & 0 & 1 \end{pmatrix}.$$

and compute

$$\hat{\mathcal{P}} = \mathcal{P} \circ T_3.$$

By setting the coefficient of x_1x_3 in the polynomials $\hat{p}^{(1)}$, $\hat{p}^{(2)}$ and $\hat{p}^{(3)}$ to zero, we obtain the three linear equations

$$\begin{aligned} \alpha^2 t_{4,3} &= 0, \\ t_{5,3} + t_{6,3} + \alpha^2 &= 0, \\ \alpha t_{4,3} + t_{5,3} + \alpha t_{6,3} + \alpha &= 0. \end{aligned}$$

Solving these equations by Gaussian elimination yields

$$(t_{1,3}, t_{2,3}, t_{4,3}, t_{5,3}, t_{6,3}) = (0, 0, 0, 1, \alpha).$$

We substitute this solution into the matrix T_3 and compute the linear transformation T by $T = (T_2 T_3)^{-1}$.

We obtain the transformed system $\mathcal{F} = (f^{(1)}, f^{(2)}, f^{(3)})$ by $\mathcal{F} = \mathcal{P} \circ \mathcal{T}^{-1}$. We find

$$\begin{aligned} f^{(1)} &= \alpha^2 x_1^2 + \alpha^2 x_1 x_4 + \alpha^2 x_1 + \alpha x_2^2 + x_2 x_4 + \alpha x_2 x_6 + x_3^2 + x_3 x_5 + \alpha x_3 x_6 \\ &\quad + \alpha x_4^2 + \alpha^2 x_4 x_5 + \alpha^2 x_4 x_6 + \alpha x_4 + \alpha^2 x_5^2 + \alpha x_5 x_6 + \alpha^2 x_5 + \alpha x_6^2 + x_6 + 1, \\ f^{(2)} &= x_1 x_5 + x_1 x_6 + \alpha x_1 + \alpha x_2 x_3 + \alpha x_2 x_4 + x_2 x_5 + x_2 x_6 + \alpha x_2 + \alpha^2 x_3^2 \\ &\quad + \alpha x_3 x_5 + \alpha x_3 x_6 + \alpha^2 x_4^2 + \alpha x_4 x_6 + x_5^2 + x_5 x_6 + \alpha^2 x_5 + \alpha x_6^2, \\ f^{(3)} &= \alpha^2 x_1^2 + \alpha x_1 x_4 + x_1 x_5 + \alpha x_1 x_6 + x_1 + \alpha^2 x_2^2 + \alpha x_2 x_3 + \alpha x_2 x_4 + \alpha^2 x_2 x_5 \\ &\quad + \alpha^2 x_2 x_6 + \alpha^2 x_2 + x_3^2 + \alpha x_3 x_4 + \alpha^2 x_3 x_6 + \alpha^2 x_3 + \alpha x_4 x_5 + \alpha^2 x_4 x_6 \\ &\quad + \alpha^2 x_5^2 + x_5 x_6 + \alpha^2 x_6 + \alpha^2. \end{aligned}$$

8.7.6.2 Step 2: Solving the System

In order to solve the system $\mathcal{F}(\mathbf{x}) = 0$, we first determine the variables x_4, \dots, x_6 in such a way that $L_1^{(i)} = 0$ holds for $i = 1, \dots, 3$. We find

$$\begin{aligned} L_1^{(1)} : \alpha^2 x_4 + \alpha^2 &= 0, \\ L_1^{(2)} : x_5 + x_6 + \alpha &= 0, \\ L_1^{(3)} : \alpha x_4 + x_5 + \alpha x_6 + 1 &= 0. \end{aligned}$$

Gaussian elimination yields $(x_4, x_5, x_6) = (1, 0, \alpha)$. By substituting this into the system \mathcal{F} , we get the system $\tilde{\mathcal{F}}$ with

$$\begin{aligned} \tilde{f}^{(1)} &= \alpha^2 x_1^2 + \alpha x_2^2 + \alpha x_2 + x_3^2 + \alpha^2 x_3 + \alpha^2, \\ \tilde{f}^{(2)} &= \alpha x_2 x_3 + \alpha x_2 + \alpha^2 x_3^2 + \alpha^2 x_3 + 1, \\ \tilde{f}^{(3)} &= \alpha^2 x_1^2 + \alpha^2 x_2^2 + \alpha x_2 x_3 + x_3^2 + \alpha^2. \end{aligned}$$

We remove the term x_1^2 from $\tilde{f}^{(2)}$ and $\tilde{f}^{(3)}$, obtaining

$$\begin{aligned} \hat{f}^{(3)} &= \tilde{f}^{(2)} - 0 \cdot \tilde{f}^{(1)} = \alpha x_2 x_3 + \alpha x_2 + \alpha^2 x_3^2 + \alpha^2 x_3 + 1, \\ \hat{f}^{(3)} &= \tilde{f}^{(3)} - \tilde{f}^{(1)} = x_2^2 + \alpha x_2 x_3 + \alpha x_2 + \alpha^2 x_3. \end{aligned}$$

and solve the multivariate quadratic system given by $\hat{f}^{(2)} = 0$ and $\hat{f}^{(3)} = 0$, obtaining $(x_2, x_3) = (\alpha, \alpha)$. By substituting this into $\tilde{f}^{(1)} = 0$, we get $\alpha^2 x_1^2 + \alpha^2 = 0$ or $x_1 = 1$.

Altogether, we find the solution $\mathbf{y} = (x_1, \dots, x_6) = (1, \alpha, \alpha, 1, 0, \alpha)$ of the system $\mathcal{F}(\mathbf{x}) = 0$.

Finally, in order to find a solution \mathbf{z} of the original system $\mathcal{P}(\mathbf{x}) = 0$, we compute

$$\mathbf{z} = \mathcal{T}(\mathbf{y}) = (1, \alpha, 0, \alpha^2, \alpha, 1).$$

By substituting \mathbf{z} into \mathcal{P} , one can easily check that \mathbf{z} is indeed a solution of $\mathcal{P}(\mathbf{x}) = 0$.

References

1. E. Bach, J. Shallit, *Algebraic Number Theory* (MIT Press, Cambridge, 1996)
2. M. Bardet, Study of overdetermined algebraic systems. Applications to correcting codes and cryptography. PhD Thesis, Paris VI University, 2004

3. M. Bardet, J.-C. Faugère, B. Salvy, On the degree of regularity of gröbner basis computation of semi regular overdetermined algebraic equations, in *International Conference on Polynomial System Solving 2004* (2004), pp. 71–75
4. E.R. Berlekamp, Factoring polynomials over finite fields. *Bell Syst. Tech. J.* **46**, 1853–1859 (1967)
5. L. Bettale, J.-C. Faugère, L. Perret, Hybrid approach for solving multivariate systems over finite fields. *J. Math. Cryptol.* **3**, 177–197 (2009)
6. B. Buchberger, Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal. PhD Thesis, Innsbruck, 1965
7. G. Cantor, H. Zassenhaus, A new algorithm for factoring polynomials over finite fields. *Math. Computat.* **36**(154), 587–592 (1981)
8. N. Courtois, A. Klimov, J. Patarin, A. Shamir, Efficient algorithms for solving overdefined systems of multivariate polynomial equations, in *Advances in Cryptology — EUROCRYPT 2000*. Lecture Notes in Computer Science, vol. 1807 (2000), pp. 392–407
9. C. Diem, The XL-algorithm and a conjecture from commutative algebra, in *Advances in Cryptology — ASIACRYPT 2004*. Lecture Notes in Computer Science, vol. 3329 (Springer, Berlin, 2004), pp. 323–337
10. J. Ding, T.J. Hodges, Inverting HFE systems is quasi-polynomial for all fields, in *Advances in Cryptology — CRYPTO 2011*. Lecture Notes in Computer Science, vol. 6841 (Springer, Berlin, 2011), pp. 724–742
11. J. Ding, T. Kleinjung, Degree of regularity for HFE-. *J. Math. Ind.* **4**, 97–104 (2012)
12. J. Ding, B. Yang, Degree of regularity for HFEv and HFEv-, in *Post-Quantum Cryptography. PQCrypto 2013*. Lecture Notes in Computer Science, vol. 7932 (2013), pp. 52–66
13. J. Ding, T.J. Hodges, V. Kruglov, Growth of the ideal generated by a quadratic boolean function, in *Post-Quantum Cryptography. PQCrypto 2010*. Lecture Notes in Computer Science, vol. 6061 (2010), pp. 13–27
14. J. Ding, T.J. Hodges, V. Kruglov, D.S. Schmidt, S. Tohaneanu, Growth of the ideal generated by a multivariate quadratic function over GF(3). *J. Algebra Appl.* **12**(05), 1–23 (2013)
15. V. Dubois, N. Gama, The degree of regularity of HFE systems, in *Advances in Cryptology — ASIACRYPT 2010*. Lecture Notes in Computer Science, vol. 6477 (Springer, Berlin, 2010), pp. 557–576
16. J.-C. Faugère, A new efficient algorithm for computing gröbner bases (F4). *J. Pure Appl. Algebra* **139**, 61–88 (1999)
17. J.-C. 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 2002* (ACM Press, New York, 2002), pp. 75–83
18. L. Granboulan, A. Joux, J. Stern, Inverting HFE is quasipolynomial, in *Advances in Cryptology — CRYPTO 2006*. Lecture Notes in Computer Science, vol. 4711 (2006), pp. 345–356
19. T. J. Hodges, S. D. Molina, J. Schlather, On the existence of homogeneous semi-regular sequences. *J. Algebra* **476**, 519–547 (2017)
20. H. Miura, Y. Hashimoto, T. Takagi, Extended algorithm for solving underdefined multivariate quadratic equations, in *International Workshop on Post-Quantum Cryptography. PQCrypto 2013*. Lecture Notes in Computer Science, vol. 7932 (Springer, Berlin, 2013), pp. 118–135
21. National Institute of Standards and Technology, Post-quantum cryptography (2018). <https://csrc.nist.gov/projects/post-quantum-cryptography/>
22. D. Nesselmann, *Gröbnerbasen und Nichtlineare Gleichungssysteme*. Vorlesungsskript (2009)
23. J. Patarin, Cryptanalysis of the Matsumoto and Imai public key scheme of Eurocrypt’88, in *Advances in Cryptology — CRYPTO 1995*. Lecture Notes in Computer Science, vol. 963 (Springer, Berlin, 1995), pp. 248–261
24. A. Petzoldt, M.-S. Chen, B.-Y. Yang, T. Chengdong, J. Ding, Design principles for HFEv-based signature schemes, in *Advances in ASIACRYPT 2015 - Part I*. Lecture Notes in Computer Science, vol. 9452 (Springer, Berlin, 2015), pp. 1–24

25. V. Shoup, Factoring polynomials over finite fields: Asymptotic complexity vs. reality, in *IMACS Symposium, Lille* (1993)
26. E. Thomae, C. Wolf, Solving underdetermined systems of multivariate quadratic equations revisited, in *Public Key Cryptography 2012*. Lecture Notes in Computer Science, vol. 7293 (Springer, Berlin, 2012), pp. 156–171
27. B. Yang, J. Chen, Theoretical analysis of XL over small fields, in *Australasian Conference on Information Security and Privacy. ACISP 2004*. Lecture Notes in Computer Science, vol. 3108 (2004), pp. 277–288

Software

For the multivariate schemes described in this book we developed implementations in MAGMA code, which were used to compute the toy examples printed in this book. The corresponding source files can be found on the website <http://dx.doi.org/10.7945/5sqr-g734>.

In particular, you can find there

- Chapter 3 Matsumoto-Imai
 - Key generation, encryption and decryption of the standard MI scheme
 - Linearization Equations attack
 - Key generation, encryption and decryption of the PMI and the PMI+ encryption schemes
 - Key generation, signature generation and verification of the (Projected) SFLASH signature scheme
- Chapter 4 Hidden Field Equations
 - Key generation, encryption and decryption of the standard HFE scheme
 - Key generation, signature generation and verification of the HFEv-signature scheme
- Chapter 5 Oil and Vinegar
 - Key generation, signature generation and verification of the (U)OV signature scheme
 - Kipnis-Shamir attack against OV (odd and even case)
 - Key generation, signature generation and verification of the Rainbow signature scheme
- Chapter 6 MQDSS
 - 3-pass MQ based identification scheme
 - 5-pass MQ based identification scheme

- Chapter 7 SimpleMatrix
 - Key generation, encryption and decryption of the standard Simple Matrix encryption scheme
 - Key generation, encryption and decryption of the rectangular SimpleMatrix encryption scheme
- Chapter 8 Solving Polynomial Systems
 - Univariate polynomials: Berlekamp's and Cantor-Zassenhaus algorithms
 - Multivariate polynomials: Relinearization technique for solving overdetermined systems, Solving underdetermined systems with $n \geq \frac{m(m+3)}{2}$ and $n = vm$.

Furthermore, you can find on this website corrections to this book. You are welcomed to submit your findings per e-mail to any of the authors (addresses in the preface).

Index

A

Algebraic cryptanalysis, 19
Attack
 differential, 43, 53
 direct, 21, 67, 82, 111, 121, 180
 linearization equations, 32
 Oil-Vinegar, 94
 Rainbow Band Separation (RBS), 125
 rank, 67, 76, 83, 122, 181
 structural, 21
 UOV Reconciliation, 113

B

Berlekamp's algorithm, 191
BigField, 25
Bipolar construction, 14
Buchberger's algorithm, 204

C

Cantor–Zassenhaus algorithm, 194
Construction
 bipolar, 14
 mixed systems, 16

D

Degree of regularity, 211, 214, 217
Differential, 13, 43, 53
Differential attack, 43, 53
Digital signature, 3
Direct attack, 21, 67, 82, 111, 121, 180

E

EIP Problem, 20
Encryption scheme
 IPHFE+, 72
 PMI+, 45
 Rect. SimpleMatrix, 174
 SimpleMatrix, 170
 ZHFE, 73
Equivalent key, 112

F

F_4 -algorithm, 207
Fiat-Shamir transformation, 162
Frobenius map, 28

G

Galois group, 28
Gröbner basis, 202

H

Hamming weight, 25, 28
HFEv- signature scheme, 77
Hidden Field Equations (HFE), 62
Higher order linearization equations (HOLE),
 32

I

Identification scheme, 17
 IP based, 17
 MQ based, 19, 153

IP based identification, 17
 IP Problem, 20
 IPHFE+ encryption scheme, 72

L

Linearization equation, 32
 Linearization equations attack, 32
 LUOV signature scheme, 138

M

Macaulay matrix, 10
 Matsumoto-Imai, 26
 Minimal invariant subspace, 97
 Min-Q-rank, 68
 MinRank Problem, 70, 122
 Minus modification, 50, 76
 Mixed systems, 16
 Monomial order, 9
 MQ based identification, 19, 153
 MQ Problem, 19
 MQDSS signature scheme, 164
 Multivariate polynomial ring, 7

O

Oil and Vinegar polynomial, 90
 Oil-Vinegar attack, 94

P

PFLASH signature scheme, 56
 Plus modification, 45
 PMI+ encryption scheme, 45
 PoSSo problem, 19
 Post-quantum cryptography (PQCrypto), 4
 Problem

- PoSSo, 19
- EIP, 20
- IP, 20
- MQ, 19

 Projection, 56
 Public key cryptography (PKC), 2

Q

Q-rank, 68

R

Rainbow Band Separation (RBS) attack, 125
 Rainbow layers, 116
 Rainbow signature scheme, 116
 Random oracle, 163
 Rank attack, 67, 76, 83, 122, 181
 Rectangular SimpleMatrix encryption scheme, 174

S

Semi-regular sequence, 213
 SFLASH signature scheme, 50
 Shor's algorithm, 4
 Signature scheme

- PFLASH, 56
- HFEv-, 77
- LUOV, 138
- MQDSS, 164
- Rainbow, 116
- SFLASH, 50
- UOV, 109

 SimpleMatrix encryption scheme, 170
 Structural attacks, 21
 StructuredRainbow, 134
 StructuredUOV, 130
 Symmetric cryptography, 2

U

UOV Reconciliation attack, 113
 UOV signature scheme, 109

V

Vinegar modification, 76

X

XL-Algorithm, 198

Z

Zero-knowledge proof, 154
 ZHFE encryption scheme, 73

List of Toy Examples

B

Berlekamp's algorithm, 192

C

Cantor–Zassenhaus algorithm, 197

F

F_4 -algorithm, 208

H

HFE encryption scheme, 65

HFEv- signature scheme, 79

L

Linearization equations attack on MI, 38

M

Matsumoto-Imai encryption scheme, 30

MQ based identification scheme, 160

O

Oil and Vinegar signature scheme, 92

Oil-Vinegar attack (even case), 104

Oil-Vinegar attack (odd case), 99

P

PFLASH signature scheme, 57

PMI encryption scheme, 46

R

Rainbow signature scheme, 119

Rectangular SimpleMatrix encryption scheme,
177

S

SFLASH signature scheme, 51

SimpleMatrix encryption scheme, 171

Solving underdetermined systems with
 $n \geq \frac{m(m+3)}{2}$, 234

Solving underdetermined systems with
 $n = vm$, 243

X

XL-Algorithm, 199