Selecting and Reducing Key Sizes for Multivariate Cryptography

Vom Fachbereich Informatik der
Technischen Universität Darmstadt genehmigte

Dissertation

zur Erlangung des Grades
Doktor rerum naturalium (Dr. rer. nat.)
von

Dipl.-Math. Albrecht Petzoldt

geboren in Roth.

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Referenten:	Prof. Dr. Johannes Buchmann Prof. Jintai Ding PhD
Tag der Einreichung:	29.05.2013
Tag der mündlichen Prüfung:	15.07.2013
Hochschulkennziffer:	D 17

Darmstadt 2013

Wissenschaftlicher Werdegang

August 2010 - Juli 2013

Promotions-Stipendium der Horst Görtz Stiftung

März 2009 - Juli 2013

Promotionsstudent an der Technischen Universität Darmstadt

Oktober 2003 - Juli 2008

Studium der Mathematik (Diplom) an der Friedrich-Alexander-Universität Erlangen-Nürnberg

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit - abgesehen von den in ihr ausdrücklich genannten Hilfen - selbständig verfasst habe.

Darmstadt, 29.05.2013 _____

Acknowledgements

I want to thank my supervisor Professor Johannes Buchmann for his advice and support and the second referee of my thesis, Professor Jintai Ding. Special thank goes to Dr. Stanislav Bulygin who is coauthor of most of my papers. Besides the excellent collaboration on the papers he contributed many new ideas for my research and helped me with numerous comments to improve this thesis.

I want to thank all the members of the CDC research group in Darmstadt, especially Dr. Michael Schneider and Patrick Weiden who helped me to solve technical problems. In this context I furthermore want to thank Marita Skrobic who helped me a lot with organisatorial stuff and Ishtiaq Shah who did much of the implementation work presented in this thesis.

Last but not least I want to thank the Horst Görtz Foundation for their financial support which enabled me to do my research without monetary needs.

List of Publications

[P1] X. Nie, A. Petzoldt and J. Buchmann. Cryptanalysis of 2-layer nonlinear Piece in Hand method. In *2nd Workshop on Modern Cryptography and Security Engineering*, *Lecture Notes in Computer Science*, to appear.

[P2] A. Petzoldt. Hybrid Approach for the Fast Verification for Improved Versions of the UOV and Rainbow Signature Schemes. Available at http://eprint.iacr.org/2013/315.

[P3] A. Petzoldt. Speeding up QUAD. Available at http://eprint.iacr.org/2013/263.

[P4] A. Petzoldt, S. Bulygin and J. Buchmann. A Multivariate Threshold Ring Signature Scheme. In *Special Issue of AAECC*, to appear.

[P5] A. Petzoldt, S. Bulygin and J. Buchmann. Fast Verification for Improved Versions of the UOV and Rainbow Signature Schemes. In *PQCrypto*, volume 7932 of *Lecture Notes in Computer Science*, pages 188-202. Springer, 2013.

[P6] A. Petzoldt and S. Bulygin. Linear Recurring Sequences for the UOV Key Generatrion Revisited. In *ICISC*, volume 7839 of *Lecture Notes in Computer Science*, pages 441 – 455. Springer, 2013.

[P7] A. Hülsing, A. Petzoldt, M. Schneider and S. M. E. Y. Alaoui. Postquantum Signaturverfahren heute. In *22. SIT Smartcard Workshop*. Fraunhofer Verlag, 2012.

[P8] F. Borges, A. Petzoldt and R. Portugal. Small Private Keys for Systems of Multivariate Quadratic Equations using Symmetric Cryptography. In *CNMAC - National Congress of the Brazilian Society of Applied and Computational mathematics*, 2012.

[P9] A. Petzoldt, E. Thomae, S. Bulygin and C. Wolf. Small Public Keys and Fast Verification for Multivariate Quadratic Public Key Systems. In *CHES*, volume 6917 of *Lecture Notes in Computer Science*, pages 475 – 490. Springer, 2011.

[P10] A. Petzoldt, S. Bulygin and J. Buchmann. Linear Recurring Sequences for the UOV Key Generation. In *PKC*, volume 6571 of *Lecture Notes in Computer Science*, pages 335–350. Springer, 2011.

[P11] A. Petzoldt, S. Bulygin and J. Buchmann. Cyclicrainbow - a Multivariate Signature Scheme with a Partially Cyclic Public Key. In *INDOCRYPT*, volume 6498 of *Lecture Notes in Computer Science*, pages 33–48. Springer, 2010.

[P12] S. Bulygin, A. Petzoldt and J. Buchmann. Towards Provable Security of the UOV Signature Scheme under Direct Attacks. In *INDOCRYPT*, volume 6498 of *Lecture Notes in Computer Science*, pages 17–32. Springer, 2010.

[P13] A. Petzoldt, S. Bulygin and J. Buchmann. A Multivariate Signature Scheme with a Partially Cyclic Public Key. In *SCC*, pages 229–235, 2010.

[P14] A. Petzoldt, S. Bulygin and J. Buchmann. Selecting Parameters for the Rainbow Signature Scheme - Extended Version. Available at http://eprint.iacr.org/2010/437

[P15] A. Petzoldt, S. Bulygin and J. Buchmann. Selecting Parameters for the Rainbow Signature Scheme. In *PQCrypto*, volume 6061 of *Lecture Notes in Computer Science*, pages 218–240. Springer, 2010.

Abstract

Cryptographic techniques are essential for the security of communication in modern society. As more and more business processes are performed via the Internet, the need for efficient cryptographic solutions will further increase in the future. Today, nearly all cryptographic schemes used in practice are based on the two problems of factoring large integers and solving discrete logarithms. However, schemes based on these problems will become insecure when large enough quantum computers are built. The reason for this is Shor's algorithm, which solves number theoretic problems such as integer factorization and discrete logarithms in polynomial time on a quantum computer. Therefore one needs alternatives to those classical public key schemes. Besides lattice, code and hash based cryptosystems, multivariate cryptography seems to be a candidate for this. Additional to their (believed) resistance against quantum computer attacks, multivariate schemes are very fast and require only modest computational resources, which makes them attractive for the use on low cost devices such as RFID chips and smart cards. However, there remain some open problems to be solved, such as the unclear parameter choice of multivariate schemes, the large key sizes and the lack of more advanced multivariate schemes like signatures with special properties and key exchange protocols.

In this dissertation we address two of these open questions in the area of multivariate cryptography. In the first part we consider the question of the parameter choice of multivariate schemes. We start with the security model of Lenstra and Verheul, which, on the basis of certain assumptions like the development of the computing environment and the budget of an attacker, proposes security levels for now and the near future. Based on this model we study the known attacks against multivariate schemes in general and the Rainbow signature scheme in particular and use this analysis to propose secure parameter sets for these schemes for the years $2012 - 2050$.

In the second part of this dissertation we present an approach to reduce the public key size of certain multivariate signature schemes such as UOV and Rainbow. We achieve the reduction by inserting a structured matrix into the coefficient matrix of the public key, which enables us to store the public key in an efficient way. We propose several improved versions of UOV and Rainbow which reduce the size of the public key by factors of 8 and 3 respectively. Using the results of the first part, we show that using structured public keys does not weaken the security of the underlying schemes against known attacks. Furthermore we show how the structure of the public key can be used to speed up the verification process of the schemes. Hereby we get a speed up of factors of 6 for UOV and 2 for Rainbow. Finally we show how to apply our techniques to the QUAD stream cipher. By doing so we can increase the data throughput of QUAD by a factor of 7.

Zusammenfassung

Kryptographische Techniken sind für die Sicherheit der Kommunikation in der modernen Gesellschaft unverzichtbar. Da der Anteil der im Internet durchgeführten Geschäftsprozesse weiter zunimmt, wird der Bedarf nach effizienten Lösungen auf diesem Gebiet noch weiter steigen. Nahezu alle heutzutage benutzten kryptographischen Verfahren beruhen entweder auf dem Problem des Faktorisierens großer Zahlen oder dem Lösen diskreter Logarithmen. Jedoch werden derartige Verfahren unsicher, sobald Quantencomputer entsprechender Größe zur Verfügung stehen. Der Grund dafür ist Shor's Algorithmus, mit dessen Hilfe zahlentheoretische Probleme wie das Faktorisieren ganzer Zahlen und das Lösen diskreter Logarithmen auf einem Quantencomputer in polynomieller Zeit durchgeführt werden können. Aufgrund dessen werden Alternativen zu diesen klassischen public-key-Verfahren benötigt. Neben gitter-, code- und hashbasierten Verfahren ist die multivariate Kryptographie ein möglicher Kandidat dafür. Zusätzlich zu der (vermuteten) Resistenz gegenüber Angriffen mit Quantencomputern sind multivariate Verfahren sehr schnell und benötigen lediglich moderate Rechenkapazitäten, was diese Verfahren für den Einsatz auf RFID chips und smart cards attraktiv macht. Jedoch bleiben noch einige offene Probleme zu lösen, wie die Parameterwahl multivariater Verfahren, das Problem großer Schlüssel sowie der Mangel an fortgeschrittenen multivariaten Verfahren wie Signaturverfahren mit speziellen Eigenschaften und key-exchange Protokollen.

In dieser Dissertation befassen wir uns mit zwei dieser offenen Fragen auf dem Gebiet der multivariaten Kryptographie. Im ersten Teil beschäftigen wir uns mit der Parameterwahl multivariater Verfahren. Wir beginnen mit dem Sicherheitsmodell von Lenstra und Verheul, das, auf der Basis einiger Annahmen wie der Entwicklung von Rechenkapazitäten und dem Budget eines Angreifers, Sicherheitslevel für die Gegenwart und die nahe Zukunft vorschlägt. Von diesem Modell ausgehend analysieren wir bekannte Angriffe gegen multivariate Systeme im Allgemeinen und das Rainbow Signaturverfahren im Besonderen, um für diese Verfahren sichere Parametersätze für die Zeit bis 2050 herzuleiten.

Im zweiten Teil der Dissertation stellen wir eine Strategie zur Verkleinerung des öffentlichen Schlüssels bestimmter multivariater Signaturverfahren wie UOV und Rainbow vor. Wir erzielen unsere Ergebnisse, indem wir eine strukturierte Matrix in die Koeffizientenmatrix des öffentlichen Schlüssels übertragen. Dies ermöglicht es uns, den öffentlichen Schlüssel auf effiziente Weise zu speichern. Wir beschreiben mehrere dieser verbesserten Varianten von UOV und Rainbow, die die Größe des öffentlichen Schlüssels um das 8-fache (UOV) bzw. das 3-fache (Rainbow) verringern. Darüber hinaus zeigen wir, wie sich die Struktur des öffentlichen Schlüssels dazu verwenden lässt, den Verifikationsprozess der beiden Signaturverfahren zu beschleunigen. Hierbei erzielen wir Beschleunigungen um das 6-fache (UOV) bzw. das 2-fache (Rainbow). Mit Hilfe der Ergebnisse des ersten Teils zeigen wir, dass durch unsere Maßnahmen die Sicherheit der Verfahren nicht beeinträchtigt wird. Im letzten Abschnitt zeigen wir, wie sich unsere Techniken auf die multivariate Stromchiffre QUAD anwenden lassen. Dadurch lässt sich der Datendurchsatz von QUAD um das 7-fache erhöhen.

Contents

Chapter 1

Introduction

Cryptographic techniques are essential for the security of communication in modern society. In the business world, the communication between trading partners needs to remain confidential. But also the private user deals with cryptography nearly every day. Examples for this are online shopping and software downloads. When logging in to an email account or visiting the website of a bank, cryptographic techniques are used, too. As more and more business processes are performed via the Internet (e.g. via cloud computing) and because of new applications like e-voting and digital payment the need for efficient cryptographic solutions will still increase in the future.
The most often used cryptographic primitives are encryption and digital signature schemes. Encryption schemes guarantee the *confidentiality* of data which means that an attacker can not get any information about the content of the encrypted message. Signature schemes on the other hand ensure that a message really comes from the sender (*authentication*) and that it was not changed after the signing process (*data integrity*). For contracts it is also important that none of the signers is able to neglect the validity of the agreement (*non-repudiation*), which can also be guaranteed by a digital signature scheme.
Today, nearly all of the cryptographic schemes used in practice are based on two mathematical problems, namely the factorization of large integers and the solving of discrete logarithms. The best known examples for such schemes are the RSA cryptosystem [51] and the Digital Signature Algorithm (DSA) [41]. However, schemes based on these two problems will become insecure as soon as large enough quantum computers are built [4]. The reason for this is Shor's algorithm [53], which solves number theoretic problems like integer factorization and discrete logarithms in polynomial time on a quantum computer. Therefore it is necessary to develop alternatives to those classical cryptosystems whose security is based on problems which are not affected by Shor's algorithm. Besides lattice, code and hash based schemes, multivariate cryptography seems to be a candidate for this.

The security of multivariate cryptography is based on the PoSSo-Problem of solving systems of multivariate nonlinear polynomial equations over finite fields. Equation (1.1) shows such a system of m quadratic equations in n variables.

$$\begin{aligned} p^{(1)}(x_1,\ldots,x_n) &= \sum_{i=1}^{n}\sum_{j=i}^{n} p_{ij}^{(1)} \cdot x_i x_j + \sum_{i=1}^{n} p_i^{(1)} \cdot x_i + p_0^{(1)} \\ p^{(2)}(x_1,\ldots,x_n) &= \sum_{i=1}^{n}\sum_{j=i}^{n} p_{ij}^{(2)} \cdot x_i x_j + \sum_{i=1}^{n} p_i^{(2)} \cdot x_i + p_0^{(2)} \\ &\vdots \\ p^{(m)}(x_1,\ldots,x_n) &= \sum_{i=1}^{n}\sum_{j=i}^{n} p_{ij}^{(m)} \cdot x_i x_j + \sum_{i=1}^{n} p_i^{(m)} \cdot x_i + p_0^{(m)}. \end{aligned} \tag{1.1}$$

The PoSSo problem is defined as follows:

Problem PoSSo: Given a system $\mathcal{P} = (p^{(1)},\ldots,p^{(m)})$ of m nonlinear polynomial equations in the variables $x_1,\ldots,x_n$, find values $\bar{x}_1,\ldots,\bar{x}_n$ such that $p^{(1)}(\bar{x}_1,\ldots,\bar{x}_n) = \ldots = p^{(m)}(\bar{x}_1,\ldots,\bar{x}_n) = 0$.

The PoSSo problem is proven to be NP-hard even for the simplest case of quadratic equations over the field with two elements [27]. In the case of all polynomials being quadratic the PoSSo problem is called MQ-Problem. For efficiency reasons, nearly all multivariate schemes restrict themselves to polynomials of degree 2.

The first attempt to build a cryptographic scheme based on multivariate quadratic polynomials was done by Ong, Schnorr and Shamir in [43], where they propose a signature scheme based on the difficulty of solving the equation

$$s_1^2 + k \cdot s_2^2 = m \pmod{n}. \tag{1.2}$$

However, the security of this scheme is still based on the difficulty of factoring the modulus $n = p \cdot q$ and therefore can not be said to be a "real" multivariate scheme. Furthermore, beyond other weaknesses, the scheme is not resistant against quantum computer attacks.
The development of multivariate schemes in today's sense started in 1988 with the $C^\star$ scheme of Matsumoto and Imai [38], which can be used both for encryption and signatures. Matsumoto and Imai used a bijective map $\mathcal{F}$ over a degree n extension field $\mathbb{E}$ of GF(2) of the form

$$\mathcal{F} : \mathbb{E} \to \mathbb{E}, \mathcal{F}(X) = X^{2^\theta+1}. \tag{1.3}$$

To make sure that this map is invertible, we have to choose θ in such a way that $\gcd(2^\theta+1, 2^n-1) = 1$. Due to the Frobenius Isomorphism, equation (1.3) yields, via the canonical isomorphism between GF(2^n) and the vector space GF$(2)^n$, a system of multivariate quadratic equations $\bar{\mathcal{F}}$ over the field GF(2). To hide the structure of $\bar{\mathcal{F}}$, Matsumoto and Imai composed it in the public key with two affine maps $\mathcal{S}$ and $\mathcal{T}$. By doing so, they produced a public key of the form $\mathcal{P} = \mathcal{S} \circ \bar{\mathcal{F}} \circ \mathcal{T}$. This construction, called *bipolar construction*, is still the standard method to build multivariate public key cryptosystems.

Since the development of these early schemes, multivariate cryptography has been an active area of research and many schemes both for encryption and signatures have been proposed. However, there is a lack of "advanced" multivariate schemes like key exchange protocols and signature schemes with special properties. A good overview of existing multivariate schemes can be found in [15].

Beyond the supposed resistance against quantum computer attacks, multivariate schemes enjoy a number of advantages:

- **Speed**
  First, multivariate schemes are very fast, especially for signatures. In fact, there are many hints that multivariate schemes can be faster than classical public key schemes like RSA and ECC [11, 7].
- **Modest Computational Requirements**
  Furthermore, the mathematical operations performed by multivariate schemes are usually very simple: Most schemes need only addition and multiplication over small finite fields. Therefore multivariate schemes require only modest computational resources, which makes them attractive for the use on low cost devices such as RFID chips and smart cards, without the need of a cryptographic coprocessor. In fact, a variant of the $C^\star$ scheme called SFLASH [48] was proposed by the European Commission as a standard for signature schemes on low cost devices [49].
- **Hardness of the MQ-Problem**
  Another argument for building cryptographic schemes on the basis of the MQ-Problem is that the best known attacks against this problem are still exponential. This is in contrast to schemes based on the problem of integer factorization, which can be solved in subexponential time by algorithms like the number field sieve [35]. From this point of view, the trust in the hardness of the MQ-Problem might be stronger than for integer factorization and the parameters of multivariate schemes have not to be adapted as drastically as those of RSA.
- **Variety of Cryptographic Schemes**
  The last argument we want to mention here is that it is always nice to have cryptographic schemes based on a large variety of problems. As mentioned above, nearly all cryptographic schemes used today are based either on the integer factorization or the discrete logarithm problem. Therefore, a major cryptanalytic success against one of these two problems would lead to a severe security problem. With having a greater range of cryptographic schemes, the impacts of such a cryptanalytic break through would be much less grave.

However, there remain a number of problems to be solved:

- **Parameter Choice of Multivariate Schemes**
  The question which parameters have to be chosen for cryptosystems to reach given levels of security is one of the central problems of cryptography. A practitioner, who wants to use a certain cryptographic scheme in one of his protocols, must know which parameters he has to choose for the cryptosystem to guarantee the security of his protocol. But also for the theorist, the problem of choosing parameters is important. Developing a new cryptographic scheme without explicit parameter choice does not make much sense. There are even some cryptographic schemes for which it is unclear if there exist parameters which make the scheme both efficient and secure.
  In the area of multivariate cryptography the question of the parameter choice has not been answered so far. We address this problem in the first part of this thesis for multivariate schemes in general and in particular for the Rainbow signature scheme, which is one of the best studied and most promising multivariate schemes. By this analysis we obtain parameter sets which guarantee the security of the schemes for the years 2012 to 2050.

- **Reducing Key Sizes for Multivariate Schemes**
  One of the biggest problems of multivariate cryptography is the large size of the public and private keys. The key sizes of multivariate schemes are in the area of 10-100 kB and therefore much larger than those of classical public key cryptosystems like RSA and ECC. This problem will increase over time since the key sizes grow cubic with the number of variables. While the size of the private key can be decreased easily by using a small random seed and a pseudo random number generator (PRNG), the question, how to reduce the size of the public key is not so easy to answer. However, in many applications, especially the public key size is important. While the private key is stored on a PC or a Hardware Security Module, everybody, who wants to verify a signature or send an encrypted message, has to get the public key. So, by reducing the public key size of a cryptographic scheme, we can reduce the data traffic by a significant factor. Furthermore, reducing the public key size also helps to reduce the size of certificates as they are used in many public key infrastructures (PKI's).
  We address the problem of reducing the public key size in the second part of this thesis, where we develop a strategy which allows us to reduce the public key size of the multivariate signature schemes UOV and Rainbow by factors of 8 and 3 respectively.

- **Developing Multivariate Schemes with Provable Security**
  Another problem of multivariate cryptography is the lack of security proofs. There are many examples for multivariate schemes which were thought to be secure but broken later. Some of these schemes were "repaired" by modifying them slightly, broken again, modified and so on. A well known example for such a scheme is the $C^\star$ scheme of Matsumoto and Imai [38], which was broken by Patarin's Linearization Equations [44]. After that the original scheme was modified by removing some of the public equations (Minus-Modification). The new scheme was called SFLASH [48] and later recommended by the European Commission as a standard for signature schemes on low cost devices [49]. However, it was broken again by the differential attacks of Dubois et al. [20]. Recently, Ding et al. proposed a way to prevent these attacks by "projection" of the equations [19].
  These developments have undermined the confidence of large parts of the cryptographic community in the security of multivariate schemes in general. It is therefore necessary to create multivariate schemes which offer provable security.
  One way to achieve this is by building multivariate schemes on the basis of the provable secure identification scheme of Sakumoto, Shirai and Hiwatari [52]. However, the so obtained signature schemes are not very efficient and it is quite unclear if this way can be used to build a provable secure multivariate encryption scheme. Another approach in this direction was proposed in [32]. In this paper, Huang et al. propose a provably secure multivariate public key cryptosystem, which can be used both for encryption and signatures. However, the security of this scheme is based on a variation of the MQ-Problem. Therefore, it remains unclear if a provable secure encryption scheme can be constructed on the basis of the MQ-Problem itself.

- **Developing "Advanced" Multivariate Cryptosystems**
  While there exists a large variety of multivariate signature and encryption schemes there is a lack of more advanced cryptographic schemes such as key exchange protocols or signature schemes with special properties. So far, there exist hardly any multivariate schemes in this area. One possible direction to solve this problem is by building provable secure multivariate signature schemes with special properties on the basis of the multivariate identification scheme of [52] by using the Fiat Shamir transform [26]. One example for such a scheme is our multivariate threshold ring signature scheme [P4]. In the Future Work section of this thesis we present an idea how to build a multivariate blind signature scheme on the basis of UOV using the techniques shown in Chapter 7.

1.1 Contribution of this thesis

The contribution of this thesis is twofold.

In the first part "Selecting Parameters for Multivariate Cryptography" we deal with the problem of the parameter choice of multivariate schemes. Starting with the security model of Lenstra and Verheul [36], we analyze known attacks against multivariate schemes in general and the Rainbow signature scheme in particular to find good parameters for these schemes.

Chapter 4 "The model" describes the security model of Lenstra and Verheul [36]. The model takes as input a year y and uses parameters such as the cryptanalytic development and the budget of an attacker to compute a concrete security level a cryptographic scheme must achieve to be thought secure in the year y.

In Chapter 5 "Complexity of the MQ-Problem" we determine, based on our model, the minimal number of equations a multivariate scheme must contain to be thought secure in the year $y \in \{2012, \ldots, 2050\}$. In this chapter we look at random systems as they are used in the multivariate identification scheme [52] and the stream cipher QUAD [2] (see also Chapter 11). We do our analysis separately for multivariate systems over GF(16), GF(31) and GF(256). Additionally, this chapter provides parameter recommendations for the UOV signature scheme for different levels of security and different underlying fields.

Chapter 6 "Selecting Parameters for the Rainbow Signature Scheme" analyzes known attacks against the Rainbow signature scheme to find secure parameters for Rainbow for the years $y \in \{2012, \ldots, 2050\}$. We optimize the parameters for both public and private key size and do the analysis separately for Rainbow schemes over GF(16), GF(31) and GF(256). Furthermore, we compare the Rainbow schemes over the three fields in regard of key sizes and signature length.

In the second part "Reducing Key Sizes for Multivariate Cryptography" we address the problem of the large key sizes of multivariate schemes. We present an approach to reduce the public key size of certain multivariate schemes like the Unbalanced Oil and Vinegar (UOV) and Rainbow signature schemes. We achieve this by creating UOV and Rainbow instances with structured public keys. Furthermore, we show how the structure in the public key can be used to speed up the verification process of the signature schemes.

In Chapter 7 "The General Approach" we describe our general approach to reduce the public key size of UOV and Rainbow. We show how to insert a structured matrix B into the coefficient matrix of the public key of the schemes. By doing so, this matrix gets the form $M_P = (B|C)$ where the matrix B can be fixed by the user. As we will see in the Chapters 8 and 9, using this technique enables us to reduce the public key size of the UOV and Rainbow schemes by large factors.

Chapter 8 "Improved Versions of UOV" presents several improved versions of the UOV signature scheme, which reduce the size of the public key of the original scheme by a factor of up to 8.0. We describe the key generation of different improved variants of the UOV scheme and discuss the security of our constructions. Furthermore, we propose concrete parameter sets for our schemes and compare them with the original UOV scheme in terms of the public key size. Finally, we describe how the implementation of our improved schemes can be done in an efficient way.

In Chapter 9 "Improved Versions of Rainbow" we present two improved versions of the Rainbow signature scheme. As we will see, using a structured matrix B together with the techniques presented in Chapter 7 reduces the public key size of the original scheme by a factor of up to 3.0. We analyze the security of our improved versions and give details about the implementation.

Chapter 10 "Speeding up the Verification Process" shows how the structure in the public key of our improved schemes can be used to speed up the verification process of the signature schemes. As we will see, we can use our techniques to speed up the verification process of UOV and Rainbow by factors of up to 6.1 and 2.4 respectively. We analyze the speed up both theoretically and by a C implementation of our schemes.

In the last chapter of this thesis (Chapter 11; "Speeding up QUAD") we show how the techniques presented in Chapter 10 can be used to speed up the multivariate stream cipher QUAD. By using specially designed polynomial systems, the data throughput of QUAD can be increased by a factor of up to 6.8. As in the previous chapter, we derive our results both theoretically and by a C implementation of QUAD.

Chapter 2

Multivariate Cryptography

In this chapter we give an overview of the basics of multivariate cryptography needed in the later parts of this thesis. After recalling the basic definitions of multivariate polynomials in Section 2.1, Section 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 Section 2.4 deals with a phenomenon called equivalent keys, by which we denote different private keys which correspond to the same public key. Finally, Section 2.5 describes known attacks against the MQ-Problem which can be used as message recovery respectively signature forgery attacks against multivariate schemes.

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 thesis.

Definition 2.1. *We define the ring of multivariate polynomials in n variables over a (finite) field* $\mathbb{F}$ *as*

$$\mathbb{F}[x_1, \ldots, x_n] = \left\{ \sum_{i=1}^{s} c_i \cdot t_{a_i} | s \in \mathbb{N}, a_i = (a_{i,1}, \ldots, a_{i,n}) \in \mathbb{N}_0^n \right\}. \tag{2.1}$$

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

Definition 2.2. *The degree of a monomial* $t_a = x_1^{a_1} \cdot x_2^{a_2} \cdot \ldots \cdot x_n^{a_n} \in \mathbb{F}[x_1, \ldots, x_n]$ *is defined as*

$$\deg(t_a) = |a| = \sum_{j=1}^{n} a_j. \tag{2.2}$$

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

$$\deg(p) = \max_{i \in \{1,\ldots,s\}} \deg(t_{a_i}) = \max_{i \in \{1,\ldots,s\}} |a_i|. \tag{2.3}$$

Definition 2.3. *For a monomial* $t_a = x_1^{a_1} \cdot x_2^{a_2} \cdot \ldots \cdot x_n^{a_n} \in \mathbb{F}[x_1, \ldots, x_n]$ *we define*

$$\log(t_a) = (a_1, a_2, \ldots, a_n) \in \mathbb{F}^n. \tag{2.4}$$

Definition 2.4. *An ordering 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 $t_1 >_\sigma t_2$. The ordering σ is called admissible if, for any $t_1, t_2, t_3 \in T^n$ we have*

- $t_1 \geq_\sigma 1$ *and*
- $t_1 >_\sigma t_2 \Leftrightarrow t_1 \cdot t_3 >_\sigma t_2 \cdot t_3$.

The three most often used examples for admissible orderings of monomials are the pure *lexicographic order* (lex), the *graded lexicographic order* (glex) and the *reversed graded lexicographic order* (grevlex).

- In the pure *lexicographic 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 lexicographic 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 *reversed graded lexicographic order* we have

$$t_1 >_{\text{grevlex}} t_2 \Leftrightarrow \deg(t_1) > \deg(t_2)$$
$$\text{or } \deg(t_1) = \deg(t_2) \text{ and the last non zero component of } \log(t_1) - \log(t_2) \text{ is negative.}$$

Unless otherwise stated, we use the graded lexicographic order throughout this thesis.

After having fixed an ordering of monomials σ (not necessarily an admissible one), we can give the following definitions:

Definition 2.5. *For a multivariate polynomial*

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

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

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

Definition 2.6. *For a system $\mathcal{P}$ of m multivariate polynomials*[1]

$$\begin{aligned} p^{(1)}(x_1, \ldots, x_n) &= \sum_{i=1}^{s} c_i^{(1)} \cdot t_{a_i} \\ p^{(2)}(x_1, \ldots, x_n) &= \sum_{i=1}^{s} c_i^{(2)} \cdot t_{a_i} \\ &\vdots \\ p^{(m)}(x_1, \ldots, x_n) &= \sum_{i=1}^{s} c_i^{(m)} \cdot t_{a_i} \end{aligned}$$

with $t_{a_1} >_\sigma t_{a_2} >_\sigma \ldots >_\sigma t_{a_s}$ we define the Macauley matrix of $\mathcal{P}$ by

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

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

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

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

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 equation (2.7) and the graded lexicographic order of monomials the Macauley matrix (see Definition 2.6) has the form

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

We get

$$\begin{pmatrix} p^{(1)} \\ p^{(2)} \\ \vdots \\ p^{(m)} \end{pmatrix}(x_1,\ldots,x_n) = M_P \cdot (x_1^2, x_1x_2, \ldots, x_n^2, x_1, \ldots, x_n, 1)^T. \tag{2.9}$$

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

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

We get

$$p^{(k)}(x_1,\ldots,x_n) = (x_1,\ldots,x_n,1)\cdot MP^{(k)}\cdot(x_1,\ldots,x_n,1)^T \quad (k=1,\ldots,m). \tag{2.11}$$

Definition 2.7. *We call a quadratic system $\mathcal{P}$ homogeneous quadratic, if it has no linear and constant terms (i.e. all the coefficients $p_i^{(k)}$ $(i = 0, \ldots, n,\ k = 1, \ldots, m)$ of equation (2.7) are fixed to zero).*

In the case of a homogeneous quadratic system $\mathcal{P}$ the matrices $MP^{(k)}$ of equation (2.9) are of the form

$$MP^{(k)} = \begin{pmatrix} p_{11}^{(k)} & p_{12}^{(k)} & p_{13}^{(k)} & \ldots & p_{1n}^{(k)} \\ 0 & p_{22}^{(k)} & p_{23}^{(k)} & \ldots & p_{2n}^{(k)} \\ 0 & 0 & p_{33}^{(k)} & & p_{3n}^{(k)} \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \ldots & 0 & p_{nn}^{(k)} \end{pmatrix} \in \mathbb{F}^{n \times n} \tag{2.12}$$

and we get

$$p^{(k)}(x_1, \ldots, x_n) = (x_1, \ldots, x_n) \cdot MP^{(k)} \cdot (x_1, \ldots, x_n)^T \quad (k = 1, \ldots, m). \tag{2.13}$$

In the context of multivariate public key cryptosystems we often use the symmetric matrices $\widetilde{MP}^{(k)}$ associated to the homogeneous quadratic part of the system $\mathcal{P}$. These matrices are defined as

$$\widetilde{MP}^{(k)} = MP^{(k)} + (MP^{(k)})^T = \begin{pmatrix} 2 \cdot p_{11}^{(k)} & p_{12}^{(k)} & p_{13}^{(k)} & \ldots & p_{1n}^{(k)} \\ p_{12}^{(k)} & 2 \cdot p_{22}^{(k)} & p_{23}^{(k)} & \ldots & p_{2n}^{(k)} \\ p_{13}^{(k)} & p_{23}^{(k)} & 2 \cdot p_{33}^{(k)} & & p_{3n}^{(k)} \\ \vdots & & \ddots & & \vdots \\ p_{1n}^{(k)} & p_{2n}^{(k)} & \ldots & p_{n-1,n}^{(k)} & 2 \cdot p_{nn}^{(k)} \end{pmatrix} \quad (k = 1, \ldots, m). \tag{2.14}$$

Note that for multivariate systems $\mathcal{P}$ defined over fields of characteristic 2 the entries on the main diagonal of the matrices $\widetilde{MP}^{(k)}$ are 0.

Definition 2.8. *For a multivariate quadratic system $\mathcal{P}$ we define the differential or polar form $\mathcal{G}$ as*

$$\mathcal{G}(\mathbf{x}, \mathbf{y}) = \mathcal{P}(\mathbf{x} + \mathbf{y}) - \mathcal{P}(\mathbf{x}) - \mathcal{P}(\mathbf{y}) + \mathcal{P}(\mathbf{0}). \tag{2.15}$$

Note that $\mathcal{G}$ is homogeneous quadratic and bilinear in $\mathbf{x}$ and $\mathbf{y}$. We can write each component of $\mathcal{G}$ as a bilinear form

$$\mathbf{x} \cdot G^{(k)} \cdot \mathbf{y}^T \tag{2.16}$$

with a symmetric matrix $G^{(k)} = MP^{(k)} + (MP^{(k)})^T$ $(k = 1, \ldots, m)$. Note that for fields of characteristic 2 the elements on the main diagonal of $G^{(k)}$ are zero.

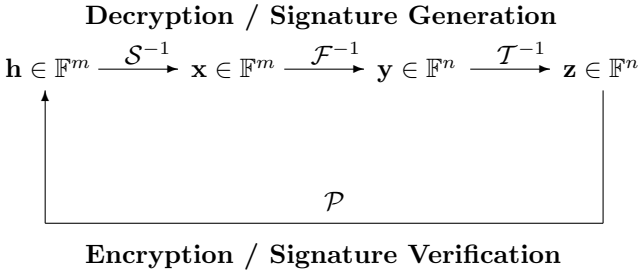


Figure 2.1: General workflow of bipolar schemes

2.2 The Standard (Bipolar) Construction

The basic idea behind the standard construction of multivariate cryptography is to choose a system $\mathcal{F} : \mathbb{F}^n \to \mathbb{F}^m$ of m multivariate quadratic polynomials in n variables which can be easily inverted (central map). After that one chooses two affine invertible maps $\mathcal{S} : \mathbb{F}^m \to \mathbb{F}^m$ and $\mathcal{T} : \mathbb{F}^n \to \mathbb{F}^n$ 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 $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$ which is supposed to be hardly distinguishable from a random system and therefore be difficult to invert. The private key consists of $\mathcal{S}$, $\mathcal{F}$ and $\mathcal{T}$ and therefore allows to invert $\mathcal{P}$.
The standard process for encryption and decryption / signature generation and verification works as shown in Figure 2.1.

Encryption Schemes ($m \geq n$)

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

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

Signature Schemes ($m \leq n$)

Signature Generation: To sign a document d, we use a hash function $\mathcal{H} : \{0,1\}^\star \to \mathbb{F}^m$ to compute the value $\mathbf{h} = \mathcal{H}(d) \in \mathbb{F}^m$. Then we compute $\mathbf{x} = \mathcal{S}^{-1}(\mathbf{h})$, $\mathbf{y} = \mathcal{F}^{-1}(\mathbf{x})$ 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 the possibly many) preimage of $\mathbf{x}$ under the central map $\mathcal{F}$. Since $n \geq m$ we can be sure that such a preimage exists. Therefore every message has a signature.

Verification: To verify the authenticity of a document, one simply computes $\mathbf{h}' = \mathcal{P}(\mathbf{z})$ and the hash value $\mathbf{h} = \mathcal{H}(d)$ of the document. If $\mathbf{h}' = \mathbf{h}$ 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 $C^\star$ [38] and ℓiC [17]), this is the IP2S-Problem, if $\mathcal{F}$ is a part of the private key (as for UOV [47] and Rainbow [16]), it is the EIP-Problem. More information regarding these underlying problems can be found in Section 2.3.

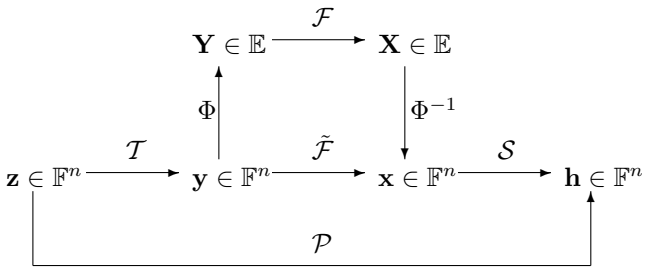


Figure 2.2: Construction of BigField schemes

2.2.1 A Short Overview of Bipolar Constructions

Most of the existing multivariate public key cryptosystems are bipolar constructions. According to the type of the central map in use one can distinguish three main types:

- *BigField Schemes*
  For a BigField scheme we have $m = n$ which makes these schemes suitable both for encryption and signatures. We define a degree n extension field $\mathbb{E}$ of $\mathbb{F}$ by $\mathbb{E} = \mathbb{F}[X]/p(X)$ for an irreducible polynomial $p(X) \in \mathbb{F}[X]$ of degree n and define $\Phi : \mathbb{F}^n \to \mathbb{E}$ to be the canonical isomorphism from the vector space $\mathbb{F}^n$ to $\mathbb{E}$, i.e.

  $$\Phi(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i \cdot X^{i-1}.$$

  The central map of the scheme is a bijective map $\mathcal{F} : \mathbb{E} \to \mathbb{E}$ which, due to the Frobenius isomorphism, can be transformed to a quadratic map $\tilde{\mathcal{F}} = \Phi^{-1} \circ \mathcal{F} \circ \Phi : \mathbb{F}^n \to \mathbb{F}^n$. The public key of the scheme is the composed map $\mathcal{P} = \mathcal{S} \circ \tilde{\mathcal{F}} \circ \mathcal{T} = \mathcal{S} \circ \Phi^{-1} \circ \mathcal{F} \circ \Phi \circ \mathcal{T}$ with two affine maps $\mathcal{S} : \mathbb{F}^n \to \mathbb{F}^n$ and $\mathcal{T} : \mathbb{F}^n \to \mathbb{F}^n$ (see Figure 2.2).
  One of the first schemes in this area and one of the first multivariate schemes in general is the $C^\star$ scheme of Matsumoto and Imai [38]. After this scheme was broken by the Linearization Equations of Patarin [44], many variations of this original scheme have been proposed. As an example we mention here the SFLASH [48] schemes of Courtois et al. which were later recommended by the European Commission as a standard for digital signatures on restricted devices [49]. However, SFLASH was broken by the differential attacks of Dubois et al. [20], after which it was recommended to use a projection technique to hide the structure of the central map [19]. Other schemes from this group are Square [12] and HFE [46].

- *MiddleField Schemes*
  MiddleField Schemes are similar to BigField schemes in the sense that they use an extension field, too. However, the extension field $\mathbb{E}$ is not of degree n, but of degree $l = \frac{n}{k}$ for some integer k. The vector $\mathbf{y} \in \mathbb{F}^n$ is lifted to a vector $\mathbf{Y} \in \mathbb{E}^k$, bijectively mapped to some $\mathbf{X} \in \mathbb{E}^k$ and sent down to $\mathbf{x} \in \mathbb{F}^n$. As for the BigField schemes we have $m = n$, which makes MiddleField schemes suitable for both encryption and signatures.
  The best known examples for MiddleField schemes are MFE [58] and ℓ-invertible cycles [17].

- *SingleField Schemes*
  In contrast to BigField and MiddleField schemes, all the computations of a SingleField scheme are done in one (relatively small) field. We usually have $m < n$ which restricts SingleField constructions to signature schemes. Two of the best analyzed and most often used multivariate schemes, namely UOV [47] and Rainbow [16] belong to this group of schemes. Other examples for SingleField schemes are the TTS schemes (Tame Transformation Systems) [61]. In the rest of this thesis, we mostly look at SingleField schemes and in particular at UOV and Rainbow.

2.2.2 Other Constructions

Mixed Systems

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

$$\mathcal{F}(x_1, \ldots, x_n, y_1, \ldots, y_m) = (h_1, \ldots, h_m). \tag{2.17}$$

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

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

$$\mathcal{F}(\bar{x}_1, \ldots, \bar{x}_n, y_1, \ldots, y_m) : \mathbb{F}^m \to \mathbb{F}^m$$

gets linear.

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

$$\mathcal{F}(x_1, \ldots, x_n, \bar{y}_1, \ldots, \bar{y}_m) : \mathbb{F}^n \to \mathbb{F}^m$$

is an efficiently invertible system of 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 $\mathcal{S} : \mathbb{F}^n \to \mathbb{F}^n$, $\mathcal{T} : \mathbb{F}^m \to \mathbb{F}^m$ and $\mathcal{L} : \mathbb{F}^m \to \mathbb{F}^m$ are invertible affine maps. Therefore, $\mathcal{P}$ is a quadratic map from $\mathbb{F}^{m+n}$ to $\mathbb{F}^m$. But, for any fixed $(\bar{x}_1, \ldots, \bar{x}_n) \in \mathbb{F}^n$, the system

$$\mathcal{P}(\bar{x}_1, \ldots, \bar{x}_n, y_1, \ldots, y_m)$$

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

Encryption Schemes ($m \geq n$)

Encryption: To encrypt a message $\bar{X} = (\bar{x}_1, \ldots, \bar{x}_n) \in \mathbb{F}^n$ one solves the linear system $\mathcal{P}(\bar{x}_1, \ldots, \bar{x}_n, y_1, \ldots, y_m) = (0, \ldots, 0)$ for $y_1, \ldots, y_m$. The solution $\bar{Y} = (\bar{y}_1, \ldots, \bar{y}_m) \in \mathbb{F}^m$ is the ciphertext of the message $\bar{X}$.

Decryption: To decrypt a ciphertext $\bar{Y} \in \mathbb{F}^m$, one first computes $\hat{Y} = (\hat{y}_1, \ldots, \hat{y}_m) = \mathcal{L}^{-1}(\bar{Y})$. Then one solves

$$\mathcal{F}(x_1, \ldots, x_n, \hat{y}_1, \ldots, \hat{y}_m) = (0, \ldots, 0) \tag{2.18}$$

for $x_1, \ldots, x_n$. Note that equation (2.18) is, due to condition (C2), efficiently invertible. We denote the solution by $\hat{X} = (\hat{x}_1, \ldots, \hat{x}_n) \in \mathbb{F}^n$. Finally, one computes the plaintext $\bar{X} = \mathcal{S}^{-1}(\hat{X})$. Since we have $m \geq n$, the resulting plaintext is unique.

Signature Schemes ($m \leq n$)

Signature Generation: To sign a document d, one uses a hash function $\mathcal{H} : \{0,1\}^\star \to \mathbb{F}^m$ to compute a hash value $\bar{Y} = \mathcal{H}(d) = (\bar{y}_1, \ldots, \bar{y}_m) \in \mathbb{F}^m$. One computes $\hat{Y} = (\hat{y}_1, \ldots, \hat{y}_m) = \mathcal{L}^{-1}(\bar{Y})$ and solves

$$\mathcal{F}(x_1, \ldots, x_n, \hat{y}_1, \ldots, \hat{y}_m) = (0, \ldots, 0). \tag{2.19}$$

for $x_1, \ldots, x_n$. Again, (2.19) is a system of quadratic equations which, due to condition (C2), is efficiently invertible. Denote this solution by $\hat{X} = (\hat{x}_1, \ldots, \hat{x}_n)$. The signature of the message m is $\bar{X} = \mathcal{S}^{-1}(\hat{X}) \in \mathbb{F}^n$. 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 $\bar{X} = (\bar{x}_1, \ldots, \bar{x}_n) \in \mathbb{F}^n$, one computes the hash value $\bar{Y} = \mathcal{H}(d) = (\bar{y}_1, \ldots, \bar{y}_m)$ and evaluates $\mathcal{P}(\bar{x}_1, \ldots, \bar{x}_n, \bar{y}_1, \ldots, \bar{y}_m)$. If the result is $(0, \ldots, 0) \in \mathbb{F}^m$, the signature is accepted, otherwise it is rejected.

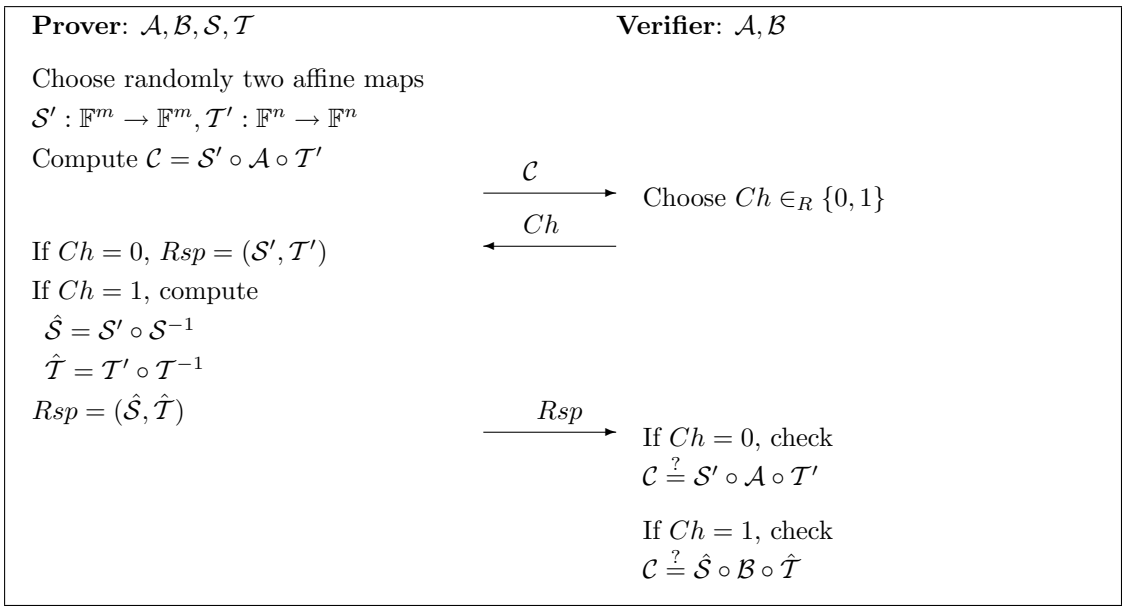


Figure 2.3: IP identification scheme of [46]

There exist only very few schemes of the mixed systems type. Examples for such schemes are the Dragon cryptosystems of Patarin [45].

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 Section 2.3).

Additionally to these constructions for multivariate 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. Usually 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 the following we describe two different constructions of such a prove based on multivariate quadratic polynomials.

IP-based Identification

The IP identification scheme [46] can be described as follows:

Key Generation: The prover P randomly chooses a system $\mathcal{A} : \mathbb{F}^n \to \mathbb{F}^m$ of multivariate quadratic polynomials and two affine maps $\mathcal{S} : \mathbb{F}^m \to \mathbb{F}^m$ and $\mathcal{T} : \mathbb{F}^n \to \mathbb{F}^n$. 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 2.3 shows one round of the protocol.

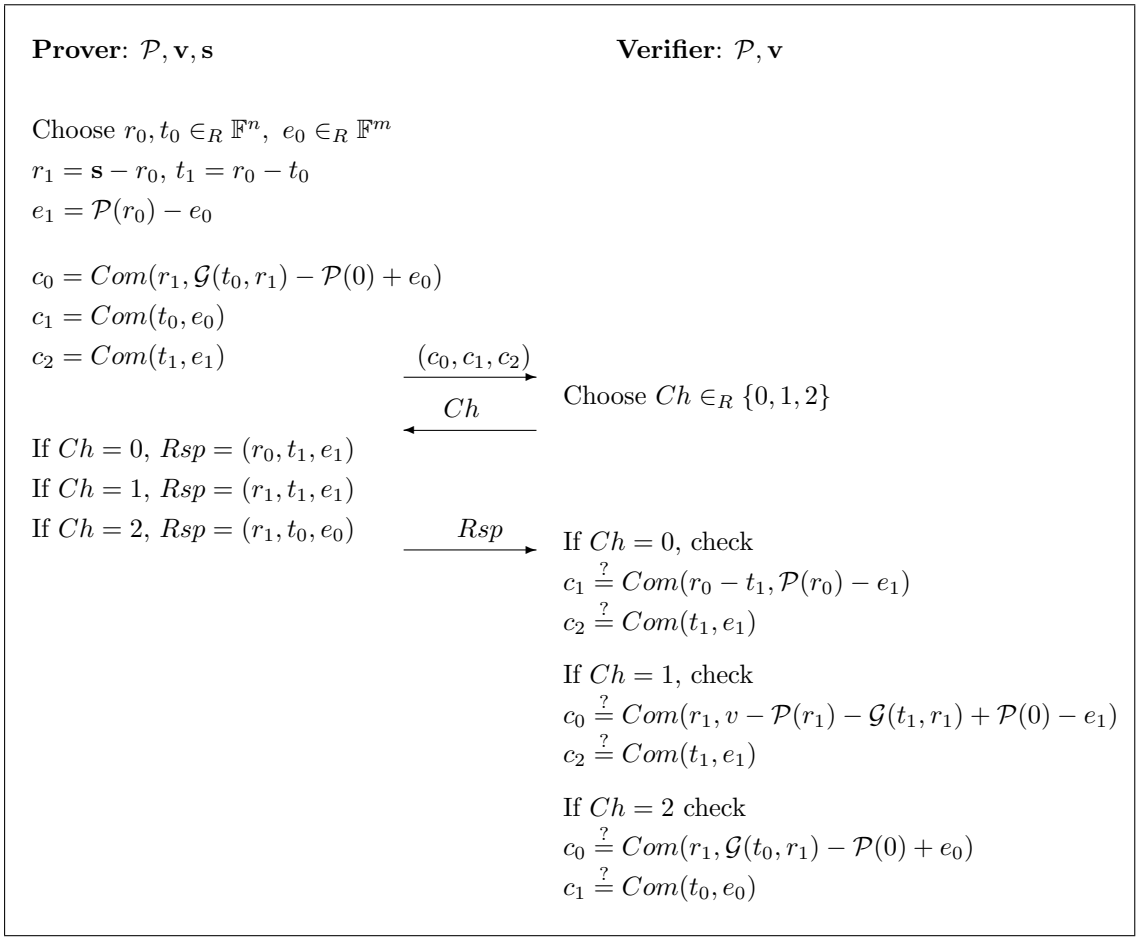


Figure 2.4: MQ identification scheme of [52]

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 construction, the IP-identification scheme can be extended to a signature scheme. The security of the scheme is based on the IP2S-Problem (see Subsection 2.3.2).

MQ-based identification

One very new construction is the identification scheme of Sakumoto et al. [52] whose security is based solely on the MQ-problem (see Subsection 2.3.1). Therefore it is one of the very few multivariate schemes which offer provable security.

Key Generation: The prover randomly chooses a system $\mathcal{P} : \mathbb{F}^n \to \mathbb{F}^m$ of multivariate quadratic polynomials and a vector $\mathbf{s} \in \mathbb{F}^n$. He computes $\mathbf{v} = \mathcal{P}(\mathbf{s}) \in \mathbb{F}^m$. The public key consists of $\mathcal{P}$ and $\mathbf{v}$, the private key is the vector $\mathbf{s}$.
Figure 2.4 shows one round of the identification protocol. Here,

$$\mathcal{G}(\mathbf{x}, \mathbf{y}) = \mathcal{P}(\mathbf{x} + \mathbf{y}) - \mathcal{P}(\mathbf{x}) - \mathcal{P}(\mathbf{y}) + \mathcal{P}(\mathbf{0}) \tag{2.20}$$

is the polar form of the system $\mathcal{P}$ (see Definition 2.8).

The scheme is a zero-knowledge argument of knowledge for a solution of the system $\mathcal{P}(\mathbf{x}) = \mathbf{v}$. The cheating probability per round is $\frac{2}{3}$. Therefore, one needs 52 rounds to reduce the impersonation probability to less than 2^{-30}.
Via the Fiat-Shamir construction [26] this identification scheme can be extended to a provable secure multivariate signature scheme. Furthermore, the identification scheme can be used as a basis for the construction of more advanced provable secure signature schemes. One example for such a scheme is our threshold ring signature scheme [P4].

We do not consider the latter three constructions in the rest of this thesis. So, whenever we speak of a multivariate public key scheme, we refer to a scheme of the bipolar type.

2.3 Underlying Problems

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

2.3.1 The MQ-Problem

The central problem on which the security of all multivariate cryptosystems is based is the

Problem PoSSo (Polynomial System Solving): Given a system $\mathcal{P} = (p^{(1)}, \ldots, p^{(m)})$ of m nonlinear polynomial equations in the variables $x_1, \ldots, x_n$, find values $\bar{x}_1, \ldots, \bar{x}_n$ such that $p^{(1)}(\bar{x}_1, \ldots, \bar{x}_n) = \ldots = p^{(m)}(\bar{x}_1, \ldots, \bar{x}_n) = 0$.

The PoSSo-Problem is proven to be NP-complete even for the simplest case of quadratic polynomials over GF(2) [27] (in its decisional variant). More precisely, it can be shown to be equivalent to the 3SAT problem.
For efficiency reasons, most multivariate schemes restrict to quadratic polynomials. For the special case of all polynomials $p^{(1)}, \ldots, p^{(m)}$ having degree 2, the PoSSo-Problem is called **MQ-Problem** (for Multivariate Quadratic).

The PoSSo-Problem is one of the central problems in all areas of cryptography, since nearly all cryptographic schemes can be written as systems of nonlinear polynomial equations. Especially for the cryptanalysis of symmetric ciphers like block and stream ciphers this so called *algebraic cryptanalysis* plays a major role [50].
In Section 2.5 we present known attacks against the MQ-Problem. In contrary to the case of integer factorization, all these attacks have exponential complexity (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.

Problem IP1S (Isomorphism of Polynomials with 1 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 $\mathcal{B} = \mathcal{A} \circ \mathcal{T}'$.

Problem IP2S (Isomorphism of Polynomials with 2 Secrets): Given nonlinear multivariate systems $\mathcal{A}$ and $\mathcal{B}$ such that $\mathcal{B} = \mathcal{S} \circ \mathcal{A} \circ \mathcal{T}$ for linear or affine maps $\mathcal{S}$ and $\mathcal{T}$, find two maps $\mathcal{S}'$ and $\mathcal{T}'$ such that $\mathcal{B} = \mathcal{S}' \circ \mathcal{A} \circ \mathcal{T}'$.

Problem EIP (Extended Isomorphism of Polynomials): Given 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 $\mathcal{F}$ belonging to a special

class of nonlinear polynomial systems $\mathcal{C}$, 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 C*, Square and ℓiC).
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.

In contrast to the MQ-Problem, there is not much known about the difficulty of the IP-Problem. In fact, for some multivariate schemes (e.g. the balanced Oil and Vinegar signature scheme [47]) the decomposition of the public key $\mathcal{P}$ turned out to be very easy [34]. This fact prevented researchers to give security proofs for multivariate public key schemes. In fact, the existing provably secure multivariate schemes are based on the MQ-based identification scheme of Subsection 2.2.2.

2.4 Equivalent Keys

A surprising fact of many multivariate public key schemes is that for one public key, there exists a large number of different private keys. This fact was first observed in [59]. We define

Definition 2.9. *Let $((\mathcal{S}, \mathcal{F}, \mathcal{T}), \mathcal{P})$ be a key pair of a multivariate public key cryptosystem. A second private key $(\mathcal{S}', \mathcal{F}', \mathcal{T}')$ is called equivalent to $(\mathcal{S}, \mathcal{F}, \mathcal{T})$ if it leads to the same public key, i.e.*

$$(\mathcal{S}', \mathcal{F}', \mathcal{T}') \equiv (\mathcal{S}, \mathcal{F}, \mathcal{T}) \Leftrightarrow \mathcal{S}' \circ \mathcal{F}' \circ \mathcal{T}' = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T} = \mathcal{P}. \tag{2.21}$$

Note that $\mathcal{F}'$ has to be a valid central map of the multivariate scheme, i.e. it must have the same structure as $\mathcal{F}$.

For a private key $(\mathcal{S}, \mathcal{F}, \mathcal{T})$, we denote the set of all private keys equivalent to $(\mathcal{S}, \mathcal{F}, \mathcal{T})$ by $EQ_{(\mathcal{S},\mathcal{F},\mathcal{T})}$.

To find for a given private key $(\mathcal{S}, \mathcal{F}, \mathcal{T})$ the set $EQ_{(\mathcal{S},\mathcal{F},\mathcal{T})}$ one introduces two additional affine maps $\Sigma : \mathbb{F}^m \to \mathbb{F}^m$ and $\Omega : \mathbb{F}^n \to \mathbb{F}^n$ (see [59]). We get

$$\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T} = \underbrace{\mathcal{S} \circ \Sigma^{-1}}_{\mathcal{S}'} \circ \underbrace{\Sigma \circ \mathcal{F} \circ \Omega}_{\mathcal{F}'} \circ \underbrace{\Omega^{-1} \circ \mathcal{T}}_{\mathcal{T}'}. \tag{2.22}$$

One then analyzes, which form Σ and Ω must have such that $\mathcal{F}'$ is a valid central map of the multivariate public key scheme. Such affine maps Σ and Ω are called *sustainers* [59]. In Section 3.2 and 3.5 we consider the question of equivalent keys for UOV and Rainbow respectively.
The concept of equivalent keys plays a major role in the cryptanalysis of multivariate schemes. Since for an attacker it is sufficient to find any of the equivalent keys, he can look for an equivalent key with simple structure. This fact is used in many attacks against multivariate schemes (see Sections 3.3 and 3.6).

2.5 Generic Attacks

In this section we give an overview of the most important methods for solving systems of nonlinear multivariate equations.
These so called direct attacks can be used against each multivariate scheme as a message recovery attack (encryption schemes) or signature forgery attack (signature schemes).
Additionally, these attacks can be used in algebraic cryptanalysis to attack symmetric schemes such as block and stream ciphers [50].

Throughout this thesis, we denote by the term "complexity" the number of field multiplications an algorithm performs before outputting a solution. Similarly, the term "bit complexity" denotes the base 2 logarithm of this number.

2.5.1 Relinearization

The Relinearization attack aims at solving overdetermined systems of multivariate quadratic equations. Let $\mathcal{P}$ be a system of m quadratic equations in the n variables $x_1, \ldots, x_n$. The main idea is to introduce a new variable x_{ij} for each quadratic monomial $x_i x_j$. By doing so, one gets a system of linear equations which, if the number of equations is large enough, can be solved by Gaussian elimination. One has to test whether the so obtained solution is really a solution of the quadratic system, i.e. if $x_{ij} = x_i x_j \ \forall i, j \in \{1, \ldots, n\}$.
To solve a (dense) system of quadratic equations in n variables by this method, one needs $m \geq \frac{(n+1)\cdot(n+2)}{2} - 1$ equations.

2.5.2 XL

In [14] Courtois et al. proposed a method called "eXtended Linearization" or XL-Algorithm.
Let T_D^n be the set of all monomials in $\mathbb{F}[x_1, \ldots, x_n]$ of degree $\leq D$. The XL-Algorithm works as shown in Algorithm 2.1.

Algorithm 2.1 XL-Algorithm

Input: Set of quadratic polynomials $F = \{f^{(1)}, \ldots, f^{(m)}\}$
Output: vector $\mathbf{x} = (x_1, \ldots, x_n)$ such that $f^{(1)}(\mathbf{x}) = \ldots = f^{(m)}(\mathbf{x}) = 0$
1: **for** $i = 1$ to n **do**
2: Fix an integer $D > 2$.
3: Generate all polynomials $h \cdot f^{(j)}$ with $h \in T_{D-2}^n$ and $j = 1, \ldots, m$.
4: Perform Gaussian Elimination on the set of all polynomials generated in the previous step to generate one equation containing only x_i.
5: If step 4 produced at least one univariate polynomial in x_i, solve this polynomial by e.g. Berlekamp's algorithm.
6: Simplify the equations $f^{(1)}, \ldots, f^{(m)}$ by substituting the value of x_i.
7: **end for**
8: **return** $\mathbf{x} = (x_1, \ldots, x_n)$

If the degree D is too small, line 4 will not produce a univariate polynomial. In this case one has to increase D and try again. The smallest degree for which the XL-Algorithm outputs a solution of the system F is called the *degree of regularity* d_{reg}.

2.5.3 Gröbner Bases

Gröbner Bases as introduced by B. Buchberger in [10] allow us to find all the 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 2.10. *An ideal in $\mathbb{F}[x_1, \ldots, x_n]$ is a subset $I \subset \mathbb{F}[x_1, \ldots, x_n]$ such that for each element $a \in \mathbb{F}[x_1, \ldots, x_n]$ and any element $b \in I$ we have $a \cdot b \in I$.*
A subset $M \subset I$ is called generating system of the ideal I (we write $I = \langle M \rangle$)

$$\Leftrightarrow \forall a \in I \ \exists m_1, \ldots, m_s \in M \text{ and } \alpha_1, \ldots, \alpha_s \in \mathbb{F}[x_1, \ldots, x_n] \text{ such that } a = \sum_{i=1}^{s} \alpha_i \cdot m_i \qquad (2.23)$$

Theorem 2.1. *For every ideal $I \subset \mathbb{F}[x_1, \ldots, x_n]$ there exists a finite generating system.*

Proof. See [42], Theorem 1.8. □

A ring which fulfills the condition of Theorem 2.1 is called *noetherian ring.*

Let σ be an admissible ordering of monomials (see Definition 2.4).

Definition 2.11. *Let $f = \sum_{i=1}^{s} c_i \cdot t_i$ be a polynomial in $\mathbb{F}[x_1, \ldots, x_n]$. W.l.o.g. we assume that we have $t_1 >_\sigma t_2 >_\sigma \ldots >_\sigma t_s$. Then we call*

- *t_1 the leading monomial of f with respect to the monomial ordering σ. We denote it by $\mathrm{LM}_\sigma(f)$ or $\mathrm{LM}(f)$.*
- *c_1 the leading coefficient of f with respect to the monomial ordering σ. We denote it by $\mathrm{LC}_\sigma(f)$ or $\mathrm{LC}(f)$.*
- *$c_1 \cdot t_1$ the leading term of f with respect to σ. We denote it by $\mathrm{LT}_\sigma(f)$ or $\mathrm{LT}(f)$.*

Note that we have

$$\mathrm{LT}(f) = \mathrm{LC}(f) \cdot \mathrm{LM}(f).$$

Definition 2.12. *A Gröbner Basis of an ideal I is a subset $G = \{g^{(1)}, \ldots, g^{(s)}\} \subset I$ such that*

$$I = \langle g^{(1)}, \ldots, g^{(s)} \rangle \text{ and } \langle \mathrm{LT(I)} \rangle = \langle \mathrm{LT}(g^{(1)}), \ldots, \mathrm{LT}(g^{(s)}) \rangle. \tag{2.24}$$

Remark 2.1. *Definition 2.12 shows that the term Gröbner Basis depends on the chosen monomial ordering. For example, a Gröbner Basis with respect to the lexicographic order is in general no Gröbner Basis with respect to the graded reversed lexicographic order.*

Reduction of Polynomials

Let $g \in \mathbb{F}[x_1, \ldots, x_n]$ and $F = \{f^{(1)}, \ldots, f^{(s)}\} \subset \mathbb{F}[x_1, \ldots, x_n]$. The polynomial g can be reduced modulo F to a polynomial $h \in \mathbb{F}[x_1, \ldots, x_n]$ (we write $g \mapsto_F h$), if $\exists\ p^{(i)} \in \mathbb{F}[x_1, \ldots, x_n]$ $(i = 1, \ldots, s)$ such that

$$h = g - \sum_{i=1}^{s} p^{(i)} \cdot f^{(i)}. \tag{2.25}$$

Definition 2.13. *The polynomial $g \in \mathbb{F}[x_1, \ldots, x_n]$ is called completely reduced with respect to $F = \{f^{(1)}, \ldots, f^{(s)}\}$ if no term of g is divisible by any $\mathrm{LM}(f^{(i)})$ for all $f^{(i)} \in F$.*

Theorem 2.2. *Let $F = \{f^{(1)}, \ldots, f^{(s)}\}$ be an ordered set of polynomials in $\mathbb{F}[x_1, \ldots, x_n]$. Then, for any $g \in \mathbb{F}[x_1, \ldots, x_n]$ there exist polynomials $p^{(1)}, \ldots, p^{(s)}$ such that*

$$g = \sum_{i=1}^{s} p^{(i)} \cdot f^{(i)} + r,$$

with a polynomial $r \in \mathbb{F}[x_1, \ldots, x_n]$ being completely reduced with respect to F.

Proof. See [42], Theorem 4.6. □

Definition 2.14. *The polynomial r of Theorem 2.2 is called normal form of g with respect to F.*

For $F = \{f^{(1)}, \ldots, f^{(s)}\}$ being a random subset of $\mathbb{F}[x_1, \ldots, x_n]$ the normal form r is not uniquely determined and depends on the order of the polynomials $f^{(i)} \in F$. However, we get

Theorem 2.3. *If $G = \{g^{(1)}, \ldots, g^{(s)}\}$ is a Gröbner Basis, the normal form r of a polynomial $h \in \mathbb{F}[x_1, \ldots, x_n]$ is uniquely determined and independent of the order of the polynomials $g^{(i)} \in G$.*

Proof. See [42], Theorem 4.13. □

Buchberger's Algorithm

In his thesis [10] Buchberger developed an efficient algorithm to compute a Gröbner Basis of the ideal generated by a set $F = \{f^{(1)}, \ldots, f^{(m)}\} \subset \mathbb{F}[x_1, \ldots, x_n]$ of polynomials. The most important notion in Buchberger's algorithm [10] is the so called S-Polynomial.

Definition 2.15. *Let $f, g \in \mathbb{F}[x_1, \ldots, x_n]$. The S-Polynomial of f and g is defined by*

$$\mathrm{Spoly}(f, g) = \frac{\mathrm{LCM}(\mathrm{LT}(f), \mathrm{LT}(g))}{\mathrm{LT}(g)} \cdot g - \frac{\mathrm{LCM}(\mathrm{LT}(f), \mathrm{LT}(g))}{\mathrm{LT}(f)} \cdot f. \tag{2.26}$$

In his thesis Buchberger discovered the following criterion on a set G for being a Gröbner Basis:

Theorem 2.4. *Let $I \subset \mathbb{F}[x_1, \ldots, x_n]$ be an ideal in the polynomial ring $\mathbb{F}[x_1, \ldots, x_n]$. A subset $G \subset I$ with $I = \langle G \rangle$ is a Gröbner Basis of I if and only if*

$$\mathrm{NormalForm}(\mathrm{Spoly}(p, q)) = 0 \quad \forall\ p, q \in G.$$

Proof. See [42], Theorem 4.18. □

We can use this criterion to construct an algorithm for finding a Gröbner Basis of an ideal $I = \langle F \rangle \subset \mathbb{F}[x_1, \ldots, x_n]$ (see Algorithm 2.2).

Algorithm 2.2 Buchberger's Algorithm

Input: $F = \{f^{(1)}, \ldots, f^{(m)}\}$, monomial ordering σ
Output: Gröbner Basis $G = \{g^{(1)}, \ldots, g^{(s)}\}$ of $I = \langle f^{(1)}, \ldots, f^{(m)} \rangle$

```
1:  G ← F
2:  repeat
3:      G' ← G
4:      for each pair {p, q}, p ≠ q ∈ G' do
5:          S ← NormalForm(Spoly(p, q), G)
6:          if S ≠ 0 then
7:              G ← G ∪ {S}
8:          end if
9:      end for
10: until G = G'
11: return G
```

Theorem 2.5. *Algorithm 2.2 outputs a Gröbner Basis of the ideal I with respect to the monomial ordering σ after finitely many steps.*

Proof. See [42], Theorem 4.19. □

Improvements of Buchberger's Algorithm and F_4/F_5

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 2.16. *A Gröbner Basis $G = \{g^{(1)}, \ldots, g^{(s)}\}$ is said to be reduced, if all polynomials $g^{(i)}$ are monic and $\mathrm{LM}(g^{(i)})$ does not divide $\mathrm{LM}(g^{(j)})$ for all $i \neq j,\ 1 \leq i < j \leq s$.*

Theorem 2.6. *If G and H are reduced Gröbner Bases generating the same ideal, then $G = H$.*

Proof. See [42], Theorem 4.21. □

Thus, if we reduce the set G each time we enlarge it (i.e. after line 7 of Algorithm 2.2), the output of the algorithm will be unique.

In his thesis [10], Buchberger discovered two criteria to avoid reductions to zero.

- If $\mathrm{LT}(g^{(i)})$ and $\mathrm{LT}(g^{(j)})$ are relatively prime, then $\mathrm{Spoly}(g^{(i)}, g^{(j)})$ reduces to zero and can be ignored.
- If there exists $g^{(k)} \in G$ such that $\mathrm{LT}(g^{(k)})$ divides $\mathrm{LCM}(\mathrm{LT}(g^{(i)}), \mathrm{LT}(g^{(j)}))$, and if $\mathrm{Spoly}(g^{(i)}, g^{(k)})$ and $\mathrm{Spoly}(g^{(j)}, g^{(k)})$ have already been considered, $\mathrm{Spoly}(g^{(i)}, g^{(j)})$ reduces to zero and can be ignored.

As a further improvement, Faugère suggested for his F_4 [23] and F_5 [24] algorithms to compute many of the normal forms in one go by using the Macauley matrix of the system and fast linear algebra.
During the computation one has to deal with polynomials of high degree, which makes the coefficient matrices of the system very large and the Gaussian Elimination extremely costly. The largest degree appearing during the computation of a Gröbner Basis is called the *degree of regularity* d_{reg}. The complexity of solving a system of m quadratic equations in n variables using the F_5 algorithm is given by [5]

$$\mathrm{complexity}_{F_5}(m, n) = \mathcal{O}\left(m \cdot \binom{n + d_{\mathrm{reg}} - 1}{d_{\mathrm{reg}}}\right)^{\omega}, \tag{2.27}$$

where the degree of regularity d_{reg} can be estimated by the lowest integer D such that the coefficient of t^D in

$$\frac{(1-t^2)^m}{(1-t)^n} \tag{2.28}$$

is less or equal to 0 and $2 < \omega \leq 3$ is the linear algebra constant of solving a linear system. Note that the upper estimation of d_{reg} holds only for semi regular ($\sim$ random) systems. For the public systems of multivariate schemes the degree of regularity can be smaller (e.g. HFE, see [21]).

When solving an underdetermined system of m equations in $n > m$ variables with a Gröbner Basis technique, one usually fixes some of the variables to create a determined system. The reason for this is that for underdetermined systems the solution space has a very complicated structure and it is very difficult to compute this variety. By fixing $n - m$ variables one creates a determined system with much simpler (often zero dimensional) variety.

The running time of the Gröbner Basis algorithms depends heavily on the chosen monomial ordering. While, to identify the solution of a system, Gröbner Bases with respect to the lexicographic order are better, Gröbner Bases with repect to the graded reversed lexicographic order can be computed much faster. Therefore, it is usually more efficient to compute a Gröbner Basis with respect to this order first and then to use linear algebra techniques to transform it into a Gröbner Basis with respect to the lexicographic order (FGLM-Algorithmus, see [25]).

Hybrid approach

When solving a determined multivariate nonlinear polynomial system it is often a good strategy to guess some variables to create an overdetermined system. This strategy is used by algorithms like HybridF_5 [5], which combines exhaustive search with Faugères F_5 algorithm. Although, by this strategy, one has to run the F_5 algorithm several times to find a solution of the original system, this often reduces the time needed to solve the system. In fact, the HybridF_5 algorithm is currently the fastest generic method to solve nonlinear polynomial systems over fields larger than GF(2).
The complexity of solving a system of m quadratic equations in n variables over a field with q elements using the HybridF_5 algorithm is given by [5]

$$\mathrm{complexity}_{\mathrm{HF}_5}(q, m, n) = \min_{k \geq 0} q^k \cdot \mathrm{comp}_{F_5}(m, n-k) = \min_{k \geq 0} q^k \cdot \mathcal{O}\left(m \cdot \binom{n - k + d_{\mathrm{reg}} - 1}{d_{\mathrm{reg}}}\right)^{\omega}. \tag{2.29}$$

Here, the degree of regularity can be estimated as for the F_5 algorithm.

2.5.4 Solving Underdetermined Systems

In [33] Kipnis et al. proposed an algorithm which solves an underdetermined system $\mathcal{P}$ of m quadratic equations in $n \geq m \cdot (m+1)$ variables over a finite field $\mathbb{F}$ of characteristic 2 in polynomial time.
The goal of this approach is to find a change of basis such that all coefficients $p_{ij}^{(k)}$ of quadratic cross terms vanish. By doing so, one has to solve basically a system of linear equations which can be done in polynomial time. To achieve this, we define a linear map $\mathcal{T}$ (given by an $n \times n$ matrix $M_T = (t_{ij})_{i,j=1}^n$) and a new quadratic map $\mathcal{F} = \mathcal{P} \circ \mathcal{T}$. The coefficients $f_{ij}^{(k)}$ of this map $\mathcal{F}$ are given by

$$f_{ij}^{(k)} = \sum_{r=1}^{n} \sum_{s=r}^{n} \sigma_{ij}^{rs} \cdot p_{rs}^{(k)} (1 \leq i \leq j \leq n), \tag{2.30}$$

where the coefficients σ_{ij}^{rs} are given by

$$\sigma_{ij}^{rs} = \begin{cases} t_{ri} \cdot t_{si} & i = j \\ t_{ri} \cdot t_{sj} + t_{rj} \cdot t_{si} & \text{otherwise} \end{cases} . \tag{2.31}$$

Let $\Pi^{(k)}$ and $\Phi^{(k)}$ be the coefficient vectors of the k-th component of $\mathcal{P}$ and $\mathcal{F}$ respectively (see Definition 2.5). Then we can write equation (2.30) in the form

$$\Phi^{(k)} = \Sigma \cdot (\Pi^{(k)})^T \ (k = 1, \ldots, m), \tag{2.32}$$

with an $\frac{n \cdot (n+1)}{2} \times \frac{n \cdot (n+1)}{2}$ matrix

$$\Sigma = \begin{pmatrix} \sigma_{11}^{11} & \sigma_{11}^{12} & \ldots & \sigma_{11}^{nn} \\ \sigma_{12}^{11} & \sigma_{12}^{12} & & \sigma_{12}^{nn} \\ \vdots & & \ddots & \vdots \\ \sigma_{nn}^{11} & \sigma_{nn}^{12} & \ldots & \sigma_{nn}^{nn} \end{pmatrix} = \begin{pmatrix} t_{11}^2 & t_{11}t_{21} & \ldots & t_{n1}^2 \\ 2 \cdot t_{11}t_{12} & t_{11}t_{22} + t_{12}t_{21} & \ldots & 2 \cdot t_{n1}t_{n2} \\ \vdots & & \ddots & \vdots \\ t_{1n}^2 & t_{1n}t_{2n} & \ldots & t_{nn}^2 \end{pmatrix}. \tag{2.33}$$

We denote the row of Σ belonging to the monomial $x_i x_j$ by (i, j). Note that all the elements of the row (i, j) are bilinear in one element of the i-th and one element of the j-th column of M_T.

Kipnis et al. now fixed the first column of the matrix M_T. Therefore all the elements in the row $(1, 1)$ of Σ are fixed, which also fixes the coefficients $f_{11}^{(k)}$ $(k = 1, \ldots, m)$ to some value of $\mathbb{F}$. Furthermore all the elements of the row (1,i) $(i = 2, \ldots, n)$ become linear. Row $(1, 2)$ gives us m relations of the form

$$f_{12}^{(k)} = \underbrace{2 \cdot t_{11} \cdot p_{11}^{(k)}}_{\in \mathbb{F}} \cdot t_{12} + \underbrace{t_{11} \cdot p_{12}^{(k)}}_{\in \mathbb{F}} \cdot t_{22} + \underbrace{t_{21} \cdot p_{12}^{(k)}}_{\in \mathbb{F}} \cdot t_{12} + \ldots + \underbrace{2 \cdot t_{n1} \cdot p_{nn}^{(k)}}_{\in \mathbb{F}} \cdot t_{n2} \ (k = 1, \ldots, m). \tag{2.34}$$

If $n \geq m$, we can use Gaussian Elimination to find values for the t_{i2} $(i = 1, \ldots, n)$ such that $f_{12}^{(k)} = 0 \ \forall k = 1, \ldots, m$.
In the next step we compute the elements t_{i3} $(i = 1, \ldots, n)$ in such a way that $f_{13}^{(k)} = f_{23}^{(k)} = 0 \ \forall k = 1, \ldots, m$. We get a system of $2 \cdot m$ equations in n variables. To be sure that this system has a solution, we need $n \geq 2 \cdot m$.
We continue this process to remove all quadratic crossterms $x_i x_j$ $(1 \leq i < j \leq n)$ from the system $\mathcal{F}$. In the last step we have to solve a linear system of $(m-1) \cdot m$ equations in n variables. To be sure that this system has a solution, we need $n \geq m \cdot (m-1)$.

At the end of this process, the transformed system $\mathcal{F} = \mathcal{P} \circ \mathcal{T}$ consists of m equations of the form

$$\sum_{i=1}^{m} \beta_i x_i^2 + \sum_{i=1}^{m} x_i \cdot L_i(x_{m+1}, \ldots, x_n) + Q_i(x_{m+1}, \ldots, x_n) = 0 \tag{2.35}$$

where L_i is some linear relation in the variables $x_{m+1}, \ldots, x_n$ and Q_i is quadratic. By Gaussian Elimination we can find values for $x_{m+1}, \ldots, x_n$ such that $L_i = 0 \ \ \forall i$. We evaluate the quadratic maps Q_i at these variables and solve the remaining system for $x_1, \ldots, x_m$.

Remark 2.2. *For multivariate systems over fields with odd characteristic the above attack has exponential running time. The reason for this is that only half of the elements of such a field are squares. Therefore one needs to run the algorithm about 2^m times to find a solution of the system.*

In [40] Miura et al. extended the above attack to systems with $n \geq m \cdot (m+3)/2$ variables. Another improvement of the attack was done by Thomae and Wolf. In [56] they found by a sharp analysis of Kipnis' attack a way to solve underdetermined systems with $m < n < m \cdot (m+1)$ variables faster. In particular, one can use their algorithm to reduce a system of m equations in $\omega \cdot m$ variables to a system of $m - \lfloor \omega - 1 \rfloor$ equations and variables. This has to be considered for the parameter choice of UOV.

Remark 2.3. *The technique used by the attack as described above is very similar to the technique used in this thesis to reduce the public key size of UOV and Rainbow (see Chapter 7). In particular, equations* (2.30) *and* (2.31) *are exactly the same as in our construction.*

2.5.5 Solving Systems over GF(2)

When solving systems over GF(2) the techniques described above are not very efficient, since Gröbner Basis attacks will soon run out of memory. In this subsection we describe two approaches for solving such systems, namely an improved exhaustive search method and a technique called SAT-Solving. Whereas most of the multivariate public key encryption and signature schemes work over larger fields than GF(2), these attacks play an important role in the algebraic cryptanalysis of block and stream ciphers.

Improved Exhaustive Search using Gray Codes

In [9] Bouillaguet et al. presented an efficient algorithm for solving multivariate quadratic systems over GF(2) which extensively makes use of the so called Gray code.

Let $p : \mathrm{GF}(2)^n \to \mathrm{GF}(2)$ be a quadratic polynomial with coefficients and variables in GF(2) and e_k be the k-th unit vector in $\mathrm{GF}(2)^n$. A possible solution $i \in \mathrm{GF}(2)^n$ of the polynomial p can be seen as an integer $i \in \{0, \ldots, 2^n - 1\}$. Furthermore let $b(i)$ be the index of the least significant non zero bit of i.

Definition 2.17. *The standard Gray code $G : \{0, \ldots, 2^n - 1\} \to \{0,1\}^n$ is defined by [31]*

$$G(0) = 0^n, \quad G(i) = G(i-1) + e_{b(i)}. \tag{2.36}$$

In contrast to the standard binary code, $G(i)$ and $G(i-1)$ differ, regardless of the choice of i, in only one bit. This fact enabled Bouillaguet et al. to speed up their computations by a large factor. For a function $f : \mathrm{GF}(2)^n \to \mathrm{GF}(2)$ we define

$$d_k \ f(i) = f(i + e_k) - f(i) \ \ (k = 1, \ldots, n). \tag{2.37}$$

Such we get

$$p(G(i)) = p(G(i-1) + e_{b(i)}) = p(G(i-1)) + d_{b(i)} \ p(G(i-1)). \tag{2.38}$$

Note that $d_k\ p$ $(k = 1, \ldots, n)$ is an affine function from $\mathrm{GF}(2)^n$ to $\mathrm{GF}(2)$, which enables us to compute $p(G(i))$ from $p(G(i-1))$ very efficiently.
By extending this technique to systems of polynomials, the authors of [9] were able to proof

Theorem 2.7. *The common zeroes of m (random) quadratic polynomials in n variables can be found after having performed approximately $log_2(n) \cdot 2^{n+2}$ bit operations.*

Proof. See [9], Theorem 2. □

In particular, this complexity is independent from the number of equations m.

Remark 2.4. *The attack as described above is an improvement of an exhaustive search in the solution space. By using the concept of Gray codes the possible solutions are reorderd in such a way that the computations are much more efficient.*

SAT solvers

Another approach to solve multivariate polynomial systems over GF(2) is called SAT-solving. As mentioned in Subsection 2.3.1, the PoSSo problem is equivalent to the 3SAT problem. Therefore it is possible to translate a system of multivariate polynomials into an instance of a SAT (Satisfiability) problem. In particular, a system of multivariate polynomial equations $\mathcal{P}(x_1, \ldots, x_n) = (p^{(1)}(x_1, \ldots, x_n), \ldots, p^{(m)}(x_1, \ldots, x_n)) = 0$ is transformed into a conjunction of clauses. A clause is a disjunction of literals, each of which is either a variable x_i (called *atom*) or its negation $\bar{x}_i$. A boolean formula of this form is called to be in *conjunctive normal form* (CNF).
The SAT solver now looks for an interpretation that satisfies the boolean formula, i.e. it tries to find truth values for all the atoms in such a way that each of the clauses is evaluated to be true.
For the actual solution of the boolean formula their exists a number of algorithms such as MINISAT [22] and CryptoMiniSat [54].

Chapter 3

UOV and Rainbow

In this chapter we describe two of the best studied and most promising multivariate schemes, namely the (Unbalanced) Oil and Vinegar and the Rainbow signature schemes. Additionally to the description of the schemes (Sections 3.1 and 3.4), we study the question of equivalent keys for UOV and Rainbow (Section 3.2 and 3.5 respectively) and present the known attacks against the two schemes (Section 3.3 and 3.6).

3.1 The (Unbalanced) Oil and Vinegar Signature Scheme

In this section we introduce the Oil and Vinegar signature scheme, which was proposed by J. Patarin in [47].

Let $\mathbb{F}$ be a finite field with q elements. Let o and v be two integers and set $n = o + v$. We define two index sets $V = \{1, \ldots, v\}$ and $O = \{v+1, \ldots, n\}$. The variables x_i $(i \in V)$ are called Vinegar variables and the variables x_j $(j \in O)$ Oil variables.
The central map $\mathcal{F}$ of the scheme consists of o polynomials $f^{(1)}, \ldots, f^{(o)} \in \mathbb{F}[x_1, \ldots, x_n]$ of the form

$$f^{(k)}(\mathbf{x}) = \sum_{i,j \in V, i \leq j} \alpha_{ij}^{(k)} x_i x_j + \sum_{i \in V, j \in O} \beta_{ij}^{(k)} x_i x_j + \sum_{i \in V \cup O} \gamma_i^{(k)} x_i + \eta^{(k)} \quad (k = 1, \ldots, o). \tag{3.1}$$

Note that Oil and Vinegar variables are not completely mixed, just like oil and vinegar in a salad dressing.

To invert the central map $\mathcal{F}$, one chooses random values for the Vinegar variables $x_1, \ldots, x_v$ and substitutes them into the polynomials $f^{(1)}, \ldots, f^{(o)}$. By doing so one gets a system of o linear equations in the o Oil variables $x_{v+1}, \ldots, x_n$ which can be solved by e.g. Gaussian Elimination.
If the system does not have a solution, one has to choose other values for the Vinegar variables and try again. However, this happens only with very small probability. Therefore, in most cases, one gets a valid signature at the first try.

To hide the structure of the central map $\mathcal{F}$ in the public key, one composes it with an affine invertible map $\mathcal{T} : \mathbb{F}^n \to \mathbb{F}^n$ (given by a matrix $M_T \in \mathbb{F}^{n \times n}$ and a vector $c_T \in \mathbb{F}^n$).

The public key is therefore given by $\mathcal{P} = \mathcal{F} \circ \mathcal{T}$, the private key consists of $\mathcal{F}$ and $\mathcal{T}$ and therefore allows to invert the public key.

Remark 3.1. *The second affine map $\mathcal{S}$ used in the bipolar construction (see Section 2.2) is not needed for the security of UOV. The reason for this is that a linear combination of polynomials of the form* (3.1) *still has the form of* (3.1)*. Therefore, the composition $\mathcal{S} \circ \mathcal{F}$ does not change the structure of the central map $\mathcal{F}$ and can be dropped.*

The process of signature generation/verification can be described as follows:

Signature generation: To sign a document d, one uses a hash function $\mathcal{H} : \{0,1\}^\star \to \mathbb{F}^o$ to compute the hash value $\mathbf{h} = \mathcal{H}(d) \in \mathbb{F}^o$ of the document. After that one computes recursively $\mathbf{y} = \mathcal{F}^{-1}(\mathbf{h})$ and $\mathbf{z} = \mathcal{T}^{-1}(\mathbf{y})$. The signature of the document d is $\mathbf{z} \in \mathbb{F}^n$. Here, $\mathcal{F}^{-1}(\mathbf{h})$ means finding one (of approximately q^v) preimage of $\mathbf{h}$ under $\mathcal{F}$. As mentioned above, this step is performed by fixing the Vinegar variables at random and solving the resulting linear system for the Oil variables.

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

In his original paper [47], Patarin suggested to use $o = v$ (Balanced Oil and Vinegar (OV)). After this scheme was broken by Kipnis and Shamir in [34], it was proposed in [33] to set $v > o$ (Unbalanced Oil and Vinegar (UOV)).

The size of the public key is given by

$$\text{size}_{\text{pk UOV}} = o \cdot \frac{(n+1)\cdot(n+2)}{2} \tag{3.2}$$

field elements, the size of the private key is

$$\text{size}_{\text{sk UOV}} = n \cdot (n+1) + o \cdot \left(\frac{v \cdot (v+1)}{2} + o \cdot v + n + 1 \right) \tag{3.3}$$

field elements. Recommended parameters for the UOV signature scheme can be found in Section 5.4.

3.2 Equivalent Keys for UOV

In this section we consider the question of equivalent keys for the UOV scheme. We follow hereby the approach of Section 2.4 and [59]. Let $((\mathcal{F}, \mathcal{T}), \mathcal{P})$ be a UOV key pair and introduce an additional affine map $\Omega : \mathbb{F}^n \to \mathbb{F}^n$. We write

$$\mathcal{P} = \mathcal{F} \circ \mathcal{T} = \underbrace{\mathcal{F} \circ \Omega}_{\mathcal{F}'} \circ \underbrace{\Omega^{-1} \circ \mathcal{T}}_{\mathcal{T}'} \tag{3.4}$$

The following theorem answers the question, for which affine maps Ω the map $\mathcal{F}' = \mathcal{F} \circ \Omega$ is a valid UOV central map.

Theorem 3.1. *Let $(\mathcal{F}, \mathcal{T})$ be a UOV private key and $\Omega : \mathbb{F}^n \to \mathbb{F}^n$ be an affine map whose linear part has the form*

$$\Omega_{\text{lin}} = \begin{pmatrix} \Omega^{(1)}_{v\times v} & 0_{v\times o} \\ \Omega^{(3)}_{o\times v} & \Omega^{(4)}_{o\times o} \end{pmatrix}. \tag{3.5}$$

Then $(\mathcal{F}', \mathcal{T}')$ with $\mathcal{F}' = \mathcal{F} \circ \Omega$ and $\mathcal{T}' = \Omega^{-1} \circ \mathcal{T}$ is an equivalent UOV private key.

Proof. see [60], Theorem 4.12. □

Note that Ω_{lin}^{-1} has the same form as Ω_{lin}.

Corollary 3.1. *Let $(\mathcal{F}, \mathcal{T})$ be a UOV private key over a field with q elements. Then the set $EQ_{(\mathcal{F},\mathcal{T})}$ of private keys equivalent to $(\mathcal{F}, \mathcal{T})$ has the size*

$$|EQ_{(\mathcal{F},\mathcal{T})}| = q^n \cdot \prod_{i=0}^{v-1}(q^v - q^i) \cdot q^{o \cdot v} \cdot \prod_{i=1}^{o-1}(q^o - q^i). \tag{3.6}$$

Proof. The number of elements in $EQ_{(\mathcal{F},\mathcal{T})}$ is the same as the number of possible choices of the map Ω. The first factor in equation (3.6) represents the q^n possible choices of the constant part of Ω, the remaining factors describe the possible choices of the matrix Ω_{lin} of Theorem 3.1. Note that both $\Omega^{(1)}$ and $\Omega^{(4)}$ must be invertible. □

Remark 3.2. *Corollary 3.1 states that for each UOV public key $\mathcal{P}$ there exist $q^{n+o\cdot v} \cdot \prod_{i=0}^{v-1}(q^v - q^i) \cdot \prod_{i=0}^{o-1}(q^o - q^i)$ UOV private keys $(\mathcal{F}, \mathcal{T})$ such that $\mathcal{F} \circ \mathcal{T} = \mathcal{P}$.*

The next theorem states that for every UOV public key there exists a corresponding UOV private key of a very special form.

Theorem 3.2. *Let $\mathcal{P}$ be a UOV public key. Then, with overwhelming probability, there exists a UOV private key $(\widetilde{\mathcal{F}}, \widetilde{\mathcal{T}})$ with $\widetilde{\mathcal{F}} \circ \widetilde{\mathcal{T}} = \mathcal{P}$, such that $M_{\widetilde{T}}$ has the form*

$$M_{\widetilde{T}} = \begin{pmatrix} 1_{v\times v} & T'_{o\times v} \\ 0_{v\times o} & 1_{o\times o} \end{pmatrix} \tag{3.7}$$

Note that $M_{\widetilde{T}}^{-1}$ has the same form as $M_{\widetilde{T}}$.

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 of the form shown in the theorem. To do this, we compute Ω_{lin} in such a way that $\Omega_{\text{lin}} \cdot M_{\widetilde{T}} = M_T$. Let $\Omega_{\text{lin}} = \begin{pmatrix} \tilde{\Omega}_1 & 0 \\ \tilde{\Omega}_3 & \tilde{\Omega}_4 \end{pmatrix}$ and $M_T = \begin{pmatrix} T_1 & T_2 \\ T_3 & T_4 \end{pmatrix}$. If T_1 is invertible[1], 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 $M_{\widetilde{T}}$ can be written as a product of matrices $T_{v+1} \cdot \ldots \cdot T_n$ with

$$T_i = \begin{pmatrix} 1 & & 0 & 0 & t'_{1i} & 0 \\ & \ddots & & \vdots & \vdots & \vdots \\ 0 & & 1 & 0 & t'_{vi} & 0 \\ 0 & \ldots & 0 & 1 & 0 & 0 \\ \vdots & & \vdots & & \ddots & \\ 0 & \ldots & 0 & 0 & & 1 \end{pmatrix} \quad (i = v+1, \ldots, n). \tag{3.8}$$

The matrices T_i contain the same non zero elements as the matrix T' of equation (3.7). By inversion we get $M_{\widetilde{T}}^{-1} = T_n^{-1} \cdot \ldots \cdot T_{v+1}^{-1}$. Note that T_i^{-1} has of the form (3.8) ($\forall\ i = v+1, \ldots, n$).

[1] If T_1 is not invertible, we can switch rows and columns of M_T by renumbering the variables until we get an invertible matrix.

3.3 Attacks against UOV

We define the matrix $F^{(k)}$ to be the symmetric matrix associated to the homogeneous quadratic part of the k-th component of $\mathcal{F}$ $(k = 1, \ldots, o)$ (see equation (2.14)). Due to the special structure of the central map the matrices $F^{(k)}$ have the form

$$F^{(k)} = \begin{pmatrix} F_{11}^{(k)} & F_{12}^{(k)} \\ F_{21}^{(k)} & 0 \end{pmatrix}, \tag{3.9}$$

where $F_{11}^{(k)}$ is a $v \times v$ matrix, $F_{12}^{(k)} \in \mathbb{F}^{v \times o}$ and $F_{21}^{(k)} = (F_{12}^{(k)})^T \in \mathbb{F}^{o \times v}$ $(k = 1, \ldots, o)$.
Analogously, we denote by $P^{(k)}$ the symmetric matrix associated to the homogeneous quadratic part of the k-th component of the public map $\mathcal{P}$ $(k = 1, \ldots, o)$. Such we obtain

$$P^{(k)} = M_T^T \cdot F^{(k)} \cdot M_T \text{ and} \tag{3.10}$$

$$F^{(k)} = (M_T^{-1})^T \cdot P^{(k)} \cdot M_T^{-1} \quad (k = 1, \ldots, o). \tag{3.11}$$

3.3.1 Direct Attack

The most straightforward method to attack the UOV signature scheme is to try to solve the public system $\mathcal{P}(\mathbf{z}) = \mathbf{h}$ for $\mathbf{z}$ (signature forgery attack). In the case of UOV, these systems are highly underdetermined (we have $n = \alpha \cdot o$, where α is usually chosen to be 3). Therefore one can use the attack on underdetermined systems of [56] to reduce the number of equations in the system by $\lfloor \alpha - 1 \rfloor$. (see Subsection 2.5.4). In order to solve the resulting quadratic system the attacker can use an arbitrary method such as XL or a Gröbner Basis method like F_4/F_5 (see Section 2.5).

3.3.2 UOV-Reconciliation Attack

The UOV-Reconciliation attack [18] is based on the following observation:
Let $((\mathcal{F}, \mathcal{T}), \mathcal{P})$ be a UOV key pair such that M_T has the form of equation (3.7). We write M_T^{-1} as a product of matrices $\tilde{T}_n \cdot \ldots \cdot \tilde{T}_{v+1}$ with matrices $\tilde{T}_j$ having the form of equation (3.8). Note that each of these matrices contains, besides the 1's on the main diagonal, only v non-zero elements. With this notation equation (3.11) yields

$$F^{(k)} = \underbrace{(\tilde{T}_{v+1})^T \cdot \ldots \cdot \overbrace{(\tilde{T}_{n-1})^T \cdot \overbrace{(\tilde{T}_n)^T \cdot P^{(k)} \cdot \tilde{T}_n}^{P_{n-1}^{(k)}} \cdot \tilde{T}_{n-1}}^{P_{n-2}^{(k)}} \ldots \cdot \tilde{T}_{v+1}}_{P_v^{(k)}} \ (k = 1, \ldots, o) \tag{3.12}$$

with matrices $P_j^{(k)}$ of the form

$$P_j^{(k)} = \begin{pmatrix} \star_{j \times j} & \star_{j \times (n-j)} \\ \star_{(n-j) \times j} & 0_{(n-j) \times (n-j)} \end{pmatrix} \quad (j = v, \ldots, n-1, \ k = 1, \ldots, o). \tag{3.13}$$

The matrices $P_v^{(k)}$ $(k = 1, \ldots, o)$ 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 $(k = 1, \ldots, o)$ a sequence of matrices $P_j^{(k)}$ $(j = n-1, \ldots, v)$ of the form (3.13). At the end of this process, we will get matrices $P_v^{(k)}$ $(k = 1, \ldots, o)$, which, together with the matrix $M_T^{-1} = \tilde{T}_n \cdot \ldots \cdot \tilde{T}_{v+1}$ 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 $P_j^{(k)}$ $(j = n-1, \ldots, v)$. We have

$$\underbrace{\begin{pmatrix} \star & \ldots & \star & \star & \ldots & \ldots & \star \\ \vdots & & \vdots & \vdots & & & \vdots \\ \star & \ldots & \star & \star & \ldots & \ldots & \star \\ \star & \ldots & \star & 0_{j,j} & \ldots & \ldots & 0_{j,n} \\ \vdots & & \vdots & \vdots & 0 & \ldots & 0 \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ \star & \ldots & \star & 0_{n,j} & 0 & \ldots & 0 \end{pmatrix}}_{P_j^{(k)}} = \tilde{T}_{j+1}^T \cdot \underbrace{\begin{pmatrix} a_{11} & \ldots & a_{1,j-1} & a_{1,j} & \ldots & \ldots & a_{1n} \\ \vdots & & \vdots & \vdots & & & \vdots \\ a_{j-1,1} & \ldots & a_{j-1,j-1} & a_{j-1,j} & \ldots & \ldots & a_{j-1,n} \\ a_{j,1} & \ldots & a_{j,j-1} & a_{j,j} & \ldots & \ldots & a_{j,n} \\ \vdots & & \vdots & \vdots & 0 & \ldots & 0 \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ a_{n1} & \ldots & a_{n,j-1} & a_{n,j} & 0 & \ldots & 0 \end{pmatrix}}_{P_{j+1}^{(k)}} \cdot \tilde{T}_{j+1}. \tag{3.14}$$

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 $\tilde{T}_{j+1}$. Most of the elements of the matrix $P_j^{(k)}$ are unknown, but we know that the elements in the lower right corner must be zero. Each of these zero elements yields one quadratic equation in the v unknown elements of $\tilde{T}_{j+1}$. Since the equations delivered by $0_{k,l}$ and $0_{l,k}$ are the same, equation (3.14) yields $n-j$ quadratic equations. Altogether we get $o \cdot (n-j)$ quadratic equations in v variables (for $k = 1, \ldots, o$). By solving this system, we get the matrix $\tilde{T}_{j+1}$ and can use equation (3.14) to compute $P_j^{(k)}$ $(k = 1, \ldots, o)$.
By repeating this process we can, starting with the matrices $P_n^{(k)} = P^{(k)}$, find o matrices $P_v^{(k)}$ $(k = 1, \ldots o)$ which, together with the matrix $M_T^{(-1)} = \tilde{T}_n \cdot \ldots \cdot \tilde{T}_{v+1}$, 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 $(n-j) \cdot o$ quadratic equations in v variables $(j = n-1, \ldots, v)$. The complexity of the attack is mainly given by the complexity of solving the first system of o quadratic equations in v variables.

Algorithm 3.1 shows the UOV-Reconciliation attack in a compact form.

Algorithm 3.1 UOV-Reconciliation attack

Input: matrices $P_n^{(k)}$ $(k = 1, \ldots, o)$ representing the homogeneous quadratic parts of the public polynomials
Output: private key (represented by matrices $F^{(k)}$ $(k = 1, \ldots, o)$ and a matrix M_T)
1: **for** $j = n-1$ to v **do**
2: Define a matrix T_{j+1} of the form (3.8).
3: Define for $k = 1, \ldots o$ matrices $P_j^{(k)}$ of the form (3.13).
4: Compute for $k = 1, \ldots, o$ 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 equation 3.14) yields for every $k = 1, \ldots, o$ $n-j$ quadratic equations in the elements of T_{j+1}. Altogether we get therefore a system of $o \cdot (n-j)$ 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 $k = 1, \ldots, o$ 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: $M_T \leftarrow T_n \cdot \ldots \cdot T_{v+1}$
9: $F^{(k)} \leftarrow P_v^{(k)}$ $(k = 1, \ldots, o)$
10: **return** $F^{(k)}$ $(k = 1, \ldots, o)$, M_T

3.3.3 UOV Attack

The goal of the UOV attack [33] is to find an equivalent private key by finding an equivalent affine map $\mathcal{T}$ of the form of equation (3.7). The attack was originally used to break the balanced Oil and Vinegar signature scheme [47]. In this subsection we describe the generalization to the unbalanced case ($v > o$).

Definition 3.1. *We define the Oil subspace of $\mathbb{F}^n$ as*

$$\mathcal{O} = \{x = (x_1, \ldots, x_n) \in \mathbb{F}^n : x_1 = \ldots = x_v = 0\}.$$

The Vinegar subspace is the set

$$\mathcal{V} = \{x = (x_1, \ldots, x_n) \in \mathbb{F}^n : x_{v+1} = \ldots = x_n = 0\}.$$

The goal of the attack is to find the preimage of the Oil subspace under the map $\mathcal{T}$.

Let $E : \mathbb{F}^n \to \mathbb{F}^n$ be a linear transformation of the form of equation (3.9). Then we get

Lemma 3.1. *1. $E(\mathcal{O})$ is an o dimensional subspace of $\mathcal{V}$.*
2. If E is invertible, $E^{-1}(\mathcal{V})$ is an v dimensional subspace of $\mathbb{F}^n$ in which $\mathcal{O}$ is a proper subspace.

Proof. See [33], Lemma 2. □

Remark 3.3. *In the case of the balanced Oil and Vinegar signature scheme we have $E(\mathcal{O}) = \mathcal{V}$ and $E^{-1}(\mathcal{V}) = \mathcal{O}$. This fact makes the whole attack much easier and enables us to find an equivalent map $\mathcal{T}$ in polynomial time.*

Let $H = \sum_{i=1}^{o} \lambda_i \cdot F^{(i)}$ be a linear combination of the matrices $F^{(i)}$ associated to the homogeneous quadratic part of the i-th central polynomial. Note that H has the form of (3.9). We obtain

Theorem 3.3. *Assume that, for some ($k \in \{1, \ldots, o\}$), the matrix $F^{(k)}$ is invertible. Then, with probability not less than q^{o-v}, the map $(F^{(k)})^{-1} \cdot H$ has a non trivial invariant subspace which is also a subspace of $\mathcal{O}$.*

Proof. See [33], Theorem 4.1. □

Theorem 3.3 yields

Theorem 3.4. *Let $W = \sum_{i=1}^{o} \lambda_i \cdot P^{(i)}$ be a linear combination of the matrices $P^{(i)}$ and let $P^{(k)}$ (for some $k \in \{1, \ldots, o\}$) be invertible. Then with probability not less than q^{o-v}, the map $(P^{(k)})^{-1} \cdot W$ has a non trivial invariant subspace which is also a subspace of $\mathcal{T}^{-1}(\mathcal{O})$.*

Proof. See [33], Theorem 4.2. □

We can therefore use Algorithm 3.2 to find $\mathcal{T}^{-1}(\mathcal{O})$.

The following Lemma can be used to test if a subspace H computed by Algorithm 3.2 is indeed a subspace of $\mathcal{T}^{-1}(\mathcal{O})$.

Lemma 3.2. *If $H \subset \mathcal{T}^{-1}(\mathcal{O})$, then, for every $\mathbf{x}, \mathbf{y} \in H$ and every $i = 1, \ldots, o$ we have*

$$\mathbf{x}^T \cdot P^{(i)} \cdot \mathbf{y} = 0.$$

Proof. See [33], Lemma 3. □

Algorithm 3.2 UOV attack

Input: UOV public key (given as matrices $P^{(1)}, \ldots, P^{(o)}$)
Output: Basis of $\mathcal{T}^{-1}(\mathcal{O})$

1: **repeat**
2: For a random vector $\lambda \in \mathbb{F}^o$ compute $W = \sum_{i=1}^{o} \lambda_i \cdot P^{(i)}$.
3: Compute $\hat{W} = (P^{(k)})^{-1} \cdot W$ for an invertible matrix $P^{(k)}$.
4: Compute the minimal invariant subspaces of $\hat{W}$.
5: For each minimal invariant subspace of $\hat{W}$, test if it is also a subspace of $\mathcal{T}^{-1}(\mathcal{O})$. For this step we can use the test described in Lemma 3.2.
6: **until** o linear independent vectors $v_1, \ldots, v_o \in \mathcal{T}^{-1}(\mathcal{O})$ are found.
7: **return** $v_1, \ldots, v_o$

After having performed q^{v-o-1} runs of the loop, Algorithm 3.2 has found with high probability at least one non zero vector in $\mathcal{T}^{-1}(\mathcal{O})$. We continue this process until we find o linear independent vectors in $\mathcal{T}^{-1}(\mathcal{O})$. By doing so, we can reconstruct the matrix M_T and thus find an equivalent private key which can be used by an attacker to generate signatures in the same way as a legitimate user.
The complexity of the whole process can be estimated by

$$\text{complexity}_{\text{UOV attack}}(q, o, v) = q^{v-o-1} \cdot o^4. \tag{3.15}$$

3.4 The Rainbow Signature Scheme

In [16] J. Ding and D. Schmidt proposed a signature scheme called Rainbow, which is based on the idea of Oil and Vinegar, but reduces both key sizes and signature lengths.

Let $\mathbb{F}$ be a finite field with q elements and S be the set $\{1, \ldots, n\}$. Let $v_1, \ldots, v_{u+1}$ be integers such that $0 < v_1 < v_2 < \ldots < v_u < v_{u+1} = n$ and define the sets of integers $S_i = \{1, \ldots, v_i\}$ for $i = 1, \ldots, u$. We set $o_i = v_{i+1} - v_i$ and $O_i = \{v_i + 1, \ldots, v_{i+1}\}$ $(i = 1, \ldots, u)$. The number of elements in S_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
$f^{(v_1+1)}, \ldots, f^{(n)} \in \mathbb{F}[x_1, \ldots, x_n]$ of the form

$$f^{(k)}(\mathbf{x}) = \sum_{i,j \in S_\ell,\ i \le j} \alpha_{ij}^{(k)} x_i x_j + \sum_{i \in O_\ell,\ j \in S_\ell} \beta_{ij}^{(k)} x_i x_j + \sum_{i \in S_\ell \cup O_\ell} \gamma_i^{(k)} x_i + \eta^{(k)} \ (k = v_1+1, \ldots, n), \tag{3.16}$$

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 $x_i \in S_\ell$ 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 Section 3.1.
The different levels of Oil and Vinegar in the map $\mathcal{F}$ are called *Rainbow layers*.

The map $\mathcal{F}(\mathbf{x}) = (f^{(v_1+1)}(\mathbf{x}), \ldots, f^{(n)}(\mathbf{x}))$ can be inverted as follows. First, we choose the values of the variables $x_1, \ldots, x_{v_1}$ at random and substitute them into the polynomials $f^{(v_1+1)}, \ldots, f^{(n)}$. 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 $x_{v_1+1}, \ldots, x_{v_2}$, 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 get values for all the variables x_i $(i = 1, \ldots, n)$. [2]

[2] It may happen, that one of the linear systems does not have a solution. If so, one has to choose other values of $x_1, \ldots x_{v_1}$ and try again. However, the probability of this is very small. Therefore, in most cases, one gets a

To hide the structure of $\mathcal{F}$ in the public key one composes it with two affine invertible maps $\mathcal{S} = (M_S, c_S) : \mathbb{F}^m \to \mathbb{F}^m$ and $\mathcal{T} = (M_T, c_T) : \mathbb{F}^n \to \mathbb{F}^n$.
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 To sign a document d, we use a hash function $\mathcal{H} : \{0,1\}^\star \to \mathbb{F}^m$ to compute the value $\mathbf{h} = \mathcal{H}(d) \in \mathbb{F}^m$. Then we compute recursively $\mathbf{x} = \mathcal{S}^{-1}(\mathbf{h})$, $\mathbf{y} = \mathcal{F}^{-1}(\mathbf{x})$ 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}) preimage of $\mathbf{x}$ under the central map $\mathcal{F}$.

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

The size of the public key is

$$\text{size}_{\text{pk Rainbow}} = m \cdot \frac{(n+1)\cdot(n+2)}{2} \tag{3.17}$$

field elements, the size of the private key is

$$\text{size}_{\text{sk Rainbow}} = m\cdot(m+1) + n\cdot(n+1) + \sum_{i=1}^{u} o_l \cdot \left(\frac{v_l \cdot (v_l+1)}{2} + v_l \cdot o_l + v_{l+1} + 1 \right) \tag{3.18}$$

field elements. Recommended parameters for the Rainbow signature scheme can be found in Chapter 6. For the remainder of this chapter we restrict for simplicity to Rainbow schemes with two layers.

3.5 Equivalent Keys for Rainbow

Let $(\mathcal{S}, \mathcal{F}, \mathcal{T})$ be a Rainbow private key. In this section we want to find the set $EQ_{(\mathcal{S},\mathcal{F},\mathcal{T})}$ of all private keys equivalent to $(\mathcal{S}, \mathcal{F}, \mathcal{T})$. To do this, we follow the approach of Section 2.4 and [59], i.e. we introduce two additional affine maps $\Sigma : \mathbb{F}^m \to \mathbb{F}^m$ and $\Omega : \mathbb{F}^n \to \mathbb{F}^n$ and set

$$\mathcal{P} = \underbrace{\mathcal{S} \circ \Sigma^{-1}}_{\mathcal{S}'} \circ \underbrace{\Sigma \circ \mathcal{F} \circ \Omega}_{\mathcal{F}'} \circ \underbrace{\Omega^{-1} \circ \mathcal{T}}_{\mathcal{T}'}. \tag{3.19}$$

The following theorem answers the question, for which affine maps Σ and Ω the map $\mathcal{F}' = \Sigma \circ \mathcal{F} \circ \Omega$ has the form of a Rainbow central map.

Theorem 3.5. *Let $(\mathcal{S}, \mathcal{F}, \mathcal{T})$ be a Rainbow private key and let $\Sigma : \mathbb{F}^m \to \mathbb{F}^m$ and $\Omega : \mathbb{F}^n \to \mathbb{F}^n$ be affine maps, whose linear parts have the form*

$$\Sigma_{\text{lin}} = \begin{pmatrix} \Sigma^{(1)}_{o_1 \times o_1} & 0_{o_1 \times o_2} \\ \Sigma^{(3)}_{o_2 \times o_1} & \Sigma^{(4)}_{o_2 \times o_2} \end{pmatrix}, \quad \Omega_{\text{lin}} = \begin{pmatrix} \Omega^{(1)}_{v_1 \times v_1} & 0_{v_1 \times o_1} & 0_{v_1 \times o_2} \\ \Omega^{(4)}_{o_1 \times v_1} & \Omega^{(5)}_{o_1 \times o_1} & 0_{o_1 \times o_2} \\ \Omega^{(7)}_{o2 \times v_1} & \Omega^{(8)}_{o_2 \times o_1} & \Omega^{(9)}_{o_2 \times o_2} \end{pmatrix}. \tag{3.20}$$

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_{lin}^{-1} and Ω_{lin}^{-1} have the same form as Σ_{lin} and Ω_{lin} respectively.

valid signature at the first try.

Proof. see [60], Theorem 4.14. □

Corollary 3.2. *Let $(\mathcal{S}, \mathcal{F}, \mathcal{T})$ be a Rainbow private key over a field with q elements. Then the set $EQ_{(\mathcal{S},\mathcal{F},\mathcal{T})}$ of private keys equivalent to $(\mathcal{S}, \mathcal{F}, \mathcal{T})$ has the size*

$$|EQ_{(\mathcal{S},\mathcal{F},\mathcal{T})}| = q^{m+n} \cdot q^{v_1\cdot(o_1+o_2)+2\cdot o_1\cdot o_2} \cdot \prod_{i=0}^{v_1-1}(q^{v_1}-q^i) \cdot \left(\prod_{i=0}^{o_1-1}(q^{o_1}-q^i)\right)^2 \cdot \left(\prod_{i=0}^{o_2-1}(q^{o_2}-q^i)\right)^2. \tag{3.21}$$

Proof. The number of elements in $EQ_{(\mathcal{S},\mathcal{F},\mathcal{T})}$ is the same as the number of possible choices of the maps Σ and Ω. The first factor in equation (3.21) represents the number of possible choices of the constant parts of the maps Σ and Ω. The remaining factors stand for the possible choices of the matrices Σ_{lin} and Ω_{lin} of Theorem 3.5. Note that each of the matrices $\Sigma^{(1)}$, $\Sigma^{(4)}$, $\Omega^{(1)}$, $\Omega^{(5)}$ and $\Omega^{(9)}$ has to be invertible. □

Remark 3.4. *Corollary 3.2 states that for each Rainbow public key $\mathcal{P}$, there exist $q^{m\cdot(v_1+1)+n+2\cdot o_1\cdot o_2} \cdot \prod_{i=0}^{v_1-1}(q^{v_1}-q^i) \cdot \left(\prod_{i=0}^{o_1-1}(q^{o_1}-q^i)\right)^2 \cdot \left(\prod_{i=0}^{o_2-1}(q^{o_2}-q^i)\right)^2$ Rainbow private keys $(\mathcal{S}, \mathcal{F}, \mathcal{T})$ such that $\mathcal{S} \circ \mathcal{F} \circ \mathcal{T} = \mathcal{P}$.*

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 3.6. *Let $\mathcal{P}$ be a Rainbow public key. Then, with overwhelming probability, there exists a Rainbow private key $(\widetilde{\mathcal{S}}, \widetilde{\mathcal{F}}, \widetilde{\mathcal{T}})$ with $\widetilde{\mathcal{S}} \circ \widetilde{\mathcal{F}} \circ \widetilde{\mathcal{T}} = \mathcal{P}$ such that*

$$M_{\tilde{S}} = \begin{pmatrix} 1_{o_1\times o_1} & S'_{o_1\times o_2} \\ 0_{o_2\times o_1} & 1_{o_2\times o_2} \end{pmatrix}, \quad M_{\tilde{T}} = \begin{pmatrix} 1_{v_1\times v_1} & T'^{(1)}_{v_1\times o_1} & T'^{(2)}_{v_1\times o_2} \\ 0_{o_1\times v_1} & 1_{o_1\times o_1} & T'^{(3)}_{o_1\times o_2} \\ 0_{o_2\times v_1} & 0_{o_2\times o_1} & 1_{o_2\times o_2} \end{pmatrix}. \tag{3.22}$$

Proof. 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 $\mathcal{S} \circ \mathcal{F} \circ \mathcal{T} = \mathcal{P}$. In the following we compute affine maps Σ and Ω which transform this private key into an equivalent private key of form (3.22), i. e. $\widetilde{\mathcal{S}} = \mathcal{S} \circ \Sigma^{-1}$ and $\widetilde{\mathcal{T}} = \Omega^{-1} \circ \mathcal{T}$ are of the form shown in the theorem. Let $S = \begin{pmatrix} S_1 & S_2 \\ S_3 & S_4 \end{pmatrix}$ and Σ_{lin} be of the form as shown in equation (3.20) .
If S_4 is invertible[3], we get from $\mathcal{S} = \widetilde{\mathcal{S}} \circ \Sigma$

- $\Sigma_1 = S_1 - S_2 \cdot S_4^{-1} \cdot S_3$
- $\Sigma_3 = S_3$ and
- $\Sigma_4 = S_4$.

The matrix S' of equation (3.22) is given as

$$S' = S_2 \cdot S_4^{-1}. \tag{3.23}$$

[3]If S_4 is not invertible, we can switch the rows and columns of S by renumbering the equations of $\mathcal{F}$ until we get an invertible matrix.

Let $T = \begin{pmatrix} T_1 & T_2 & T_3 \\ T_4 & T_5 & T_6 \\ T_7 & T_8 & T_9 \end{pmatrix}$ and Ω_{lin} be of the form (3.20). If T_1 and $T_5 - T_4 \cdot T_1^{-1} \cdot T_2$ are invertible[4], we get from $\mathcal{T} = \Omega \circ \widetilde{\mathcal{T}}$

- $\Omega_1 = T_1$
- $\Omega_4 = T_4$
- $\Omega_7 = T_7$
- $T'^{(1)} = T_1^{-1} \cdot T_2$
- $T'^{(2)} = T_1^{-1} \cdot T_3$
- $\Omega_5 = T_5 - T_4 \cdot T_1^{-1} \cdot T_2$
- $T'^{(3)} = \Omega_5^{-1} \cdot (T_6 - T_4 \cdot T_1^{-1} \cdot T_3)$
- $\Omega_8 = T_8 - T_7 \cdot T_1^{-1} \cdot T_2$ and
- $\Omega_9 = T_9 - T_7 \cdot T_1^{-1} \cdot T_3 - \Omega_8 \cdot T'^{(3)}$.

□

The matrix $M_{\tilde{T}}$ of equation (3.22) can be written as a product of matrices $T_{v_1+1} \cdot \ldots \cdot T_n$ such that

$$T_i = \begin{pmatrix} 1 & & 0 & 0 & t'_{1i} & 0 \\ & \ddots & & \vdots & \vdots & \vdots \\ 0 & & 1 & 0 & t'_{v_2 i} & 0 \\ 0 & \ldots & 0 & 1 & 0 & 0 \\ \vdots & & \vdots & & \ddots & \\ 0 & \ldots & 0 & 0 & & 1 \end{pmatrix} \quad (i = v_2 + 1, \ldots, n) \text{ and}$$

$$T_i = \begin{pmatrix} 1 & & 0 & 0 & t'_{1i} & 0 \\ & \ddots & & \vdots & \vdots & \vdots \\ 0 & & 1 & 0 & t'_{v_1 i} & 0 \\ 0 & \ldots & 0 & 1 & 0 & 0 \\ \vdots & & \vdots & & \ddots & \\ 0 & \ldots & 0 & 0 & & 1 \end{pmatrix} \quad (i = v_1 + 1, \ldots, v_2). \tag{3.24}$$

Besides the 1's on the main diagonal, the matrices T_i $(i = v_1 + 1, \ldots, n)$ contain exactly the same non zero elements as the matrix $M_{\tilde{T}}$.
By inversion we get $M_{\tilde{T}}^{-1} = T_n^{-1} \cdot \ldots \cdot T_{v_1+1}^{-1}$. Note that the matrices T_i^{-1} have the same structure as the matrices T_i $(i = v_1 + 1, \ldots, n)$.

[4]Again it is possible to switch rows and columns of T by renumbering the variables.

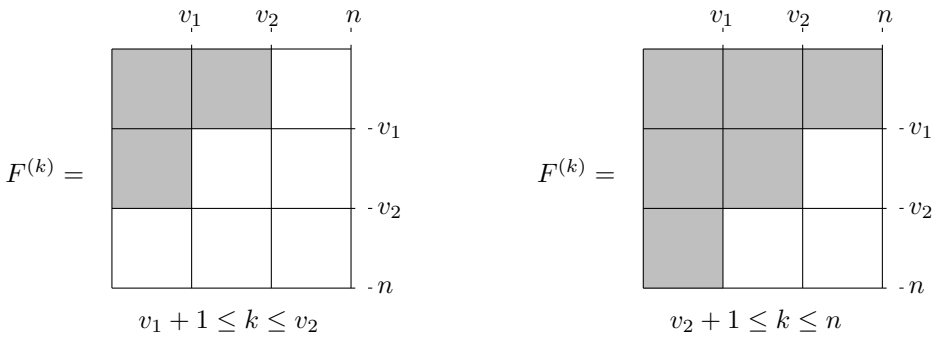


Figure 3.1: Layout of the matrices $F^{(k)}$. The white parts of the matrices are fixed to zero.

3.6 Attacks against Rainbow

Let $F^{(k)}$ $(v_1+1 \leq k \leq n)$ be the symmetric matrix associated to the homogeneous quadratic part of the k-th component of the central map $\mathcal{F}$ (see equation (2.14)). Due to the special structure of $\mathcal{F}$ the matrices $F^{(k)}$ have the form shown in Figure 3.1. Analogously, we denote the symmetric matrix representing the homogeneous quadratic part of the k-th component of $\mathcal{P}$ by $P^{(k)}$ $(v_1+1 \leq k \leq n)$. With this notation we get

$$P^{(i+v_1)} = \sum_{j=1}^{m} s_{ij}\ M_T^T \cdot F^{(j+v_1)} \cdot M_T \text{ and} \tag{3.25}$$

$$F^{(i+v_1)} = \sum_{j=1}^{m} \tilde{s}_{ij}\ (M_T^{-1})^T \cdot P^{(j+v_1)} \cdot M_T^{-1}\ (i = 1, \ldots, m) \tag{3.26}$$

where s_{ij} and $\tilde{s}_{ij}$ $(i, j = 1, \ldots, m)$ are the elements of the matrices M_S and M_S^{-1} respectively.

3.6.1 Direct Attack

The most straightforward method to attack the Rainbow signature scheme is to try to solve the public system $\mathcal{P}(\mathbf{x}) = \mathbf{h}$ (signature forgery attack). Since the public systems of Rainbow are underdetermined, one usually fixes/guesses some of the variables before applying an algorithm such as XL or a Gröbner Basis method such as F_4/F_5 (see Section 2.5).

3.6.2 MinRank Attack

The goal of the MinRank attack [30, 6] 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 given by v_2). Such a matrix corresponds to a linear combination of the o_1 matrices $M_T^T \cdot F^{(k)} \cdot M_T$ $(k \in O_1)$ representing the central polynomials of the first Rainbow layer. Therefore, we have to solve the following problem.

MinRank-Problem: Given a set of m $n \times n$ matrices $P^{(v_1+1)}, \ldots, P^{(n)}$, find a linear combination $H = \sum_{i=v_1+1}^{n} \lambda_i P^{(i)}$ which has rank $\leq v_2$.

This problem can be solved as shown in Algorithm 3.3.

By finding o_1 linear independent matrices C of this form, we can extract the first Rainbow layer. This step costs approximately $o_1 \cdot q^{v_1+1}$ operations, where q is the cardinality of the underlying field.

The remaining Rainbow layers can now be extracted by a similar technique. However, since

Algorithm 3.3 MinRank attack

Input: matrices $P^{(v_1+1)}, \ldots, P^{(n)}$
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 $P = \sum_{i=v_1+1}^{n} \lambda_i \cdot P^{(i)}$.
3: **if** Rank $(P) > 1$ and Rank$(P) < n$ **then**
4: Choose randomly a vector γ from Ker(P).
5: $C \leftarrow \sum_{i=v_1+1}^{n} \gamma_i \cdot P^{(i)}$
6: **end if**
7: **until** Rank $(C) \leq v_2$
8: **return** C

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

$$\text{complexity}_{\text{MinRank}}(q, v_1, o_1) = o_1 \cdot q^{v_1+1}. \tag{3.27}$$

For the following we define the sets $\mathcal{O}_i$ as $\mathcal{O}_i = \{x \in \mathbb{F}^n : x_1 = \ldots = x_{v_i} = 0\}$ $(i = 1, \ldots, u)$.

3.6.3 HighRank Attack

The HighRank attack as proposed by Coppersmith et al. in [13] can be seen as the counterpart of the MinRank attack.
As the matrices C found during the MinRank attack are linear combinations of the matrices $M_T^T \cdot F^{(k)} \cdot M_T$ $(k \in O_1)$, we have $C(x) = 0$ for all $x \in \mathcal{T}^{-1}(\mathcal{O}_1)$ and therefore $\mathcal{T}^{-1}(\mathcal{O}_1) \subset \ker(C)$. Hence the MinRank attack can be seen as an attack looking for this space $\mathcal{T}^{-1}(\mathcal{O}_1)$ or as an attack that finds a large kernel shared by a small number of linear combinations $C = \sum_{k=v_1+1}^{n} \lambda_k \cdot P^{(k)}$. In the HighRank attack, we turn this around. The goal of this attack is to find a small kernel $(\mathcal{T}^{-1}(\mathcal{O}_u))$ shared by a large number of linear combinations $\sum_{k=v_1+1}^{n} \lambda_k \cdot P^{(k)}$.

The HighRank attack is based on the following observation. The variables $x_{v_u+1}, \ldots, x_n$ appear only in the quadratic cross terms of the central polynomials $f^{(v_u+1)}, \ldots, f^{(n)}$ of the last Rainbow layer. Therefore we get $\mathcal{O}_u \subset \ker \sum_{k=v_1+1}^{v_u} \alpha_k \cdot F^{(k)}$ for arbitrary $\alpha \in \mathbb{F}^k$ which means that $\mathcal{T}^{-1}(\mathcal{O}_u)$ lies in the kernel of certain linear combinations of the matrices $P^{(k)}$ $(k = 1, \ldots, n)$. Algorithm 3.4 shows the functioning of this attack to find the space $\mathcal{T}^{-1}(\mathcal{O}_u)$.

Algorithm 3.4 HighRank attack

Input: matrices $P^{(v_1+1)}, \ldots, P^{(n)}$
Output: $\mathcal{T}^{-1}(\mathcal{O}_u)$

1: Form an arbitrary linear combination $H = \sum_{k=v_1+1}^{n} \lambda_k \cdot P^{(k)}$. Find $V = \ker H$.
2: If $\dim V \geq 1$, set $\left(\sum_{k=v_1+1}^{n} \lambda_k P^{(k)}\right) V = 0$. 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** $\mathcal{T}^{-1}(\mathcal{O}_u)$

The complexity of this step can be estimated by $q^{o_u} \cdot \frac{n^3}{6}$.

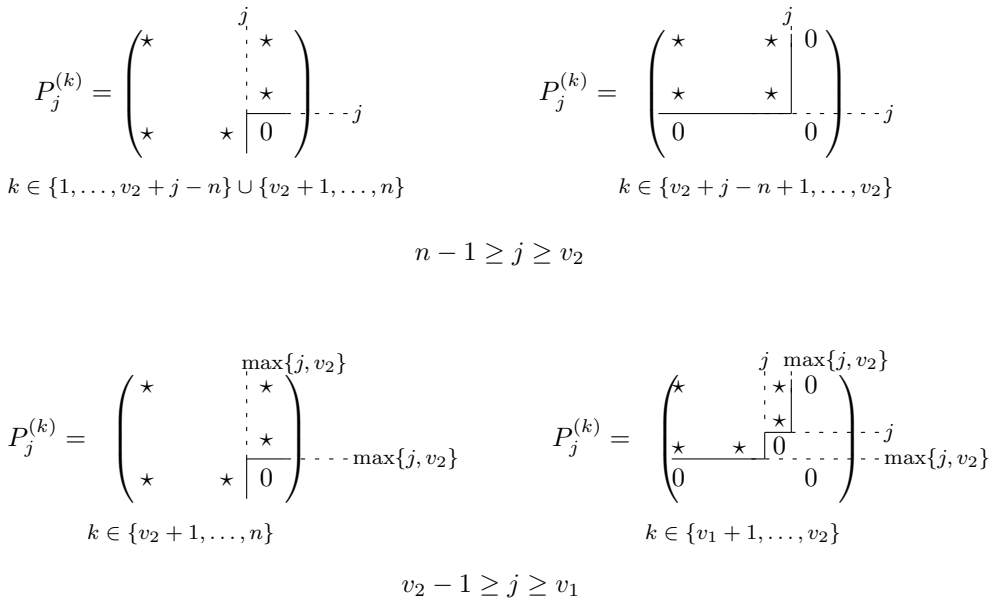


Figure 3.2: Structure of the matrices $P_j^{(i)}$

By studying the subspaces of $\mathcal{T}^{-1}(\mathcal{O}_u)$ we can find bigger kernels (which correspond to $\mathcal{T}^{-1}(\mathcal{O}_i)$ $(i = u - 1, \ldots, 1)$). Since the complexity of this step can be neglected, we get

$$\text{complexity}_{\text{HighRank}}(q, o_u, n) = q^{o_u} \cdot \frac{n^3}{6}. \tag{3.28}$$

3.6.4 Rainbow-Band-Separation Attack

The Rainbow-Band-Separation attack [18] can be seen as an extension of the UOV-Reconciliation attack (see Subsection 3.3.2) to Rainbow and uses the layer structure of this scheme. The 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 M_S and M_T are of the form of equation (3.22). We write M_T^{-1} as a product of matrices $\tilde{T}_n \cdot \ldots \cdot \tilde{T}_{v_1+1}$ as shown in equation (3.24) and get

$$F^{(k)} = \sum_{l=1}^{m} \tilde{s}_{kl} \ \left(\tilde{T}_{v_1+1}^T \cdot \ldots \cdot \tilde{T}_n^T \cdot P^{(l)} \cdot \tilde{T}_n \cdot \ldots \cdot \tilde{T}_{v_1+1} \right) \tag{3.29}$$

(see equation 3.26). Here, $\tilde{s}_{kl}$ $(k, l = 1, \ldots, m)$ denote the elements of the matrix M_S^{-1}.

As for the UOV-Reconciliation attack (see Subsection 3.3.2), the goal of the Rainbow-Band-Separation attack is to find for every $k = v_1 + 1, \ldots, n$ a sequence of matrices $P_{n-1}^{(k)}, \ldots, P_{v_1}^{(k)}$, such that $P_{v_1}^{(k)}$ has the form of $F^{(k)}$. The matrices $P_{v_1}^{(k)}$ $(k = v_1 + 1, \ldots, n)$ (together with the matrices $\tilde{T}_i$ $(i = v_1 + 1, \ldots, n)$ and the elements $\tilde{s}_{kl}$) yield therefore an equivalent Rainbow public key which can be used by the attacker to sign messages in the same way as a legitimate user.
In more detail, the matrices $P_j^{(k)}$ the attacker wants to find have the form shown in Figure 3.2.

The transition from $P_j^{(k)}$ to $P_{j-1}^{(k)}$ looks like

$$\begin{array}{ll} P_{j-1}^{(k)} = \tilde{T}_j^T \cdot P_j^{(k)} \cdot \tilde{T}_j & j \in \{v_1+1,\ldots,v_2\} \text{ or} \\ & j \in \{v_2+1,\ldots,n\},\ k \in \{v_1+1,\ldots n\} \setminus \{j\} \\ P_{j-1}^{(k)} = \sum_{l=v_2+1}^{n} \tilde{s}_{kl} \left(\tilde{T}_j^T \cdot P_j^{(l)} \cdot \tilde{T}_j \right) & j \in \{v_2+1,\ldots,n\},\ k = j \end{array} \tag{3.30}$$

As in the case of the UOV-Reconciliation attack, every zero position in the matrices $P_{j-1}^{(k)}$ yields one (usually quadratic) equation in the unknown elements of $\tilde{T}_j$ and $\tilde{s}_{jl}$. Altogether, in the j-th step, we get $(n-j+1)\cdot(m+n-1)$ (mostly quadratic) equations in n variables from (3.30) $(j = n,\ldots,v_1+1)$ [5]. By solving this system, the attacker is able to find the matrix $\tilde{T}_j$ and the elements $\tilde{s}_{kl}$ and therefore can compute the matrices $P_{j-1}^{(k)}$ $(k = v_1+1,\ldots,n)$.
To find the matrices $P_{v_1}^{(k)}$ and therefore an equivalent private key, the attacker has to solve m of these systems.
The complexity of the Rainbow-Band-Separation attack is mainly determined by the complexity of solving the first system 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 $\tilde{s}_{nk}$ $(k = v_2+1,\ldots,n)$).

In Chapter 6 we carry out a number of experiments to estimate the complexity of solving such systems.

3.6.5 UOV and UOV-Reconciliation Attack

For these two attacks Rainbow is seen as a UOV scheme: We define a new map $\tilde{\mathcal{F}} = \mathcal{S} \circ \mathcal{F}$ and get $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T} = \tilde{\mathcal{F}} \circ \mathcal{T}$. Note that $\tilde{\mathcal{F}}$ has the form of a UOV central map with v_u Vinegar variables and o_u Oil variables [6]. Therefore we can use all specific attacks against UOV against Rainbow, too.

For the complexity of the UOV attack we get (see equation (3.15))

$$\text{complexity}_{\text{UOV attack Rainbow}}(q, v_u, o_u, n) = q^{v_u - o_u - 1} \cdot o_u^4 = q^{n - 2\cdot o_u - 1} \cdot o_u^4, \tag{3.31}$$

the complexity of the UOV-Reconciliation attack is given mainly by the complexity of solving a system of m quadratic equations in v_u variables.

[5] For $j \in \{v_1+1,\ldots,v_2\}$, the number of variables reduces to v_1.
[6] However, unlike a UOV scheme, we have still m equations in the system.

Part I

Selecting Parameters for Multivariate Schemes

The question, which parameters one has to choose for a cryptographic scheme to meet given levels of security is one of the central problems of cryptography and is important both for the practitioner as for the theorist. In this first part of the dissertation we address this question for multivariate cryptosystems in general and in particular for the Rainbow signature scheme. We start our analysis with the security model of Lenstra and Verheul. This model takes as input a year y and computes, using parameters like the development of computational power and the budget of an attacker, a security level cryptographic schemes must reach to be thought secure in the year y. Based on this model we then analyze generic attacks against the MQ-Problem (see Section 2.5) as well as specialized attacks against Rainbow (see Section 3.6) and provide parameters which guarantee the security of this scheme for now and the near future.

In the first chapter of this part (Chapter 4) we introduce the security model of Lenstra and Verheul and describe our computing environment which will be used for the experiments in the following two chapters. In Chapter 5 we analyze the complexity of the MQ-Problem both theoretically and with computer experiments. We combine the results with our security model to give lower bounds for the parameter choice of multivariate schemes. Furthermore we give parameters for the UOV signature scheme for different levels of security. Finally, in Chapter 6, we analyze known attacks against the Rainbow signature scheme, which is one of the best studied and most promising multivariate schemes and propose parameter sets for this scheme for the years 2012 - 2050. Hereby we use a strategy which allows us to optimize the parameters in regard of both public and private key size.

Chapter 4

The model

In [36] Lenstra and Verheul developed a security model, which they used to find appropriate parameter sets for symmetric cryptography and some asymmetric schemes. The main points of their model are:

1. Security margin: a definition of the term "adequate security".

2. Computing environment: the expected change in computational resources available to attackers.

3. Cryptanalytic development: the expected development in cryptanalysis.

In the following we take a closer look at these items.

Security margin

To decide, whether a given instance of a cryptographic scheme offers an adequate level of security, one has to define the term "adequate security". The authors of [36] define it by the security offered by the Data Encryption Standard (DES) in 1982. That is, in 1982 a symmetric key size of 56 bit provided an adequate level of security. We follow this definition.

Computing environment

Here the authors of [36] use a slightly modified version of Moore's law, which states that the amount of computing power and random access memory one gets for 1 dollar doubles every t months. Our default setting of t is 18, see [36].

Remark 4.1. *In the last years there has been an active discussion whether Moore's law will hold in the future. The reason for these concerns is that an increase in processor performance goes hand in hand with a diminution of the electronic circuits on the processor. This will not be possible forever since with smaller circuits the laws of quantum mechanics play an increasing role. However, part of this trend can be reversed by multicore computing, which means using more than one processor on the same CPU. We believe that this process will continue and the number of processors on one CPU will increase even further. Multicore computing especially speeds up paralellizable programs, as they are used for many cryptanalytic purposes. Furthermore, with better production techniques, the cost of a processor will decrease. So, we believe that the modified version of Moore's law used in [36] will hold over the next 50 years.*

Another thing we have to take into account in this context is the budget of an attacker, which might increase over time. The parameter $b > 0$ is defined as the number of years it takes on average for an expected twofold increase of budget. Statistical data says, that the world wide economic performance (in today's prices) doubles about every ten years. So our default setting for b is 10.

Cryptanalytic development

It is impossible to say what cryptanalytic developments will take place in the future, or even what developments have already taken place, but have not been published yet. In a relatively modern field like multivariate public key cryptography this is especially difficult. Furthermore, many multivariate schemes thought to be secure were completely broken by new attacks (e.g. SFLASH [48]). However, we can assume that, at least for schemes whose security is only based on the MQ-Problem (e.g. the identification scheme of [52] and the stream cipher QUAD [2]), the pace of future cryptanalytic findings and their impact will not vary dramatically compared to the pace of findings about classical public key cryptosystems in the last decades. We assume the same for the UOV and Rainbow signature schemes (see Chapter 3), since despite of active research in the last 20 years, there has not been a break through in cryptanalysis, at least for schemes with carefully chosen parameters. In fact, similarly to the case of classical public key schemes like RSA, known attacks against UOV and Rainbow have been stepwise improved over time.

We define the number $r > 0$ to be the number of months it is expected to take on average for cryptanalytic techniques affecting multivariate public key cryptosystems to become twice as effective. Under the assumption, that the pace of cryptanalytic findings in the area of multivariate cryptography will not vary dramatically from those in the field of classical public key cryptosystems, our default setting for r is $r = 18$.

year	symmetric key size	security level with todays algorithms
2012	79	79
2015	82	84
2020	86	91
2030	93	105
2040	101	120
2050	109	134

Table 4.1: Required security levels until 2050

Given that breaking DES with a key length of 56 bit was infeasible to do in the year 1982, we get the key size a secure symmetric scheme must have in the year $y \geq 1982$ by the formula

$$\text{symmetric key size}(y) = 56 + 12 \cdot \frac{y - 1982}{t} + \frac{y - 1982}{b} \text{ bit } \quad (y \geq 1982). \tag{4.1}$$

With our default settings we get

$$\text{symmetric key size}(y) = \frac{23}{30} \cdot y - 1463.5 \text{ bit } \quad (y \geq 1982). \tag{4.2}$$

So far, we have not considered the possible advances in cryptanalysis. To cover these, we have to adapt the above formula slightly. So, a multivariate cryptosystem, which shall be secure in the year y, must reach (with todays algorithms) the security level of

$$\begin{aligned} \text{security level}(y) &= \text{symmetric key size}(y) + 12 \cdot \frac{y - 2012}{r} \\ &\overset{r=18}{=} \frac{43}{30} \cdot y - 2804.9 \text{ bit } \quad (y \geq 2012). \end{aligned} \tag{4.3}$$

Table 4.1 shows the symmetric key size and the security level a (multivariate) scheme must reach to be thought secure in the year y. As the table shows, a symmetric scheme, which shall be secure in the year 2050, must have a key size of at least 109 bit. Therefore, to be secure in the year 2050, a multivariate scheme must reach the same level of security (with the algorithms available in 2050). But, with the algorithms available today, it must reach a higher security level of 134 bit. In the following, we denote the symmetric key size required in the year y by $\ell(y)$ and the security level needed for multivariate schemes by $\bar{\ell}(y)$.

4.1 Our Computing Environment

To estimate the complexity of attacks against multivariate schemes, we carried out a large number of computer experiments. For these experiments (see the next two chapters) we used a server with 128 GB RAM and 16 AMD Opteron processors with 2.8 GHz. Systems of multivariate quadratic polynomials are solved with MAGMA [8] version 2.13-10, which contains an efficient implementation of Faugère's F_4 algorithm [23], on a single processor.
To find a relation between the running time of MAGMA and the complexity of a system we use a random system of 37 equations in 22 variables over GF(256). The bit complexity of solving this system with F_4 is about 46.7 [18]. Since MAGMA needed about $1.43 \cdot 10^5$ seconds to solve this system, we get the following relation between the running time of our experiments and the bit complexity of solving a multivariate system.

$$\text{bit complexity} = \lg\left(\text{ running time} \cdot \frac{2^{46.7}}{1,43 \cdot 10^5}\right) \approx \lg(\text{running time}) + 29.6. \tag{4.4}$$

The above formula enables us to compare the results of our computer experiments with the theoretical results obtained by the analysis of the HybridF_5 algorithm (see Section 5.1).

Chapter 5

Complexity of the MQ-Problem

In this chapter we study the complexity of the MQ-Problem and answer the question, how many equations are needed to defend a multivariate scheme against direct attacks[1]. We do this both by a theoretical analysis of the HybridF_5 algorithm (see [5] and Subsection 2.5.3), which is currently the fastest algorithm to solve systems of multivariate nonlinear equations and experiments with MAGMA, which contains an efficient implementation of Faugère's F_4 algorithm. Note that, for multivariate public key schemes, the numbers presented in this section are necessarily lower bounds since we do not study key recovery attacks. Furthermore, for some multivariate schemes (e.g. HFE) Gröbner Basis algorithms can use the special structure of the public key to reduce the running time [39]. For schemes with highly underdetermined systems (e.g. UOV) one has to consider special attacks against underdetermined systems, too (see [56] and Subsection 2.5.4). Furthermore, since overdetermined systems are easier to solve, our analysis can not be applied to encryption schemes such as the SimpleMatrix scheme of [55].

5.1 Theoretical Analysis with HybridF_5

HybridF_5 [5] is currently the fastest algorithm to solve generic systems of nonlinear multivariate equations. The basic idea is to guess some of the variables to create overdetermined systems before applying Faugère's F_5 [24] algorithm. When doing so, one has to run the F_5 algorithm several times to find a solution of the original system. When guessing k variables over a finite field of q elements, this number is given by q^k.
The complexity of solving a system of m quadratic equations in n variables over a finite field with q elements by the HybridF_5 algorithm can be estimated by [5]

$$\text{complexity}_{\text{HF}_5}(q,m,n) = \min_{k\geq 0}\ q^k \cdot \mathcal{O}\ \left(m \cdot \binom{n-k+d_{\text{reg}}-1}{d_{\text{reg}}}\right)^{\omega}, \tag{5.1}$$

where, for random systems, the degree of regularity d_{reg} is given as the lowest integer D for which the coefficient of t^D in $\frac{(1-t^2)^m}{(1-t)^n}$ is less or equal to 0 and $2 < \omega \leq 3$ denotes the linear algebra constant of solving a linear system (see Subsection 2.5.3).
Unfortunately, the details of the F_5 algorithm are not publicly known, which makes it difficult to estimate the concrete running time the algorithm needs to solve a multivariate system. Nevertheless, to derive secure parameters for multivariate schemes from formula (5.1), we follow the approach of [56, P12]. We set $\omega = 2$ and get by

$$\text{lower bound}(q,m,n) = \min_{k\geq 0}\ q^k \cdot \left(m \cdot \binom{n-k+d_{\text{reg}}-1}{d_{\text{reg}}}\right)^2 \tag{5.2}$$

[1] In this chapter, we consider only determined systems. We therefore set $n = m$.

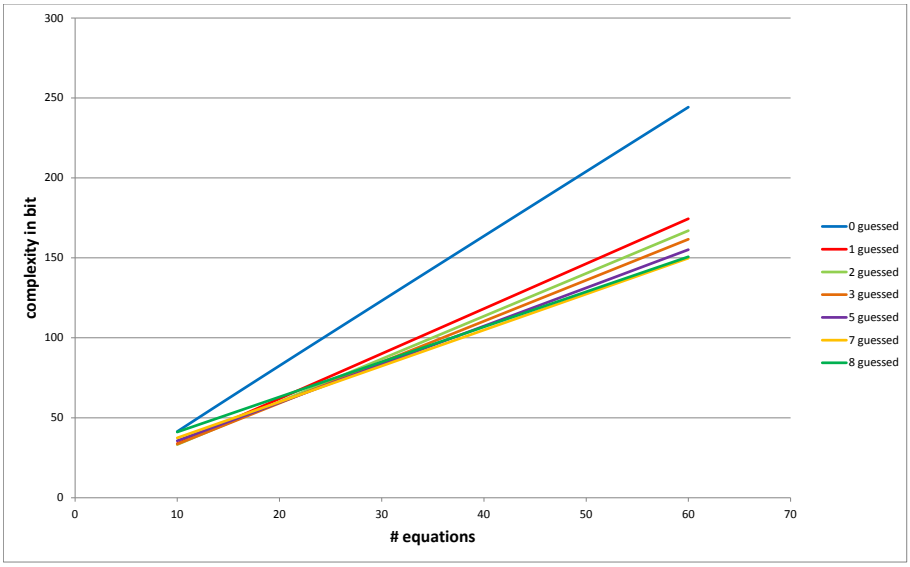


Figure 5.1: Complexity of solving determined random systems over GF(16) with HybridF_5

a lower bound of the complexity of the HybridF_5 algorithm. If this lower bound is larger than the security level, we can be sure that an attack against the multivariate quadratic system with the HybridF_5 algorithm is infeasible.
We used equation (5.2) to compute a lower bound of the complexity of solving determined random systems over GF(16), GF(31) and GF(256) with the HybridF_5 algorithm when guessing different numbers of variables. Figures 5.1 - 5.3 show the results.

A natural question in this context is, how many variables should be guessed before applying the F_5 algorithm, i.e. for which number $k \geq 0$ the complexity of the algorithm as shown by equation (5.2) gets minimal. Guessing variables to build overdetermined systems reduces the complexity of each run of the F_5 algorithm, but also increases the number of this runs. We find

- for determined systems over GF(16) it is the best strategy to guess 5 ($m \leq 29$), 6 ($30 \leq m \leq 40$) or 7 ($m \geq 41$) variables,
- for determined systems over GF(31) it is the best strategy to guess 2 ($m \leq 27$), 3 ($28 \leq m \leq 35$) or 4 ($m \geq 36$) variables and
- for determined systems over GF(256) it is the best strategy to guess 1 ($m \leq 32$) or 2 ($m \geq 33$) variables.

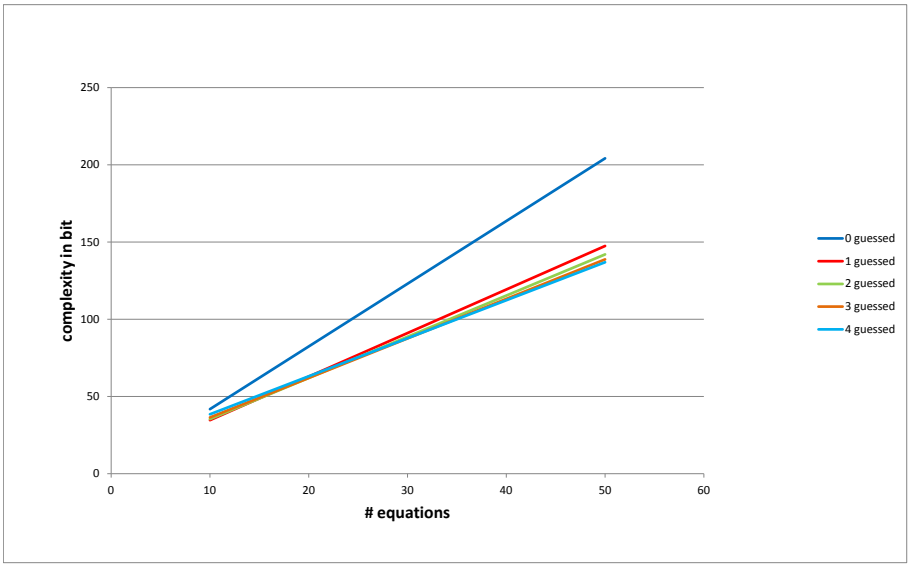


Figure 5.2: Complexity of solving determined random systems over GF(31) with HybridF_5

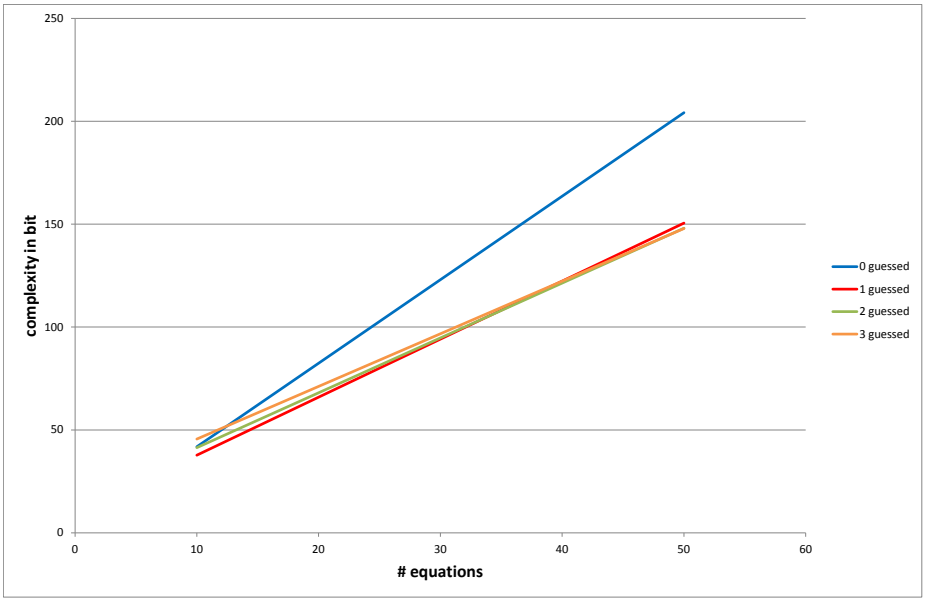


Figure 5.3: Complexity of solving determined random systems over GF(256) with HybridF_5

By applying the above strategy and evaluating equation (5.2) for $m \in \{20, \dots 60\}$, we find that the bit complexity of solving a determined random system of m multivariate quadratic equations using the HybridF_5 algorithm is roughly

$$2.30 \cdot m + 14.4 \tag{5.3}$$

for systems over GF(16),

$$2.50 \cdot m + 13.0 \tag{5.4}$$

for systems over GF(31) and

$$2.65 \cdot m + 12.1 \tag{5.5}$$

for systems over GF(256).

5.2 Experiments with MAGMA

Additionally to our theoretical analysis presented in the previous section we carried out a large number of computer experiments. For these experiments we used MAGMA v. 2.13-10 which contains an efficient implementation of Faugère's F_4 algorithm. Although MAGMA v. 2.13-10 was already released in 2008, it is still one of the fastest publicly available solvers for multivariate nonlinear systems and in fact often faster than newer versions of MAGMA. Before applying the MAGMA command `Variety`, we guessed some of the variables to create overdetermined systems. As we will see, this reduces the running time of the algorithm for a large class of parameters, even if we have to run the F_5 algorithm several times to find a solution of the original system.

Solving determined random systems over GF(16)

Table 5.1 shows the results of our MAGMA experiments with determined random systems over GF(16).

# equations	9	10	11	12	13	14	15	16
no guessing	4.6 s 30.5 MB	34.6 s 82.5 MB	4.2 m 460 MB	33.8 m 1,416 MB	273.4 m 7,041 MB	41.7 h 21,949 MB	- ooM[1]	-
1 guessed	1.7 s 9.4 MB	12.4 s 11.7 MB	2.1 m 21.3 MB	7.1 m 65.7 MB	98.9 m 201.7 MB	8.6 h 782 MB	86.9 h 1,750 MB	23.0 d 2,425 MB
2 guessed	**1.6 s** 8.9 MB	**9.8 s** 11.2 MB	**1.0 m** 19.3 MB	**5.5 m** 39.5 MB	**41.4 m** 73.9 MB	**4.8 h** 86.3 MB	**21.6 h** 120.1 MB	6.2 d 225.4 MB
3 guessed	3.0 s 8.2 MB	16.8 s 10.7 MB	1.5 m 15.3 MB	8.6 m 18.6 MB	47.5 m 29.7 MB	5.9 h 34.2 MB	25.7 h 43.6 MB	**4.5 d** 58.7 MB
4 guessed	6.2 s 8.5 MB	32.8 s 10.3 MB	2.9 m 11.7 MB	15.4 m 13.2 MB	80.7 m 16.4 MB	7.1 h 20.3 MB	47.9 h 29.8 MB	6.1 d 35.7 MB
5 guessed	19.4 s 8.3 MB	97.3 s 9.4 MB	8.1 m 11.3 MB	40.2 m 12.7 MB	199.3 m 14.5 MB	16.5 h 16.5 MB	82.5 h 21.3 MB	25.5 d 28.7 MB
6 guessed	69.0 s 8.1 MB	328.7 s 9.2 MB	26.1 m 10.7 MB	124.3 m 11.9 MB	592.0 m 14.3 MB	47.0 h 15.9 MB	223.8 h 20.7 MB	44.4 d 27.3 MB
7 guessed	169.1 s 8.2 MB	781.6 s 9.1 MB	60.2 m 10.5 MB	278.1 m 11.4 MB	1,285 m 13.9 MB	98.7 h 18.7 MB	457.4 h 26.9 MB	88.1 d 30.4 MB
8 guessed	786.0 s 8.2 MB	3,552 s 9.3 MB	269.2 m 10.4 MB	1,230 m 11.3 MB	5,538 m 13.6 MB	417.5 h 17.8 MB	1,889 h 25.3 MB	356.2 d 29.2 MB

[1] out of memory

Table 5.1: Running time of solving determined random systems over GF(16) with MAGMA

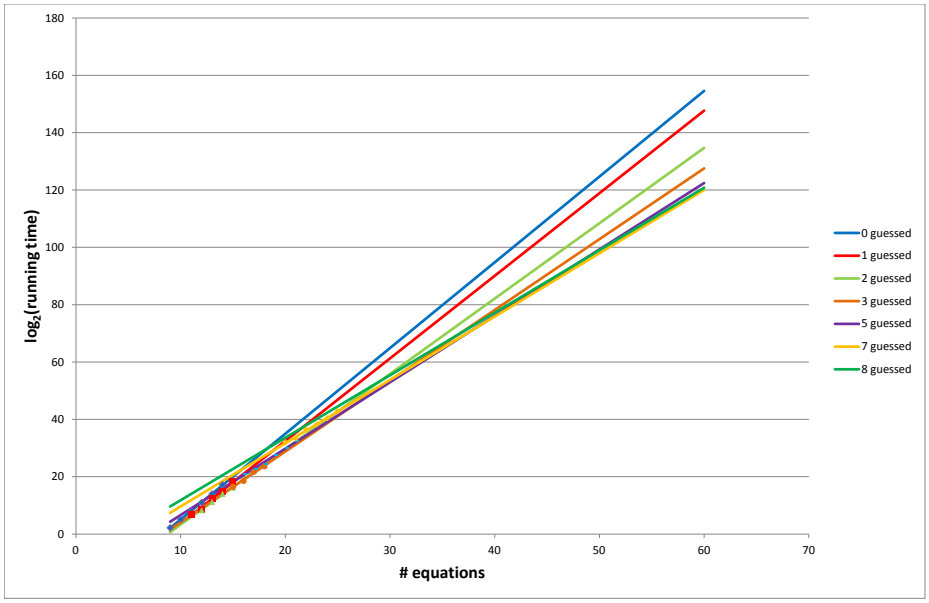


Figure 5.4: Running time of solving determined random systems over GF(16) with MAGMA

As the table shows, we get for our examples the best results when guessing 2 ($9 \leq m \leq 15$) or 3 ($m = 16$) variables. One can further see that the time MAGMA needs to find a solution of the system grows exponentially with the number of equations.
To estimate the running time for larger systems, we extrapolated the results of Table 5.1 for $m \in \{17, \ldots, 60\}$. This extrapolation (see Figure 5.4) shows that for more than 21 equations it is better to guess 4 or more variables. For the parameters usually used for multivariate schemes ($30 \leq m \leq 60$), it seems to be a good strategy to guess 6 or 7 variables before applying the F_4 algorithm. Guessing even more variables helps only for very large systems, since one has to run the F_4 algorithm too often.
We find that the actual running time of the F_4 algorithm can be closely estimated by

$$\text{running time}_{F_4}(16, m) = 2^{2.31 \cdot m} - 14.2 \text{ sec.} \tag{5.6}$$

Together with equation (4.4) we get

$$\text{bit complexity}_{F_4}(16, m) = 2.31 \cdot m + 15.4, \tag{5.7}$$

which is very similar to the result of (5.3).

Solving determined random systems over GF(31)

Table 5.2 shows the results of our MAGMA experiments with determined random systems over GF(31).

# equations	9	10	11	12	13	14	15	16
no guessing	4.8 s 30.7 MB	35.5 s 83.2 MB	4.4 m 387 MB	35.5 m 1,269 MB	286.4 m 6,186 MB	39.6 h 24,505 MB	- ooM [1]	-
1 guessed	**4.5 s** 9.2 MB	**29.8 s** 9.7 MB	**3.4 m** 12.9 MB	**19.8 m** 28.6 MB	**152.5 m** 81.3 MB	**15.6 h** 284 MB	134.5 h 1,025 MB	28.6 d 1,684 MB
2 guessed	7.6 s 7.9 MB	45.8 s 8.1 MB	4.6 m 8.3 MB	22.9 m 10.1 MB	200.7 m 16.0 MB	17.0 h 42.1 MB	**130.9 h** 106.1 MB	**20.4 d** 310.4 MB
3 guessed	31.7 s 8.0 MB	171.3 s 8.2 MB	15.4 m 8.4 MB	83.1 m 8.5 MB	448.5 m 10.8 MB	40.5 h 13.4 MB	176.7 h 23.3 MB	30.1 d 35.2 MB
4 guessed	92.6 s 8.1 MB	486.4 s 8.4 MB	42.6 m 8.7 MB	223.5 m 9.3 MB	1,173 m 9.8 MB	102.7 h 10.3 MB	487.4 h 13.5 MB	81.2 d 16.7 MB

[1] out of memory

Table 5.2: Running time of solving determined random systems over GF(31) with MAGMA

As the table shows, we get for our examples the best results when guessing 1 ($9 \leq m \leq 14$) or 2 ($15 \leq m \leq 16$) variables.
Similarly to the case of GF(16) (see above), we extrapolated these results to larger systems. This extrapolation (see Figure 5.5) shows that for more than 25 equations we get the best results when guessing 3 variables. When guessing 4 or more variables we have to run the F_4 algorithm so often that, for the parameters usually used in multivariate cryptography ($30 \leq m \leq 50$), we do not get a speed up of the running time.
We find that the actual running time of the F_4 algorithm can be closely estimated by

$$\begin{aligned} \text{running time}_{F_4}(31, m) &= 2^{2.63 \cdot m} - 20.4 \text{ sec } (17 \leq m \leq 25) \\ &= 2^{2.51 \cdot m} - 16.2 \text{ sec } (26 \leq m \leq 50). \end{aligned} \tag{5.8}$$

Together with equation (4.4) we get for $m \in \{26, \ldots, 50\}$

$$\text{bit complexity}_{F_4}(31, m) = 2.51 \cdot m + 13.4, \tag{5.9}$$

which is very similar to the result of (5.4).

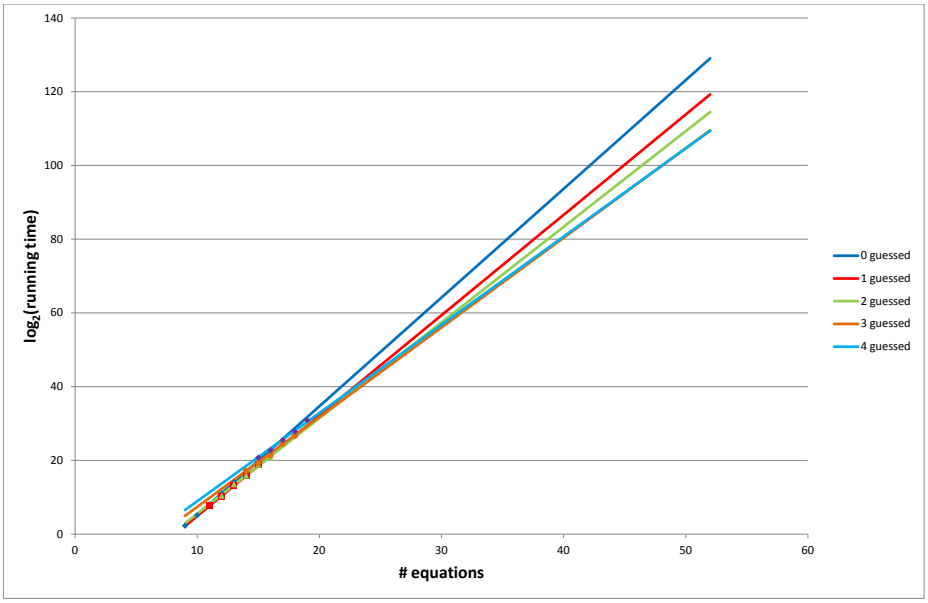


Figure 5.5: Running time of solving determined random systems over GF(31) with MAGMA

Solving determined random systems over GF(256)

Table 5.3 shows the results of our MAGMA experiments with determined random systems over GF(256).

# equations	9	10	11	12	13	14	15	16
0 guessed	**5.5 s** 28.4 MB	**40.9 s** 98.6 MB	**5.0 m** 352 MB	**39.8 m** 1,438 MB	**5.3 h** 6,456 MB	**47.0 h** 23,495 MB	- ooM [1]	-
1 guessed	128.2 s 9.8 MB	332.8 s 11.2 MB	31.6 m 13.3 MB	247.0 m 29.4 MB	41.0 h 83.4 MB	267.0 h 291.4 MB	78.0 d 724 MB	436.3 d 1,173 MB
2 guessed	9.2 m 8.1 MB	54.0 m 8.4 MB	382.3 m 8.6 MB	32.8 h 10.7 MB	122.0 h 14.5 MB	50.8 d 42 MB	230.2 d 118 MB	1,894 d 335 MB
3 guessed	309.5 m 8.1 MB	1,688 m 8.3 MB	153.4 h 8.6 MB	836.6 h 9.2 MB	174.8 d 11.3 MB	1,612 d 18.2 MB	4,777 d 26.1 MB	20,292 d 52.9 MB

[1] out of memory

Table 5.3: Running time of solving determined random systems over GF(256) with MAGMA

As one can see from the table, for our examples we get the best results without guessing. But, as the extrapolation (see Figure 5.6) shows, for more than 20 variables it is better to guess one variable and for more than 25 variables we get the best results when guessing two variables. When guessing 3 or more variables we have to run the F_4 algorithm so often that, for the parameters usually used in multivariate cryptography ($25 \leq m \leq 50$), we do not get a speed up of the running time.

We find that the actual running time of the F_4 algorithm can be closely estimated by a exponential function of the form

$$\text{running time}_{F_4}(256, m) = 2^{2.65 \cdot m} - 15.1 \text{ sec } (26 \leq m \leq 50). \tag{5.10}$$

Together with equation (4.4) we get for $m \in \{26, \ldots, 50\}$

$$\text{bit complexity}_{F_4}(256, m) = 2.65 \cdot m + 14.5, \tag{5.11}$$

which is very similar to the result of (5.5).

Memory

The memory consumption of Gröbner Basis techniques still remains a major problem of cryptanalysis. As can be seen from the Tables 5.1 - 5.3 we ran, regardless of the underlying field, out of memory when trying to solve determined systems with more than 14 equations. However, by guessing some of the variables and solving overdetermined systems, it is possible to reduce the memory consumption by a significant factor.

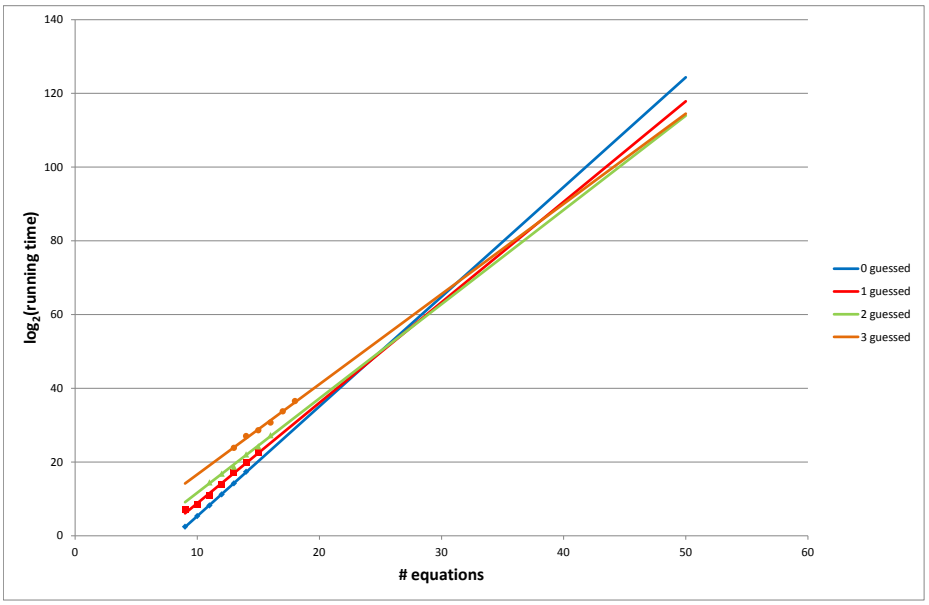


Figure 5.6: Running time of solving determined random systems over GF(256) with MAGMA

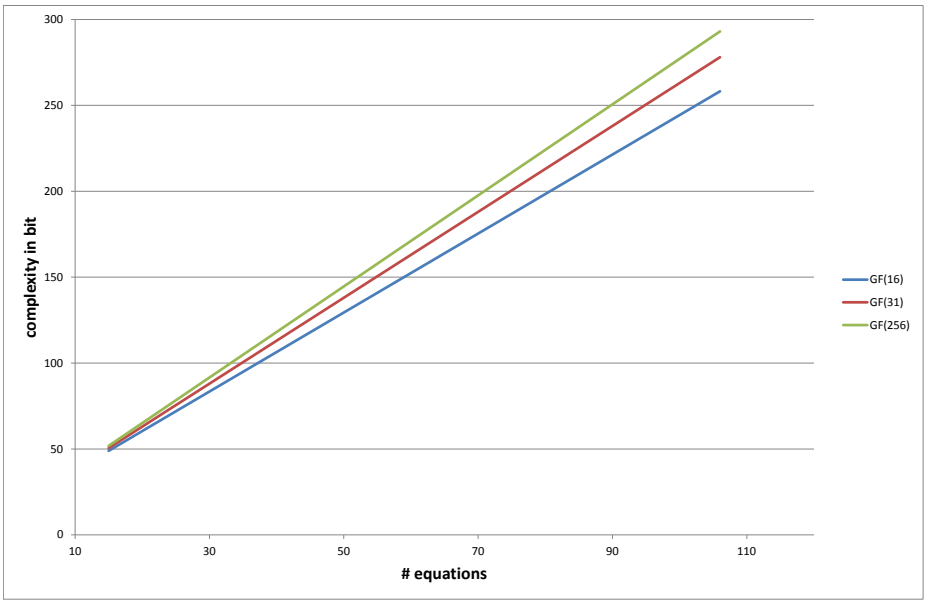


Figure 5.7: Relation between number of equations and bit complexity for determined random systems over different fields

security level (bit)	number of equations GF(16)	GF(31)	GF(256)
80	30	28	26
100	39	36	33
128	51	48	43
192	80	75	68
256	110	103	93

Table 5.4: Minimal number of equations needed to reach given levels of security

5.3 Results

In this section we summarize the results of the previous two sections and give recommendations for the minimal number of equations a multivariate scheme should contain to reach given levels of security.
As we have seen, the theoretical behavior of the HybridF_5 algorithm and the actual running time of MAGMA's F_4 implementation are very similar. Since the HybridF_5 algorithm seems to be slightly faster, we derive the results presented in this section mainly from the theoretical analysis of this algorithm. Therefore we find that the bit complexity of solving a determined multivariate system of m equations with randomly chosen coefficients directly is roughly given by

$$2.30 \cdot m + 14.4 \tag{5.12}$$

for systems over GF(16),

$$2.50 \cdot m + 13.0 \tag{5.13}$$

for systems over GF(31) and

$$2.65 \cdot m + 12.1 \tag{5.14}$$

for systems over GF(256).

Figure 5.7 illustrates this relation.
The minimal number m of equations needed to reach a security level of ℓ bit is therefore given by

$$m = \lceil \frac{\ell - 14.4}{2.30} \rceil \tag{5.15}$$

equations over GF(16),

$$m = \lceil \frac{\ell - 13.0}{2.50} \rceil \tag{5.16}$$

equations over GF(31) or

$$m = \lceil \frac{\ell - 12.1}{2.65} \rceil \tag{5.17}$$

equations over GF(256).

Table 5.4 shows the minimal number of equations needed to reach given levels of security.

Table 5.5 shows the minimal number of equations needed for the security of a multivariate scheme in the years 2012 to 2050. Note that in this chapter we considered only random systems, as they are used by the MQ-based identification scheme [52] or by the QUAD stream cipher [2] (see also Chapter 11). While for some multivariate public key schemes (e.g. Rainbow) the public systems $\mathcal{P}$ behave very similar to random systems (see Chapter 6), the public systems of other multivariate schemes (e.g. HFE) are much easier to solve [39]. Therefore, for multivariate public key schemes, the numbers presented in the Tables 5.4 and 5.5 can only be seen as lower bounds.

year	symmetric key size (bit)	minimal number of equations over GF(16)	GF(31)	GF(256)
1982	56			
2010	78	29	27	25
2011	79	29	27	25
2012	79	30	28	26
2013	80	30	28	26
2014	81	30	28	26
2015	82	31	28	27
2016	83	32	29	27
2017	83	32	30	28
2018	84	33	31	28
2019	85	33	31	28
2020	86	35	31	29
2021	86	35	32	30
2022	87	35	33	30
2023	88	36	34	30
2024	89	37	35	31
2025	89	38	35	32
2026	90	38	36	32
2027	91	39	36	33
2028	92	39	36	33
2029	93	40	37	34
2030	93	40	37	34
2031	94	42	38	35
2032	95	42	39	35
2033	96	43	40	36
2034	96	43	40	37
2035	97	44	41	37
2036	98	45	42	38
2037	99	45	42	38
2038	99	46	42	39
2039	100	46	43	39
2040	101	47	44	40
2041	102	48	44	40
2042	102	48	44	41
2043	103	49	45	41
2044	104	50	46	42
2045	105	50	47	42
2046	106	51	47	43
2047	106	52	48	43
2048	107	52	48	44
2049	108	53	49	45
2050	109	54	49	45

Table 5.5: Minimal number of equations needed for the security of MQ-schemes

Furthermore, to find good parameters for a certain multivariate scheme, one has to analyze key recovery attacks (i.e. attacks against the EIP-Problem), too. Since, for each multivariate scheme, different attacks have to be considered, one has to perform this analysis for each scheme separately. In the next chapter we do this for the Rainbow signature scheme.

5.4 Parameters for UOV

In this section we use the results of the previous sections to derive good parameters for the Unbalanced Oil and Vinegar signature scheme of Section 3.1. To defend the scheme against the UOV attack of Kipnis and Shamir (see Subsection 3.3.3) we choose $v = 2 \cdot o$ [33]. The number of variables in the public system is therefore given by $n = 3 \cdot o$, the number of equations is o.
The public systems of UOV behave very similar to random systems [5]. We can therefore use the results derived in this chapter to estimate the impact of direct attacks against UOV. By using the attack on underdetermined systems (see [56] and Subsection 2.5.4), we can reduce the number of equations in the public systems by 2 before applying an algorithm like XL or a Gröbner Basis method. Therefore, in order to get a secure UOV scheme, we have to increase the numbers shown in Table 5.4 and 5.5 by 2. Furthermore we have to test, if the so obtained parameters lead to secure hash lengths.

Table 5.6 presents our proposed parameter sets for the UOV scheme for different levels of security and different underlying fields.

	security level (bit)	parameters (o, v)	public key size (kB)	private key size (kB)	hash size (bit)	signature size (bit)
GF(16)[1]	80	(40,80)	144.2	135.2	160	**480**
	100	(50,100)	280.2	260.1	200	**600**
	128	(64,128)	585.0	538.1	256	**768**
	192	(96,192)	1,964.3	1,786.7	384	**1,152**
	256	(128,256)	4,644.1	4,200.3	512	**1,536**
GF(31)[1]	80	(33,66)	108.5	102.9	160	528 [2]
	100	(41,82)	206.9	193.8	200	656 [2]
	128	(52,104)	419.9	389.2	256	832 [2]
	192	(78,156)	**1,408.2**	**1,287.5**	384	1,248 [2]
	256	(104,208)	**3,327.3**	**3,021.1**	512	1,664 [2]
GF(256)	80	(28,56)	**99.9**	**95.8**	224	672
	100	(35,70)	**193.8**	**183.2**	280	840
	128	(45,90)	**409.4**	**381.8**	360	1080
	192	(70,140)	1,528.9	1,402.3	560	1,680
	256	(95,190)	3,807.5	3,464.1	760	2,280

1 For UOV schemes over GF(16) and GF(31), the number of equations is determined by the length of the hash value.

2 Theoretically, the length of a signature should be three times the length of the hash value. The differences are caused by data conversions between GF(31) and $\{0,1\}^\star$ (see Subsection 6.3.1).

Table 5.6: Recommended parameters for UOV over GF(16), GF(31) and GF(256) for different levels of security

As Table 5.6 shows, we get the shortest signatures when using UOV over GF(16). On the other hand, the key sizes of these schemes are much larger than those of UOV schemes over larger fields. Here we get, at least for moderate security levels (80 to 128 bit) the best results when using UOV over GF(256). For high levels of security (192 and 256 bit), UOV schemes over GF(31) lead to smaller key sizes. However, even for those schemes, the sizes of the public and private key are very large (> 1 MB).

Chapter 6

Selecting Parameters for the Rainbow Signature Scheme

The Rainbow signature scheme (see Section 3.4) is one of the best studied and most promising multivariate signature schemes. In this chapter we develop parameter sets which guarantee the security of this scheme for the years 2012 - 2050. We optimize these parameters simultaneously for both public and private key size. Furthermore we recommend Rainbow parameters for some fixed levels of security and compare Rainbow schemes over GF(16), GF(31) and GF(256) in terms of key and signature sizes.
In our approach we follow the model of Lenstra and Verheul (see Chapter 4), which, for a given year y, yields the required security level in bits.
To find good and secure parameters for Rainbow, we have to study the known attacks against the scheme. These include (see Section 3.6)

- direct attacks,
- the MinRank attack,
- the HighRank attack,
- the Rainbow-Band-Separation attack,
- the UOV attack and
- the UOV-Reconciliation attack.

Furthermore, since the parameters of Rainbow directly correspond to the length of the hash function in use, we have to choose them in such a way that a collision attack against the hash function is infeasible.
In this chapter, we denote by q the cardinality of the underlying field.

6.1 Our Strategy

Our strategy to find good parameters for Rainbow guaranteeing the security of the scheme in the year $y \in \{2012, \ldots, 2050\}$ consists of the following four steps:

1. Compute the symmetric key size $\ell(y)$ and the security level $\bar{\ell}(y)$ needed for the security of a multivariate scheme in the year y (following the model of Chapter 4).
2. Find the minimal number of equations needed to reach the required level of security. To do this we have to consider direct attacks as well as attacks against the hash function. In particular,

- the complexity of direct attacks (signature forgery attacks) against Rainbow must be greater or equal to the security level, i.e.

$$\text{complexity}_{\text{direct attack}}(q, m) \stackrel{!}{\geq} 2^{\bar{\ell}(y)} \text{ and} \tag{6.1}$$

- finding a collision of the hash function must be infeasible, i.e

$$m \cdot \lg q \stackrel{!}{\geq} 2 \cdot \ell(y) \text{ or } m = \lceil \frac{2 \cdot \ell(y)}{\lg q} \rceil. \tag{6.2}$$

Remark 6.1. *Note that in equation* (6.1) *and* (6.2) *we consider two different security levels. For the direct attacks (equation* (6.1)*) we have to consider possible advances in cryptanalysis (see Chapter 4). Therefore we have to compute the security level* $\bar{\ell}(y)$ *following equation* (4.3)*. On the contrary, the best attack against the collision resistance of hash functions is the birthday attack, which does not underly this development. Therefore, in equation* (6.2)*, we have to compute the security level* $\ell(y)$ *following equation* (4.2)*.*

3. Find the minimal number o_u of equations in the last Rainbow layer. Minimizing this number will help to reduce the size of the private key (see Table 6.1).

   - To find the number o_u we have to analyze the HighRank attack (see Subsection 3.6.3). We get

$$\text{complexity}_{\text{HighRank}}(q, o_u, n) = q^{o_u} \cdot \frac{n^3}{6} \geq q^{o_u} \frac{m^3}{6} \stackrel{!}{\geq} 2^{\bar{\ell}(y)}$$

$$\Rightarrow o_u = \lceil \frac{\bar{\ell}(y) - 3 \cdot \lg m + \lg 6}{\lg q} \rceil. \tag{6.3}$$

4. Find the minimal number n of variables needed to reach the given level of security. The relevant attacks here are the MinRank, the UOV, the UOV-Reconciliation and the Rainbow-Band-Separation attack.

   - For the MinRank attack (see Subsection 3.6.2) we have

$$\text{complexity}_{\text{MinRank}}(q, m, n) = q^{v_1+1} \cdot m \cdot \left(\frac{n^2}{2} - \frac{m^2}{6} \right) \geq q^{n-m+1} \cdot m \cdot \frac{m^2}{3} \stackrel{!}{\geq} 2^{\bar{\ell}(y)}$$

$$\Rightarrow n = \lceil \frac{\bar{\ell}(y) - 3 \cdot \lg m + \lg 3 - \lg q}{\lg q} + m \rceil. \tag{6.4}$$

   - For the UOV attack (see Subsection 3.3.3) we get

$$\text{complexity}_{\text{UOV-Attack}}(q, o_u, n) = q^{n-2 \cdot o_u - 1} \cdot o_u^4 \stackrel{!}{\geq} 2^{\bar{\ell}(y)}$$

$$\Rightarrow n = \lceil \frac{\bar{\ell}(y) - 4 \cdot \lg o_u + (2 \cdot o_u + 1) \cdot \lg q}{\lg q} \rceil. \tag{6.5}$$

   - To defend the scheme against the UOV-Reconciliation attack (see Subsection 3.3.2) we need

$$v_u \geq m \text{ or } v_1 \geq o_u \text{ and} \tag{6.6}$$

   - to defend the scheme against the Rainbow-Band-Separation attack (see Subsection 3.6.4) we need

$$\text{complexity}_{\text{RBS attack}}(q, m, n) \stackrel{!}{\geq} 2^{\bar{\ell}(y)}. \tag{6.7}$$

The most interesting items hereby are defending the scheme against direct attacks (equation (6.1)) and the Rainbow-Band-Separation attack (equation (6.7)). In the next three sections we analyze these two points for Rainbow schemes over GF(16), GF(31) and GF(256).

$(\mathbb{F}, m, n)$	(v_1, o_1, o_2)	private key size (byte)
(GF(16),40,57)	(17,17,23)	23,030
	(17,19,21)	22,887
	(17,21,19)	22,687
	(17,23,17)	22,444
(GF(256),26,43)	(17, 10,16)	19,786
	(17,12,14)	19,628
	(17,14,12)	19,460
	(17,16,10)	19,288

Table 6.1: Private key sizes for different layouts of the Rainbow layers

Remark 6.2. *As stated above, the number of equations is, apart from the length of the hash value, determined by the complexity of signature forgery attacks (see 2.)). On the other hand, the number o_u of equations in the last Rainbow layer and the number of variables n is determined by the complexity of key recovery attacks (see 3. and 4.).*

By following the above strategy, we simultaneously optimize the parameters of Rainbow for both public and private key size. The size of the public key (see equation (3.17)) depends only on the number of equations m and the number of variables n. So, by finding minimal values for m and n we minimize the size of the public key. By finding the minimal number o_u of equations in the last Rainbow layer (step 3) we additionally minimize the size of the private key (see equation (3.18) and Table 6.1). By dividing the middle layer into several sublayers, it is possible to reduce the size of the private key even further without weakening the security of the scheme. Furthermore this helps to speed up the signature generation process, as the systems which have to be solved by Gaussian Elimination get smaller.

Remark 6.3. *While, in order to reduce the size of the private key, it is a good strategy to choose the last Rainbow layer as small as possible, the opposite holds for the public key size of our improved versions of Rainbow (see Chapter 9). So, in order to find a good compromise between private and public key size, we choose for our improved versions the size of the middle and the last Rainbow layer as equal as possible. Note that this does not affect the security of the scheme (at least as long as $o_u \leq v_1$, see equation* (6.6)*).*

6.2 Rainbow Schemes over GF(16)

6.2.1 Number of Equations

The minimal number of equations required for the security of Rainbow is determined by the complexity of direct attacks and the length of the hash function output, i. e. m has to be chosen in such a way that a collision attack against the hash function is infeasible.
To estimate the complexity of direct attacks against Rainbow schemes over GF(16), we carried out a large number of experiments of solving the public systems of Rainbow over GF(16) with MAGMA's F_4 algorithm. Before we could apply the MAGMA function `Variety`, we had to convert the underdetermined Rainbow systems into determined ones by fixing $n - m$ variables. Since an underdetermined system over GF(16) with m equations in $n > m$ variables has approximately 16^{n-m} solutions, it can be expected that the determined system has on average exactly one solution.
By guessing additional variables we created overdetermined systems to see whether this reduces the time needed to find a solution of the original system. When guessing k variables, one has to run the F_4 algorithm 16^k times in order to find a solution of the original system. Table 6.2 and Figure 6.1 show the results of these experiments.

parameters	(5,4,5)	(6,5,5)	(7,5,6)	(8,6,6)	(9,6,7)	(10,7,7)	(11,7,8)	(12,8,8)
no guessing	4.5 s 30.5 MB	34.4 s 82.5 MB	4.2 m 460 MB	33.8 m 1,416 MB	273.2 m 7,041 MB	41.7 h 21,949 MB	- ooM	-
1 guessed	1.7 s 9.4 MB	12.2 s 11.7 MB	2.0 m 21.3 MB	7.0 m 65.7 MB	98.7 m 201.7 MB	8.6 h 782 MB	86.7 h 1,750 MB	22.9 d 2,425 MB
2 guessed	**1.5 s** 9.0 MB	**9.6 s** 16.4 MB	**1.0 m** 17.5 MB	**5.3 m** 29.7 MB	**41.2 m** 93.6 MB	**4.7 h** 116.2 MB	**21.6 h** 231.1 MB	6.2 d 306.4 MB
3 guessed	2.9 s 8.2 MB	16.7 s 10.7 MB	1.5 m 15.3 MB	8.5 m 18.6 MB	47.3 m 29.7 MB	5.8 h 34.2 MB	25.6 h 43.6 MB	**4.5 d** 58.7 MB
4 guessed	6.1 s 8.5 MB	32.7 s 10.3 MB	12.9 m 11.7 MB	15.3 m 13.2 MB	80.6 m 16.4 MB	7.0 h 20.3 MB	47.8 h 29.8 MB	6.1 d 35.7 MB
5 guessed	19.3 s 8.3 MB	97.2 s 9.4 MB	8.1 m 11.3 MB	40.1 m 12.8 MB	199.1 m 14.6 MB	16.3 h 16.5 MB	82.2 h 21.3 MB	25.4 d 28.8 MB
6 guessed	68.7 s 8.1 MB	327.9 s 9.3 MB	26.0 m 10.6 MB	123.9 m 11.7 MB	591.3 m 14.1 MB	47.1 h 15.9 MB	223.5 h 20.7 MB	44.3 d 27.3 MB
7 guessed	168.8 s 8.2 MB	780.3 s 9.1 MB	59.9 m 10.5 MB	278.0 m 11.4 MB	1,284 m 13.9 MB	98.5 h 18.7 MB	456.8 h 26.9 MB	88.0 d 30.1 MB
8 guessed	785.4 s 8.2 MB	3,542 s 9.3 MB	269.0 m 10.4 MB	1,227 m 11.3 MB	5,529 m 13.6 MB	417.3 h 17.8 MB	1,885 h 25.3 MB	355.9 d 28.7 MB

Table 6.2: Running time of solving Rainbow systems over GF(16) with MAGMA

As both Table 6.2 and the extrapolation in Figure 6.1 show, the public systems of Rainbow behave very similar to random systems. Therefore, the numbers m shown in the third column of Table 5.5 are high enough to defend a Rainbow scheme against direct attacks.

Remark 6.4. *As we found, the actual layer structure of Rainbow has no significant effect on the running time of direct attacks. In particular, the running time of our experiments was independent of the number $v_1 = n - m$ of variables we fixed to get a determined system.*

However, the numbers presented in Table 5.5 are not high enough to guarantee the collision resistance of a hash function. So, to reach a security level of ℓ bit, the number of equations in Rainbow schemes over GF(16) has to be chosen as

$$m^{\mathrm{GF}(16)}_{\mathrm{Rainbow}}(y) = \lceil \frac{2 \cdot \ell(y)}{\lg(16)} \rceil = \lceil \frac{\ell(y)}{2} \rceil. \tag{6.8}$$

The number o_u of equations in the last Rainbow layer is determined by complexity of the HighRank attack. For Rainbow schemes over GF(16) we get thereby (see equation (6.3))

$$o^{\mathrm{GF}(16)}_{u\ \mathrm{Rainbow}}(y) = \lceil \frac{\bar{\ell}(y) - 3 \cdot \lg m^{\mathrm{GF}(16)}_{\mathrm{Rainbow}}(y) + \lg 6}{4} \rceil. \tag{6.9}$$

6.2.2 Number of Variables

As stated in the previous section, the number of variables n required for the security of the Rainbow scheme is determined by the complexities of the MinRank, the UOV, the UOV-Reconciliation and the Rainbow-Band-Separation attack. Hereby, equations (6.4), (6.5) and (6.6) give a lower bound on n. We find that, for $y \leq 2035$, this number is determined by the UOV-Reconciliation attack, while, for $y > 2035$, n is determined by the complexity of the UOV attack. Therefore we get (see equations (6.6) and (6.5))

$$\begin{aligned} n^{\mathrm{GF}(16)}_{\mathrm{Rainbow}}(y) &= o^{\mathrm{GF}(16)}_{u\ \mathrm{Rainbow}}(y) + m^{\mathrm{GF}(16)}_{\mathrm{Rainbow}}(y) \ (y \leq 2035) \text{ or} \\ n^{\mathrm{GF}(16)}_{\mathrm{Rainbow}}(y) &= \lceil \frac{\bar{\ell}(y) - 4 \cdot \lg o^{\mathrm{GF}(16)}_{u\ \mathrm{Rainbow}}(y) + 4 \cdot (2 \cdot o^{\mathrm{GF}(16)}_{u\ \mathrm{Rainbow}}(y) + 1)}{4} \rceil \ (y > 2035). \end{aligned} \tag{6.10}$$

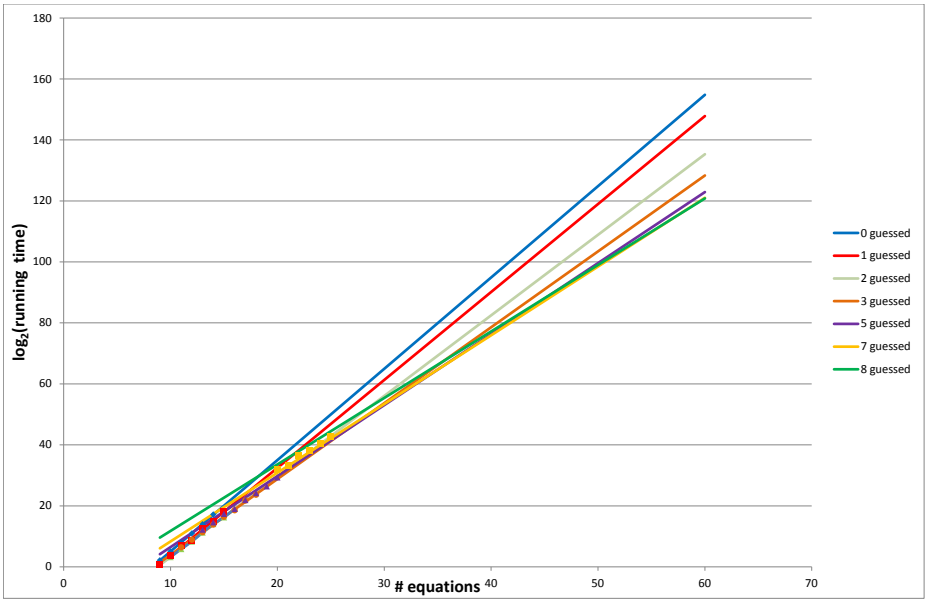


Figure 6.1: Running time of the direct attack against Rainbow schemes over GF(16) with guessing

# equations	9	10	11	12	13	14
(v_1, o_1, o_2)	(5,4,5)	(5,5,5)	(6,5,6)	(7,6,6)	(7,6,7)	(9,7,7)
time (s)	1.2	2.2	94.2	1,558	3,659	46,438
(v_1, o_1, o_2)	(6,4,5)	(6,5,5)	(7,5,6)	(8,6,6)	(8,6,7)	(10,7,7)
time (s)	4.8	44.2	1,550	4,775	26,103	163,207
for comparison: running time of the direct attack						
time (s)	4.5	34.4	255	2,027	16,405	150,120

Table 6.3: Running time of the RBS attack against Rainbow schemes over GF(16)

To check whether this number is high enough to defend the scheme against the Rainbow-Band-Separation (RBS) attack, we carried out some experiments to estimate the running time of this attack (see Table 6.3). We implemented the Rainbow-Band-Separation attack (as shown in Subsection 3.6.4) in MAGMA and ran it for different parameter sets. The systems appearing during the attack were solved with the F_4 algorithm integrated in MAGMA. By doing so, we found for some small m the minimal number n_{RBS} of variables such that

$$\text{complexity}_{\text{RBS attack}}(16, m, n_{\text{RBS}}) \geq \text{complexity}_{\text{direct attack}}(16, m). \tag{6.11}$$

For our small examples (see Table 6.3), this number is approximately given by

$$n_{\text{RBS}}^{\text{GF(16)}} \approx \frac{5}{3} \cdot m. \tag{6.12}$$

A theoretical analysis with the F_5 algorithm (see Figure 6.2) shows that this $n_{RBS}^{\text{GF(16)}}$ is high enough to defend the scheme against the RBS attack for arbitrary m. However, this number is lower than the bound given by equation (6.10). Therefore, choosing the number n according to equation (6.10) leads to Rainbow schemes which are also secure against the RBS attack.

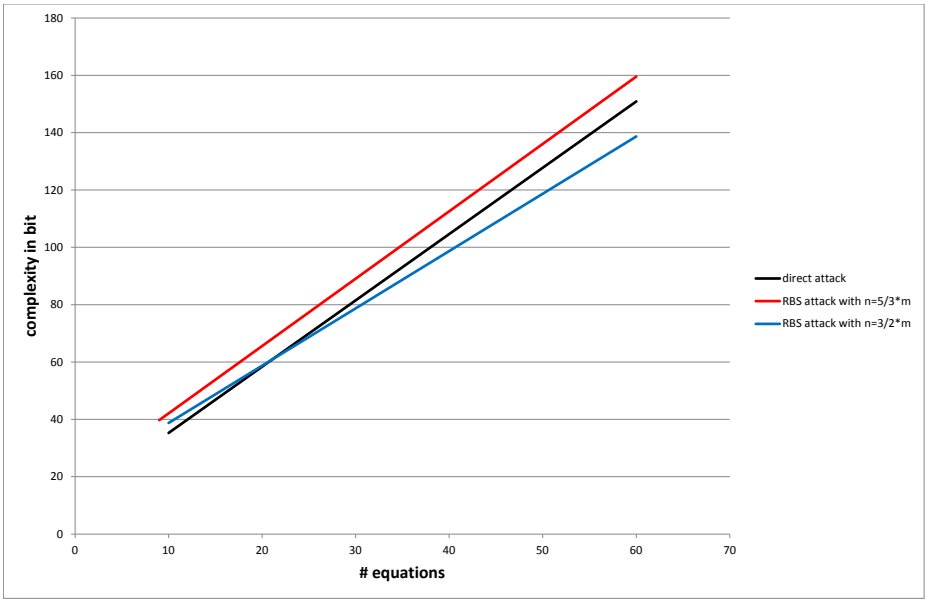


Figure 6.2: Complexity of the RBS attack against Rainbow schemes over GF(16)

6.2.3 Parameter Choice

Table 6.4 shows the proposed parameters for Rainbow schemes over GF(16) for the years 2012 - 2050. The parameters are optimized for small key sizes (both public and private).

To find these parameter sets, we applied the following strategy:
First, we used equation (6.8) to find the minimal number $m^{\mathrm{GF}(16)}_{\mathrm{Rainbow}}$ of equations needed to guarantee the security of Rainbow in the year y. Using this $m^{\mathrm{GF}(16)}_{\mathrm{Rainbow}}$ we computed the number of equations needed in the last Rainbow layer $o^{\mathrm{GF}(16)}_{u\ \mathrm{Rainbow}}$ by formula (6.9). Finally, we used both $m^{\mathrm{GF}(16)}_{\mathrm{Rainbow}}$ and $o^{\mathrm{GF}(16)}_{u\ \mathrm{Rainbow}}$ to compute the required number of variables in the scheme by formula (6.10).

Table 6.5 presents our parameter recommendations for Rainbow schemes over GF(16) to meet given levels of security. As in Table 6.4, the number of equations is determined by the required length of the hash value (equation (6.8)).

6.3 Rainbow Schemes over GF(31)

In [11] Chen et al. suggested to use multivariate schemes over the field GF(31). Using this field seems to be especially appropriate on PC's with modern CPU's supporting the SSE vector instruction set extensions. In this section we want to find the optimal parameters for the Rainbow signature scheme over GF(31) guaranteeing the security of the scheme for the years 2012 to 2050.

year	symmetric key size (bit)	required m	o_u	n	example scheme (v_1, o_1, o_2)	public key size (kB)	private key size (kB)	hash size (bit)	signature size (bit)
1982	56								
2010	78	39	17	56	(17, 22, 17)	31.5	20.9	156	224
2011	79	40	17	57	(17, 23, 17)	33.4	21.9	160	228
2012	79	40	17	57	(17, 23, 17)	33.4	21.9	160	228
2013	80	40	17	57	(17, 23, 17)	33.4	21.9	160	228
2014	81	41	18	59	(18, 23, 18)	35.6	23.6	164	236
2015	82	41	18	59	(18, 23, 18)	36.6	24.2	164	236
2016	83	42	18	60	(18, 24, 18)	37.7	24.8	168	240
2017	83	42	19	61	(19, 23, 19)	38.9	26.0	168	244
2018	84	42	19	61	(19, 23, 19)	40.1	26.6	168	244
2019	85	43	19	62	(19, 24, 19)	41.2	27.2	172	248
2020	86	43	20	63	(20, 23, 20)	43.7	29.2	172	252
2021	86	43	20	63	(20, 23, 20)	43.7	29.2	172	252
2022	87	44	20	64	(20, 24, 20)	44.9	29.8	176	256
2023	88	44	21	65	(21, 23, 21)	47.5	31.9	176	260
2024	89	45	21	66	(21, 24, 21)	48.8	32.6	180	264
2025	89	45	21	66	(21, 24, 21)	48.8	32.6	180	264
2026	90	45	22	67	(22, 23, 22)	51.5	34.8	180	268
2027	91	46	22	68	(22, 24, 22)	52.9	35.5	184	272
2028	92	46	23	69	(23, 23, 23)	55.8	37.8	184	276
2029	93	47	23	70	(23, 24, 23)	57.2	38.6	188	280
2030	93	47	23	70	(23, 24, 23)	57.2	38.6	188	280
2031	94	47	23	70	(23, 24, 23)	58.7	39.4	188	280
2032	95	48	24	72	(25, 24, 24)	63.3	44.1	192	288
2033	96	48	24	72	(24, 24, 24)	63.3	42.7	192	288
2034	96	48	24	72	(24, 24, 24)	63.3	42.7	192	288
2035	97	49	25	75	(27, 24, 25)	70.0	49.3	196	304
2036	98	49	25	75	(26, 24, 25)	70.0	47.8	196	300
2037	99	50	26	78	(29, 24, 26)	77.1	54.8	200	316
2038	99	50	26	78	(29, 24, 26)	77.1	54.8	200	316
2039	100	50	26	78	(28, 24, 26)	77.1	53.2	200	312
2040	101	51	27	80	(30, 24, 27)	82.7	58.9	204	324
2041	102	51	27	81	(30, 24, 27)	84.7	58.9	204	324
2042	102	51	27	81	(30, 24, 27)	84.7	58.9	204	324
2043	103	52	28	83	(32, 24, 28)	90.6	65.0	208	336
2044	104	52	28	84	(32, 24, 28)	92.8	65.1	208	336
2045	105	53	28	84	(32, 25, 28)	94.6	67.4	212	340
2046	106	53	29	87	(34, 24, 29)	101.3	71.7	212	348
2047	106	53	29	87	(34, 24, 29)	101.3	71.7	212	348
2048	107	54	29	87	(34, 25, 29)	103.3	74.1	216	352
2049	108	54	30	90	(36, 24, 30)	110.4	78.6	216	360
2050	109	55	30	90	(35, 25, 30)	112.4	79.1	220	360

Table 6.4: Recommended Parameters for Rainbow over GF(16)

security level (bit)	parameters (v_1, o_1, o_2)	public key size (kB)	private key size (kB)	hash size (bit)	signature size (bit)
80	(17,23,17)	33.4	21.9	160	228
100	(22,28,22)	65.9	42.7	200	288
128	(29,35,29)	139.5	89.3	256	372
192	(43,53,43)	462.7	287.4	384	556
256	(59,69,59)	1,110.4	687.2	512	748

Table 6.5: Recommended Parameters for Rainbow over GF(16) for different levels of security

6.3.1 Data Conversion between GF(31) and $\{0,1\}^\star$

Since both hash values and signatures are usually given as bit strings, one needs to convert elements of $\{0,1\}^\star$ into elements of GF(31) and vice versa. To store the keys, it is necessary to convert elements of GF(31) into bit strings, too.
Analogously to [11] we use the following data conversion between GF(31) and $\{0,1\}^\star$:

- 3 elements of GF(31) fit into a 2-byte block and
- an 8-byte block fits into 13 elements of GF(31).

While the second transformation is relatively strict, the transformation from GF(31) to $\{0,1\}^\star$ is not optimal. In fact, we would need only 15 bits to store 3 elements of GF(31) instead of 16 as in the upper transformation. So, with a better data conversion it would be possible to reach smaller key sizes and signatures. However, for the reason of simplicity, we use the same transformations as in [11].

6.3.2 Number of Equations

In this subsection we want to find the minimal number of equations needed for the security of the Rainbow scheme over GF(31) in the year y ($2012 \leq y \leq 2050$). This number is determined by the complexity of direct attacks and the required length of the hash function (see Section 6.1).
To estimate the impact of direct attacks we carried out a large number of experiments of solving the public systems of Rainbow over GF(31) with MAGMA's F_4 algorithm. Again we had to convert the underdetermined Rainbow systems into determined ones by fixing $n-m$ of the variables before applying the MAGMA function `Variety`. By further guessing 1, 2, 3 or 4 additional variables we created overdetermined systems to see whether this reduces the time needed to find a solution of the system. When guessing k variables, one has to run the F_4 algorithm 31^k times to find a solution of the original system. Table 6.6 shows the results of these experiments. Figure 6.3 shows the datapoints of the table together with an extrapolation to larger systems.

As both Table 6.6 and Figure 6.3 show, Rainbow systems over GF(31) behave very much like random systems over this field. Therefore, the numbers presented in the fourth column of Table 5.5 are high enough to defend a Rainbow scheme over GF(31) against direct attacks.

parameters	(5,4,5)	(6,5,5)	(7,5,6)	(8,6,6)	(9,6,7)	(10,7,7)	(11,7,8)	(12,8,8)
no guessing	4.7 s 30.1 MB	35.3 s 82.9 MB	4.3 m 385 MB	35.2 m 1,268 MB	285.9 m 6,184 MB	39.3 h 24,502 MB	- ooM	-
1 guessed	**4.3 s** 9.1 MB	**29.6 s** 9.3 MB	**3.4 m** 12.7 MB	**19.7 m** 28.5 MB	**152.0 m** 81.1 MB	**15.5 h** 282 MB	134.3 h 1,023 MB	28.4 d 1,679 MB
2 guessed	7.4 s 7.4 MB	45.7 s 8.0 MB	4.5 m 8.2 MB	22.7 m 9.9 MB	197.5 h 15.7 MB	16.8 h 41.7 MB	**130.5 h** 105.6 MB	**20.3 d** 307.6 MB
3 guessed	31.4 s 7.0 MB	170.9 s 8.2 MB	15.1 m 8.4 MB	82.9 m 8.5 MB	447.2 m 10.8 MB	40.5 h 13.4 MB	176.6 h 23.3 MB	30.1 d 35.2 MB
4 guessed	92.4 s 7.0 MB	486.0 s 8.3 MB	42.3 m 8.6 MB	222.3 m 9.1 MB	1,167 m 9.6 MB	102.5 h 10.2 MB	487.4 h 13.3 MB	80.9 d 16.5 MB

Table 6.6: Running time of solving Rainbow systems over GF(31) with MAGMA

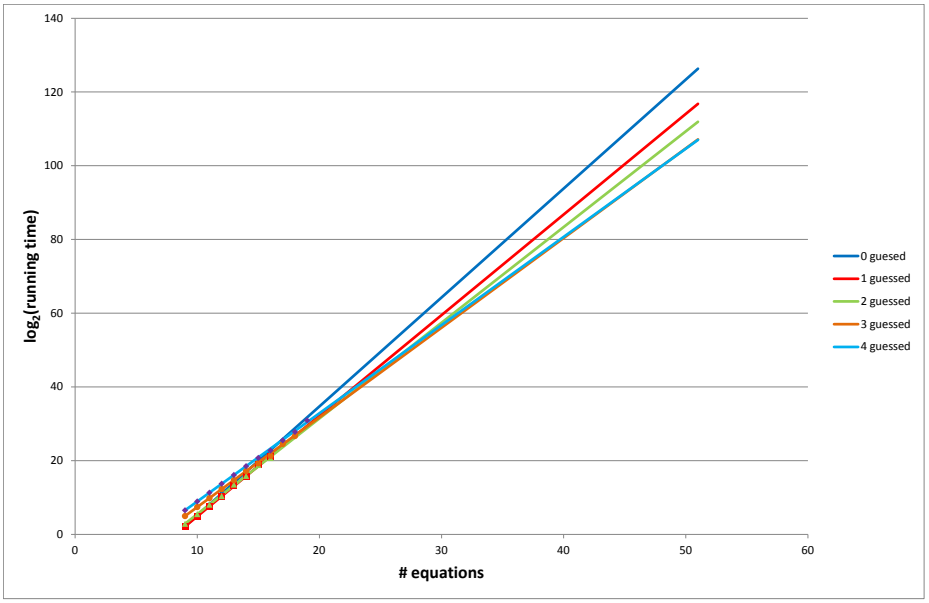


Figure 6.3: Running time of the direct attack against Rainbow schemes over GF(31)

# equations	9	10	11	12	13	14
	(5,4,5)	(6,5,5)	(6,5,6)	(8,6,6)	(8,6,7)	(9,7,7)
time (s)	2.1	22.2	60.9	1,102	2,988	44,491
	(6,4,5)	(7,5,5)	(7,5,6)	(9,6,6)	(9,6,7)	(10,7,7)
time (s)	6.9	41.2	299.2	4,091	21,302	415,458
	for comparison: running time of the direct attack					
time (s)	4.7	35.3	263	2,132	17,186	141,565

Table 6.7: Complexity of the RBS attack against Rainbow Schemes over GF(31)

However, for the years $y \leq 2035$ these numbers are not high enough to guarantee the collision resistance of the hash function in use. So, the minimal number of equations required for a Rainbow scheme over GF(31) being secure in the year y is given by

$$\begin{aligned} m^{\mathrm{GF}(31)}_{\mathrm{Rainbow}}(y) &= \lceil \frac{2 \cdot \ell(y)}{\lg(31)} \rceil \ (y \leq 2035) \text{ or} \\ m^{\mathrm{GF}(31)}_{\mathrm{Rainbow}}(y) &= \lceil \frac{\bar{\ell}(y) - 13.0}{2.5} \rceil (y > 2035). \end{aligned} \tag{6.13}$$

The minimal number o_u of equations in the last Rainbow layer is determined by the complexity of the HighRank attack. For Rainbow schemes over GF(31) we get (see equation (6.3))

$$o^{\mathrm{GF}(31)}_{u\ \mathrm{Rainbow}}(y) = \lceil \frac{\bar{\ell}(y) - 3 \cdot \lg m^{\mathrm{GF}(31)}_{\mathrm{Rainbow}}(y) + \lg 6}{\lg(31)} \rceil. \tag{6.14}$$

6.3.3 Number of Variables

The number n of variables needed in a Rainbow scheme is determined by the complexities of the MinRank, the UOV, the UOV-Reconciliation and the Rainbow-Band-Separation attack. Let us first take a look at the first three of these attacks which give a lower bound for n. We found that, for $y \leq 2032$, this lower bound is determined by the UOV-Reconciliation attack, while for $y \geq 2033$ the UOV attack is more efficient. Therefore we get (see equations (6.5) and (6.6))

$$\begin{aligned} n^{\mathrm{GF}(31)}_{\mathrm{Rainbow}} \quad (y) &= m^{\mathrm{GF}(31)}_{\mathrm{Rainbow}}(y) + o^{\mathrm{GF}(31)}_{u\ \mathrm{Rainbow}}(y) \ (y \leq 2032) \text{ or} \\ n^{\mathrm{GF}(31)}_{\mathrm{Rb}} &= \lceil \frac{\bar{\ell}(y) - 4 \cdot \lg o^{\mathrm{GF}(31)}_{u\mathrm{Rainbow}}(y) + \lg(31) \cdot (2 \cdot o^{\mathrm{GF}(31)}_{u\ \mathrm{Rainbow}}(y) + 1)}{\lg(31)} \rceil \ (y > 2032) \end{aligned} \tag{6.15}$$

To estimate the running time of the Rainbow-Band-Separation (RBS) attack, we carried out a number of computer experiments. We implemented the RBS attack as shown in Subsection 3.6.4 in MAGMA and used it to attack Rainbow schemes over GF(31) with different parameter sets (see Table 6.7). The quadratic systems were solved with the F_4 algorithm integrated in MAGMA.

For our small example parameters we found that for $n^{\mathrm{GF}(31)}_{\mathrm{RBS}} \approx \frac{5}{3} \cdot m$ we get

$$\mathrm{complexity}_{\mathrm{RBS\ attack}}(31, m, n^{\mathrm{GF}(31)}_{\mathrm{RBS}}) \geq \mathrm{complexity}_{\mathrm{direct\ attack}}(31, m) \ (9 \leq m \leq 14) \tag{6.16}$$

The theoretical analysis with F_5 (see Figure 6.4) shows that this relation holds for all $m \leq 60$ (i.e. for the values of m we need to guarantee the security of the scheme). As we find, the number $n^{\mathrm{GF}(31)}_{\mathrm{RBS}}$ is less than the lower bound of equation (6.15). Therefore, by choosing the number of variables according to (6.15) we get Rainbow parameters which are secure against the Rainbow-Band-Separation attack, too.

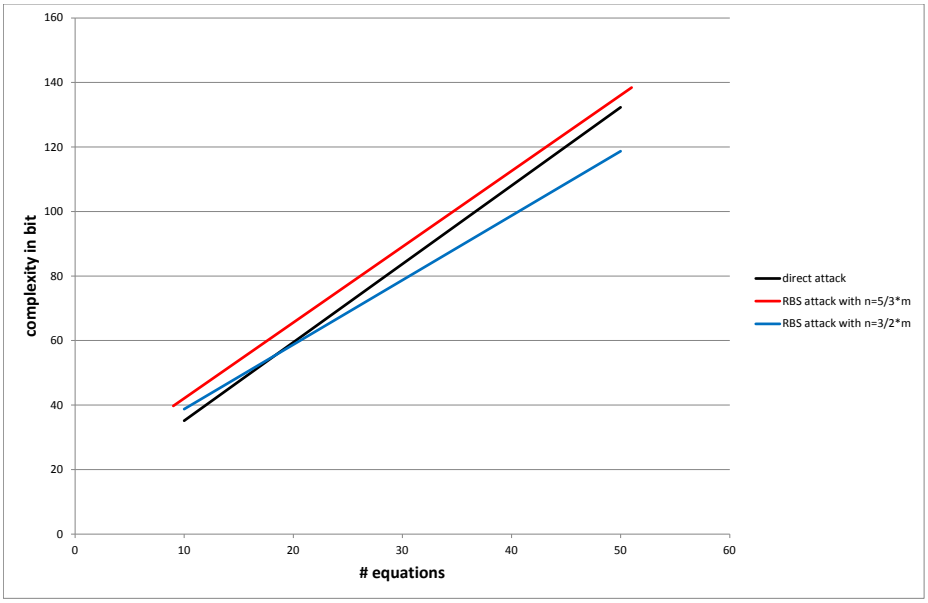


Figure 6.4: Running time of the RBS attack against Rainbow schemes over GF(31)

6.3.4 Parameter Choice

Table 6.8 presents the recommended parameters for Rainbow schemes over GF(31) for the years 2012 - 2050. The parameters are optimized for small key sizes (both public and private).

The parameters proposed in Table 6.8 are identified as follows: First we used equation (6.13) to find the minimal number of equations $m_{\text{Rainbow}}^{\text{GF(31)}}$ required for the security of the scheme in the year y. We then used this number to compute the minimal number $o_{u\ \text{Rainbow}}^{\text{GF(31)}}$ of equations in the last Rainbow layer (equation (6.14)). Finally, we used both $m_{\text{Rainbow}}^{\text{GF(31)}}$ and $o_{u\ \text{Rainbow}}^{\text{GF(31)}}$ to compute the required number of variables (following equation (6.15)).

Table 6.9 presents our parameter recommendations for Rainbow schemes over GF(31) to meet given levels of security. The number of equations is hereby given by the required length of the hash value.

6.4 Rainbow Schemes over GF(256)

6.4.1 Number of Equations

In this subsection we want to find the number m of equations needed in a Rainbow scheme over GF(256) to reach a given level of security. As we find, this number is mainly determined by direct attacks or signature forgery attacks. To find the minimal number m for Rainbow schemes over GF(256), we carried out a large number of experiments of solving the public systems of Rainbow over GF(256) with MAGMA's [8] implementation of the F_4 algorithm. Before we could apply the MAGMA function `Variety`, we had to convert the underdetermined Rainbow systems into determined ones by fixing $n-m$ of the variables. By further guessing 1, 2 or 3 additional variables

year	symmetric key size (bit)	required			example scheme	public key size (kB)	private key (kB)	hash size (bit)	signature size (bit)
		m	o_u	n	(v_1, o_1, o_2)				
1982	56								
2010	78	32	14	46	(14, 18, 14)	23.5	16.0	176	248
2011	79	33	14	47	(14, 19, 14)	25.3	17.0	176	256
2012	79	33	14	47	(14, 19, 14)	25.3	17.0	176	256
2013	80	33	14	47	(14, 19, 14)	25.3	17.0	176	256
2014	81	33	14	47	(14, 19, 14)	25.3	17.0	176	256
2015	82	34	15	49	(15, 19, 15)	28.2	19.1	184	264
2016	83	34	15	51	(17, 19, 15)	30.5	21.1	184	272
2017	83	34	15	52	(18, 19, 15)	31.7	22.2	184	280
2018	84	35	16	52	(17, 19, 16)	32.6	22.5	192	280
2019	85	35	16	52	(17, 19, 16)	32.6	22.5	192	280
2020	86	35	16	53	(18, 19, 16)	33.8	23.6	192	288
2021	86	35	16	54	(19, 19, 16)	35.1	24.7	192	288
2022	87	36	17	57	(21, 19, 17)	40.1	28.6	192	304
2023	88	36	17	57	(21, 19, 17)	40.1	28.6	192	304
2024	89	37	17	58	(21, 20, 17)	42.6	30.1	200	312
2025	89	37	17	58	(21, 20, 17)	42.6	30.1	200	312
2026	90	37	18	58	(21, 19, 18)	42.6	30.2	200	312
2027	91	37	18	61	(24, 19, 18)	47.0	34.2	200	328
2028	92	38	18	63	(25, 20, 18)	51.5	37.3	208	336
2029	93	38	19	63	(25, 19, 19)	51.5	37.4	208	336
2030	93	38	19	63	(25, 19, 19)	51.5	37.4	208	336
2031	94	39	19	63	(24, 20, 19)	52.8	37.8	208	336
2032	95	39	20	65	(26, 19, 20)	56.1	40.9	208	352
2033	96	40	20	68	(28, 20, 20)	62.9	46.0	216	368
2034	96	40	20	68	(28, 20, 20)	62.9	46.0	216	368
2035	97	41	20	69	(28, 21, 20)	66.3	48.2	224	368
2036	98	42	21	70	(28, 21, 21)	69.9	50.5	224	376
2037	99	42	21	70	(28, 21, 21)	69.9	50.5	224	376
2038	99	42	21	70	(28, 21, 21)	69.9	50.5	224	376
2039	100	43	21	73	(30, 22, 21)	77.7	56.4	232	392
2040	101	44	22	75	(31, 22, 22)	83.8	60.9	240	400
2041	102	44	22	75	(31, 22, 22)	83.8	60.9	240	400
2042	102	44	22	75	(31, 22, 22)	83.8	60.9	240	400
2043	103	45	23	76	(31, 22, 23)	88.0	63.6	240	408
2044	104	46	23	77	(31, 23, 23)	92.3	66.2	248	416
2045	105	47	23	80	(33, 24, 23)	101.6	73.3	256	432
2046	106	47	23	82	(35, 24, 23)	106.7	77.9	256	440
2047	106	48	24	82	(34, 24, 24)	108.9	78.7	256	440
2048	107	48	24	82	(34, 24, 24)	108.9	78.7	256	440
2049	108	49	24	83	(34, 25, 24)	113.9	81.7	264	448
2050	109	49	25	84	(35, 24, 25)	116.6	84.3	264	448

Table 6.8: Recommended Parameters for Rainbow over GF(31)

security level (bit)	parameters (v_1, o_1, o_2)	public key size (kB)	private key size (kB)	hash size (bit)	signature size (bit)
80	(14,19,14)	25.3	17.0	160	256
100	(18,23,18)	48.8	32.3	200	320
128	(23,29,23)	99.1	64.0	256	400
192	(35,43,35)	270.6	209.4	384	608
256	(48,56,48)	663.5	498.4	512	816

Table 6.9: Recommended Parameters for Rainbow over GF(31) for different levels of security

parameters	(5,4,5)	(6,5,5)	(7,5,6)	(8,6,6)	(9,6,7)	(10,7,7)	(10,7,8)	(11,8,8)
no guessing	**5.3 s** 28.1 MB	**40.6 s** 96.2 MB	**4.9 m** 345 MB	**39.7 m** 1,163 MB	**5.3 h** 7,419 MB	**46.8 h** 35,173 MB	- ooM	-
guessing 1 variable	128.0 s 9.9 MB	329.6 s 10.3 MB	31.4 m 11.2 MB	245.8 m 23.4 MB	40.4 h 76.1 MB	266.6 h 283 MB	78.0 d 989 MB	435.9 d 3,954 MB
guessing 2 variables	9.1 m 8.2 MB	53.7 m 8.2 MB	381.5 m 8.4 MB	32.6 h 10.3 MB	120.9 h 14.8 MB	50.6 d 41 MB	229.8 d 121 MB	1,886 d 334 MB
guessing 3 variables	307.4 m 8.1 MB	1,673 m 8.4 MB	152.6 h 8.7 MB	835.4 h 9.0 MB	172.4 d 11.2 MB	1,598 d 17.8 MB	4,751 d 24.8 MB	20,247 d 51.7 MB

Table 6.10: Running time of solving Rainbow systems over GF(256) with MAGMA

we created overdetermined systems to see whether this reduces the time needed to find a solution of the system. When guessing k variables, one has to run the F_4 algorithm 256^k times to find a solution of the original system. Table 6.10 shows the results of these experiments. Figure 6.5 shows the datapoints of the table as well as an extrapolation for larger values of m.
As a comparison to Table 5.6 and Figure 5.6 shows, the public systems of Rainbow behave very similar to random systems. We can therefore use the values shown in the fifth column of Table 5.5 to build secure Rainbow schemes. Note that these numbers are much higher than needed for the collision resistance of the hash function.

We therefore get (see equation (5.17))

$$m_{\text{Rainbow}}^{\text{GF(256)}}(y) = \lceil \frac{\bar{\ell}(y) - 12.1}{2.65} \rceil. \tag{6.17}$$

The number o_u of equations in the last Rainbow layer is determined by the complexity of the HighRank attack. For Rainbow schemes over GF(256) we get thereby (see equation (6.3))

$$o_{u\ \text{Rainbow}}^{\text{GF(256)}}(y) = \lceil \frac{\bar{\ell}(y) - 3 \cdot \lg m_{\text{Rainbow}}^{\text{GF(256)}}(y) + \lg 6}{8} \rceil. \tag{6.18}$$

6.4.2 Number of Variables

The number of variables needed in a Rainbow scheme is determined by the complexities of the MinRank, the UOV, the UOV-Reconciliation and the Rainbow-Band-Separation attack. For Rainbow schemes over GF(256) we find that the Rainbow-Band-Separation (RBS) attack is the most efficient of these attacks. To estimate the complexity of the RBS attack against Rainbow schemes over GF(256), we carried out a number of experiments with MAGMA. We implemented the RBS attack as shown in Subsection 3.6.4 in MAGMA code and used it to attack Rainbow instances over GF(256) with different parameter sets. The quadratic systems appearing during the attack were solved with the F_4 algorithm integrated in MAGMA. Table 6.11 shows the results.

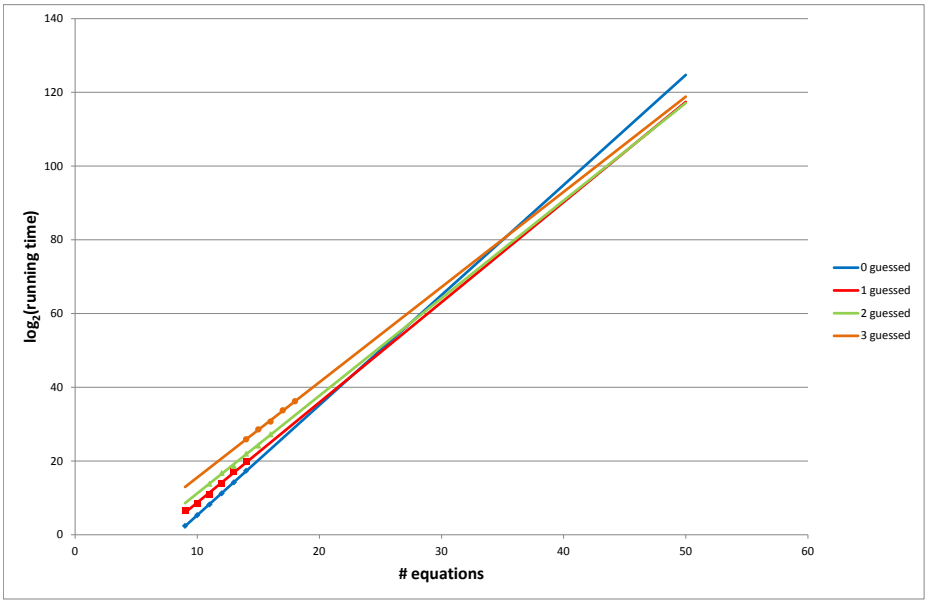


Figure 6.5: Running time of the direct attack against Rainbow schemes over GF(256) with guessing

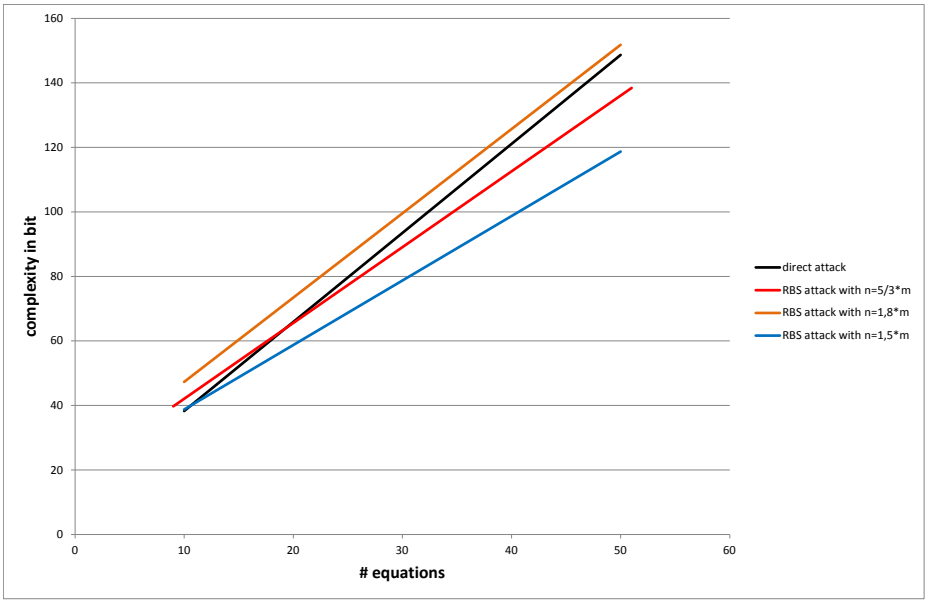


Figure 6.6: Complexity of the RBS attack against Rainbow schemes over GF(256)

# equations	9	10	11	12	13	14
	(5,4,5)	(6,5,5)	(8,5,6)	(8,6,6)	(8,6,7)	(10,7,7)
time (s)	1.3	12.6	115.2	1,141	3,115	46,438
	(6,4,5)	(7,5,5)	(9,5,6)	(9,6,6)	(9,6,7)	(11,7,7)
time (s)	20.9	44.2	693.3	4206	22,064	936,659
	for comparison: running time of the direct attack					
time (s)	5.3	40.6	292.5	2,382	19,031	168,578

Table 6.11: Running time of the RBS attack against Rainbow schemes over GF(256)

security level (bit)	parameters (v_1, o_1, o_2)	public key size (kB)	private key size (kB)	hash size (bit)	signature size (bit)
80	(17,17,9)	25.1	18.8	208	344
100	(26,22,11)	59.0	44.4	264	472
128	(36,28,15)	136.1	101.3	344	632
192	(63,46,22)	582.9	430.3	544	1048
256	(85,63,30)	1,463.1	1,061.4	744	1424

Table 6.12: Recommended parameters for Rainbow over GF(256) for different levels of security

As Table 6.11 shows, we get for our small examples

$$\text{complexity}_{\text{RBS attack}}(256, m, \frac{5}{3} \cdot m) \geq \text{complexity}_{\text{direct attack}}(256, m). \tag{6.19}$$

Furthermore, we analyzed the RBS attack theoretically when solving the quadratic systems with the HybridF_5 algorithm (see Figure 6.6) to find for $20 \leq m \leq 50$ the minimal number $n_{\text{RBS}}^{GF(256)}$ such that

$$\text{complexity}_{\text{RBS attack}}(256, m, n_{\text{RBS}}^{\text{GF}(256)}) \geq \text{complexity}_{\text{direct attack}}(256, m). \tag{6.20}$$

This analysis shows, that for large m, we need a bit more variables than indicated by (6.19). The values of n shown in Table 6.13 are derived by this analysis.

6.4.3 Parameter Choice

Table 6.13 shows our parameter recommendations for Rainbow schemes over GF(256) for the years 2012 - 2050. The parameters are optimized for small key sizes (both public and private).

To get the parameters proposed in Table 6.13, we first used equation (6.17) to compute the number of equations $m_{\text{Rainbow}}^{\text{GF}(256)}$ needed for the security of the Rainbow scheme in the year y. We used this number and equation (6.18) to compute the number $o_{u\ \text{Rainbow}}^{\text{GF}(256)}$ of equations in the last Rainbow layer and finally determined the required number of variables by the theoretical analysis of the Rainbow-Band-Separation attack.

Table 6.12 presents our parameter recommendations for Rainbow schemes over GF(256) to meet given levels of security.

year	symmetric key size	required m	required o_u	required n	example scheme (v_1, o_1, o_2)	public key size (kB)	private key size (kB)	hash size (bit)	signature size (bit)
1982	56								
2010	78	25	9	42	(17,16,9)	23.1	17.5	200	336
2011	78	25	9	42	(17,16,9)	23.1	17.5	200	336
2012	79	26	9	43	(17, 17, 9)	25.1	18.8	208	344
2013	80	26	9	44	(18, 17, 9)	26.3	19.8	208	352
2014	81	26	9	46	(20, 17, 9)	28.6	22.1	208	368
2015	82	27	9	47	(20, 18, 9)	31.0	23.6	216	376
2016	83	27	10	47	(20, 17, 10)	31.0	23.7	216	376
2017	83	28	10	48	(20, 18, 10)	33.5	25.2	224	384
2018	84	28	10	48	(20, 18, 10)	33.5	25.2	224	384
2019	85	28	10	50	(22, 18, 10)	36.3	27.8	224	400
2020	86	29	10	53	(24, 19, 10)	42.1	32.5	232	424
2021	86	30	10	54	(24, 20, 10)	45.1	34.4	240	432
2022	87	30	11	54	(24, 19, 11)	45.1	34.5	240	432
2023	88	30	11	54	(24, 19, 11)	45.1	34.5	240	432
2024	89	31	11	55	(24, 20, 11)	48.3	36.5	248	440
2025	89	32	11	56	(24, 21, 11)	51.7	38.6	256	448
2026	90	32	11	58	(26, 21, 11)	55.3	42.1	256	464
2027	91	33	11	59	(26, 22, 11)	59.0	44.4	264	472
2028	92	33	12	59	(26, 21, 12)	59.0	44.5	264	472
2029	93	34	12	60	(26, 22, 12)	62.8	46.9	272	480
2030	93	34	12	60	(26, 22, 12)	62.8	46.9	272	480
2031	94	35	12	62	(27, 23, 12)	68.9	51.3	280	496
2032	95	35	12	65	(30, 23, 12)	75.6	57.6	280	520
2033	96	36	13	66	(30, 23, 13)	80.1	60.5	288	528
2034	96	37	13	67	(30, 24, 13)	84.8	63.5	296	536
2035	97	37	13	67	(30, 24, 13)	84.8	63.5	296	536
2036	98	38	13	68	(30, 25, 13)	89.6	66.6	304	544
2037	99	38	13	69	(31, 25, 13)	92.2	69.0	304	552
2038	99	39	13	70	(31, 26, 13)	97.3	72.2	312	560
2039	100	39	14	72	(33, 25, 14)	102.9	77.4	312	576
2040	101	40	14	73	(33, 26, 14)	108.4	80.9	320	584
2041	102	40	14	73	(33, 26, 14)	108.4	80.9	320	584
2042	102	41	14	74	(33, 27, 14)	114.1	84.5	328	592
2043	103	41	14	74	(33, 27, 14)	114.1	84.5	328	592
2044	104	41	14	77	(36, 27, 14)	123.4	93.1	328	616
2045	105	42	15	78	(36, 27, 15)	129.6	97.2	336	624
2046	106	43	15	79	(36, 28, 15)	136.1	101.3	344	632
2047	106	43	15	79	(36, 28, 15)	136.1	101.3	344	632
2048	107	44	15	80	(36, 29, 15)	142.7	105.4	352	640
2049	108	45	15	81	(36, 30, 15)	149.5	109.7	360	648
2050	109	45	15	84	(39, 30, 15)	160.6	120.0	360	672

Table 6.13: Recommended parameters for Rainbow over GF(256)

	GF(16)		GF(31)		GF(256)	
year	public key size (kB)	private key size (kB)	public key size (kB)	private key size (kB)	public key size (kB)	private key size (kB)
2012	33.4	21.9	**25.3**	**17.0**	25.1	18.8
2015	36.6	24.2	**28.2**	**19.1**	31.0	23.6
2020	43.7	29.2	**33.8**	**23.6**	42.1	32.5
2030	57.2	38.6	**51.5**	**37.4**	62.8	46.9
2040	**82.7**	**58.9**	83.8	60.9	108.4	80.9
2050	**112.4**	**79.1**	116.6	84.3	160.6	120.0

Table 6.14: Key sizes (in kB) of Rainbow schemes over different fields for the years 2012 - 2050

security level (bit)	GF(16)		GF(31)		GF(256)	
	public key size (kB)	private key size (kB)	public key size (kB)	private key size (kB)	public key size (kB)	private key size (kB)
80	33.4	21.9	**25.3**	**17.0**	26.3	18.8
100	65.9	42.7	**48.8**	**32.3**	59.0	44.4
128	139.5	89.3	**99.1**	**64.0**	136.1	101.3
192	462.7	287.3	**270.6**	**209.4**	582.9	430.3
256	1,110.4	687.2	**663.5**	**498.4**	1,463.1	1,061.5

Table 6.15: Key sizes (in kB) of Rainbow schemes over different fields for different levels of security

6.5 Summary and Comparison

In this section we summarize the results presented in the previous sections. We compare Rainbow schemes over the three fields GF(16), GF(31) and GF(256) in terms of key sizes and signature lengths.

6.5.1 Key Sizes

The Tables 6.14 and 6.15 show the key sizes of Rainbow schemes over GF(16), GF(31) and GF(256).

As we can see from Table 6.14, choosing GF(31) as underlying field seems to be a good choice to get small key sizes for Rainbow for now and the next 25 years. Starting from the year 2039, the key sizes of Rainbow schemes over GF(16) are the smallest ones.
When we compare Rainbow schemes of the same security level (see Table 6.15), the situation looks similar: Again we get the smallest keys for Rainbow schemes over GF(31). For Rainbow schemes over GF(16) the number of equations increases much faster than for Rainbow schemes over larger fields to meet the length of a collision resistant hash function. Therefore, using Rainbow over GF(16) does not help to decrease the key sizes in this context.
An interesting fact hereby is the ratio between the public and private key size of Rainbow schemes. In the year 2012 this ratio is about 1.3 for Rainbow over GF(256) and 1.5 for Rainbow over GF(16) and GF(31). The reason for this is the different ratio between the number of variables and the number of equations (1.6 and 1.4 for Rainbow schemes over GF(256) and GF(16) respectively). The reason for this again is that for Rainbow schemes over GF(256) the Rainbow-Band-Separation attack plays a major role, while, for Rainbow schemes over smaller fields, the number of variables is determined by the MinRank and the UOV-Reconciliation attack.

year	Signature size (bit) of Rainbow schemes over GF(16)	GF(31)	GF(256)
2012	**228**	256	344
2015	**236**	264	376
2020	**252**	288	424
2030	**280**	336	480
2040	**324**	400	584
2050	**360**	448	672

Table 6.16: Signature sizes (in bit) of Rainbow schemes over different fields for the years 2012 - 2050

security level (bit)	Signature size (bit) of Rainbow schemes over GF(16)	GF(31)	GF(256)
80	**228**	256	344
100	**288**	320	472
128	**372**	400	632
192	**556**	608	1,048
256	**748**	816	1,424

Table 6.17: Signature sizes (in bit) of Rainbow schemes over different fields for different levels of security

6.5.2 Signature Lengths

The Tables 6.16 and 6.17 compare Rainbow schemes over GF(16), GF(31) and GF(256) in terms of the signature length.

As we see from the tables, we get the shortest signatures when using Rainbow over GF(16). These signatures are about 10 to 20 percent shorter than the signatures one gets when using Rainbow over GF(31). The signatures of Rainbow schemes over GF(256) are much longer and this difference in length will increase over time.

Part II

Reducing Key Sizes for Multivariate Schemes

One of the biggest problems of multivariate cryptosystems is the large size of their public (and private) keys. In this second part of the thesis we present an approach to reduce the public key size of the UOV and Rainbow signature schemes without weakening the security of the schemes. We achieve our results by generating UOV and Rainbow instances with structured public keys. Unfortunately it does not seem to be possible to generate a UOV or Rainbow scheme with completely structured public key. The Macauley matrix M_P of the public key $\mathcal{P}$ of our schemes therefore consists out of two parts B and C, i.e. we have $M_P = (B|C)$ with a structured matrix B and a matrix C without visible structure. The structure inside the matrix B enables us to give a compact representation of this matrix and therefore of the public key. This leads to a significant reduction of the public key size of factors of up to 8.0 (UOV) and 3.1 (Rainbow). Furthermore, we can use our approach to get some results about "provable security" of the UOV scheme. The structure of the matrix B can also be used to speed up the verification process of our schemes. We achieve here a speed up of factors of up to 6.1 and 2.4 for UOV and Rainbow respectively. By applying our technique to the QUAD stream cipher, we can increase the data throughput of QUAD by a factor of 6.8.

In the first chapter of this part (Chapter 7) we present our general approach to generate UOV and Rainbow instances with structured public keys. In Chapter 8 we describe several improved versions of the UOV signature scheme which reduce the public key size of the original scheme by large factors. We discuss the security of these schemes and compare our schemes with the original UOV signature scheme in terms of the public key size. Furthermore we present in this chapter a scheme called UOVrand, which offers some kind of provable security and give details about the implementation of our schemes. Chapter 9 presents two improved versions of the Rainbow signature scheme which reduce the public key size of the original scheme by large factors. We analyze the efficiency of the known attacks on Rainbow against our improved versions and give details about the implementation of our schemes. In Chapter 10 we show how the structure in the public key of our improved versions of UOV and Rainbow can be used to speed up the verification process of the signature schemes. The analysis is done both theoretically and experimentally using C implementations of our schemes. Finally, Chapter 11 applies our techniques to the QUAD stream cipher and shows how the key stream generation process of QUAD can be sped up by using structured polynomials. We discuss the security of these constructions and present the results of our computer experiments.

Chapter 7

The General Approach

In this chapter we present our general approach to reduce the public key size of the UOV and Rainbow signature schemes. We achieve this reduction by inserting a structured matrix B into the Macauley matrix M_P of the public key. Our construction allows the user to fix a major part of the public key and to compute out of it the central map of the scheme (see Figures 7.3 and 7.6). Section 7.1 shows, how this can be done for the UOV signature scheme and in Section 7.2 we describe how our technique can be extended to Rainbow.

7.1 UOV

Recall that the public key of the UOV signature scheme is given by

$$\mathcal{P} = \mathcal{F} \circ \mathcal{T}, \tag{7.1}$$

where $\mathcal{F}$ is a UOV central map and $\mathcal{T}$ is a randomly chosen affine invertible map (given by an $n \times n$ matrix $M_T = (t_{ij})_{i,j=1}^{n}$ and a vector $c_T \in \mathbb{F}^n$).
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

$$f_{ij}^{(k)} = 0 \ \forall i \in O \wedge j \in O, 1 \leq k \leq o \Leftrightarrow v+1 \leq i \leq j \leq n, 1 \leq k \leq o. \tag{7.2}$$

The key observation for our construction is the following. Equation (7.1) implies

$$p_{ij}^{(k)} = \sum_{r=1}^{n}\sum_{s=r}^{n} \alpha_{ij}^{rs} \cdot f_{rs}^{(k)} \overset{(7.2)}{=} \sum_{r=1}^{v}\sum_{s=r}^{n} \alpha_{ij}^{rs} \cdot f_{rs}^{(k)} \quad (1 \leq i \leq j \leq n, 1 \leq k \leq o) \tag{7.3}$$

with

$$\alpha_{ij}^{rs} = \begin{cases} t_{ri} \cdot t_{si} & (i = j) \\ t_{ri} \cdot t_{sj} + t_{rj} \cdot t_{si} & \text{otherwise} \end{cases}. \tag{7.4}$$

After fixing the elements of the matrix M_T to some random values of $\mathbb{F}$, equation (7.3) 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

- $D := \frac{v \cdot (v+1)}{2} + o \cdot v$ be the number of non zero quadratic terms in the components of $\mathcal{F}$ and
- $D' := \frac{n \cdot (n+1)}{2}$ be the number of quadratic terms in the public polynomials.

Let M_P and M_F be the Macauley matrices of $\mathcal{P}$ and $\mathcal{F}$ respectively (w.r.t. the graded lexicographic ordering of monomials, see Definition 2.6). We divide the matrices M_P and M_F into submatrices as shown in Figure 7.1. Note that, due to the absence of oil $\times$ oil terms in the central polynomials, we have a block of zeros in the middle of M_F (c.f. equation (7.2)).

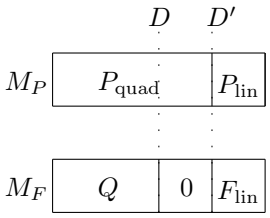


Figure 7.1: Layout of the matrices M_P and M_F

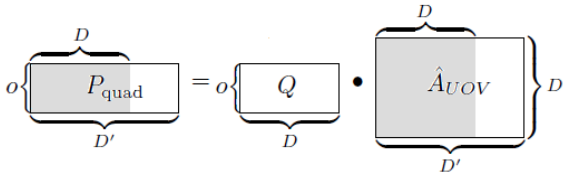


Figure 7.2: Sizes of the matrices P_{quad}, Q and $\hat{A}_{\text{UOV}}$

Furthermore we define a transformation matrix $\hat{A}_{\text{UOV}} \in \mathbb{F}^{D \times D'}$ containing the coefficients α_{ij}^{rs} of equation (7.3) by

$$\hat{A}_{\text{UOV}} = \left(\alpha_{ij}^{rs}\right) \ (1 \le r \le v, r \le s \le n \text{ for the rows, } 1 \le i \le j \le n \text{ for the columns}), \text{i.e.}$$

$$\hat{A}_{\text{UOV}} = \begin{pmatrix} \alpha_{11}^{11} & \alpha_{12}^{11} & \dots & \alpha_{nn}^{11} \\ \alpha_{11}^{12} & \alpha_{12}^{12} & \dots & \alpha_{nn}^{12} \\ \vdots & & & \vdots \\ \alpha_{11}^{vn} & \alpha_{12}^{vn} & \dots & \alpha_{nn}^{vn} \end{pmatrix}. \tag{7.5}$$

For both rows and columns, the elements of $\hat{A}_{\text{UOV}}$ are ordered with respect to the graded lexicographic order.
With this notation, equation (7.3) yields

$$P_{\text{quad}} = Q \cdot \hat{A}_{\text{UOV}} \tag{7.6}$$

(see Figure 7.2). We call the column of $\hat{A}_{\text{UOV}}$ containing the entries α_{ij}^{rs} $(1 \le r \le v,\ r \le s \le n)$ the (i,j)-th column of $\hat{A}_{\text{UOV}}$. Analogously we call the column of P_{quad} containing the coefficients $p_{ij}^{(k)}$ $(k = 1, \dots, o)$ the (i,j)-th column of P_{quad}.
In order to solve equation (7.6) for Q, we define for an index set I containing D of the pairs (i,j) $(1 \le i \le j \le n)$

- $B := P_{\text{quad}}|_I \in \mathbb{F}^{o \times D}$ contains the (i,j)-th column of $P_{\text{quad}} \Leftrightarrow (i,j) \in I$ and
- $A_{\text{UOV}} := \hat{A}_{\text{UOV}}|_I \in \mathbb{F}^{D \times D}$ contains the (i,j)-th column of $\hat{A}_{\text{UOV}} \Leftrightarrow (i,j) \in I$.

More information on this step and the choice of I can be found in Subsection 7.1.1.

For the restricted matrices B and A_{UOV} equation (7.6) yields

$$B = Q \cdot A_{\text{UOV}}. \tag{7.7}$$

		GF(16)		GF(31)		GF(256)	
(o,v)		(10,20)	(20,40)	(10,20)	(20,40)	(10,20)	(20,40)
% of invertible	sel. str. 1	93.9	93.7	96.9	96.8	99.7	99.6
matrices A_{UOV}	sel. str. 2	93.8	93.8	96.6	96.8	99.6	99.8

Table 7.1: Percentage of the matrices A_{UOV} being invertible

If the $D \times D$ matrix A_{UOV} is invertible, we can fix the elements of the matrix B and compute the elements of Q by

$$Q = B \cdot A_{\mathrm{UOV}}^{-1}. \tag{7.8}$$

We can then use Algorithm 7.1 to generate a key pair for UOV.

Algorithm 7.1 Alternative Key Generation for UOV

Input: parameters $(\mathbb{F}, o, v)$, matrix $B \in \mathbb{F}^{o \times D}$, selection strategy (given by an index set I)
Output: UOV key pair $((\mathcal{F}, \mathcal{T}), \mathcal{P})$

1: Choose randomly an affine map $\mathcal{T}$ (represented by an $n \times n$-matrix M_T and an n-vector c_T). If M_T is not invertible, choose again.
2: Compute for $\mathcal{T}$ the corresponding transformation matrix A_{UOV} (using equations (7.4) and (7.5) and the given selection strategy). If A_{UOV} is not invertible, go back to line 1.
3: Compute the matrix Q containing the quadratic coefficients of the central polynomials by equation (7.8).
4: Choose the linear and constant terms of the central map $\mathcal{F}$ at random.
5: Compute the public key as $\mathcal{P} = \mathcal{F} \circ \mathcal{T}$.
6: **return** $((\mathcal{F}, \mathcal{T}), \mathcal{P})$

7.1.1 Selection Strategies

In this subsection we look at different possibilities to define the index set I used in Algorithm 7.1. The two selection strategies we use in this thesis can be described as follows:

1. The first possibility (used e.g. for cyclicUOV, see Section 8.1) is to choose the index set I as

$$I_1 = \{(i, j) : 1 \leq i \leq v, i \leq j \leq n\}. \tag{7.9}$$

   This selection strategy has the nice property that the monomials $x_i x_j$ ($(i, j) \in I_1$) are the first monomials with respect to the graded lexicographic order. So the matrix B consists exactly of the first D columns of the matrix P_{quad}. The grey parts in Figure 7.2 show the matrices B and A_{UOV} for this selection strategy.

2. The second possibility (used for 0/1UOV, see Section 8.6) is to choose the index set I as

$$I_2 = \{(i, j) : (i, j) \in E\}, \tag{7.10}$$

   where E with $|E| = D$ is the set of edges of a specially designed graph G. In Section 8.6 we use for G the Turán graph. By this choice we can ensure that each $k \in \{1, \ldots, n\}$ appears in the elements $(i, j) \in I_2$ approximately the same number of times. We need this fact later to show the security of the 0/1UOV scheme. More details on this issue can be found in Section 8.6.

Table 7.1 shows that, for both of the above choices of I, we get an invertible matrix A_{UOV} with overwhelming probability. The values presented in the table are close to the theoretical probability

of a $D \times D$ matrix being invertible of

$$\prod_{i=0}^{D-1} \left(1 - q^{i-D}\right). \tag{7.11}$$

Remark 7.1. *The large percentage of invertible matrices A_{UOV} guarantees the functioning of Algorithm 7.1 for the selection strategies 1 and 2. In particular, one only needs very few trials to pass the test in line 2 of the algorithm.*

Surprisingly enough there exist selection strategies which seem to lead always to singular matrices A_{UOV}.

Claim 7.1. *Let I_3 be the set containing the pairs (i,j) $(1 \leq i \leq j \leq n)$ corresponding to the D smallest quadratic monomials with respect to the graded lexicographic order. Then the matrix $A_{\mathrm{UOV}} = \hat{A}_{\mathrm{UOV}}|_{I_3}$ is singular.*

Proposition 7.1 proves this claim for the (balanced) case $o = v = 2$.

Proposition 7.1. *For the (balanced) case $o = v = 2$ the matrix $\bar{A}_{\mathrm{UOV}}$ consisting of the last D columns of $\hat{A}_{\mathrm{UOV}}$ is singular.*

Proof. We show even more, namely that the last $D-1$ columns of the $D \times D'$ matrix $\hat{A}_{\mathrm{UOV}}$ are linearly dependent. Let $M_T = (t_{ij})_{i,j=1}^4 \in \mathbb{F}^{4\times 4}$ be the matrix of the invertible affine map $\mathcal{T}$. Then the matrix $\tilde{A}_{\mathrm{UOV}}$ consisting of the last $D - 1 = 6$ columns of $\hat{A}_{\mathrm{UOV}}$ has the form

$$\tilde{A}_{\mathrm{UOV}} = \begin{pmatrix} t_{12}^2 & 0 & 0 & t_{13}^2 & 0 & t_{14}^2 \\ t_{12}\cdot t_{22} & t_{12}\cdot t_{23} + t_{13}\cdot t_{22} & t_{12}\cdot t_{24} + t_{14}\cdot t_{22} & t_{13}\cdot t_{23} & t_{13}\cdot t_{24} + t_{14}\cdot t_{23} & t_{14}\cdot t_{24} \\ t_{12}\cdot t_{32} & t_{12}\cdot t_{33} + t_{13}\cdot t_{32} & t_{12}\cdot t_{34} + t_{14}\cdot t_{32} & t_{13}\cdot t_{33} & t_{13}\cdot t_{34} + t_{14}\cdot t_{33} & t_{14}\cdot t_{34} \\ t_{12}\cdot t_{42} & t_{12}\cdot t_{43} + t_{13}\cdot t_{42} & t_{12}\cdot t_{44} + t_{14}\cdot t_{42} & t_{13}\cdot t_{43} & t_{13}\cdot t_{44} + t_{14}\cdot t_{43} & t_{14}\cdot t_{44} \\ t_{22}^2 & 0 & 0 & t_{23}^2 & 0 & t_{24}^2 \\ t_{22}\cdot t_{32} & t_{22}\cdot t_{33} + t_{23}\cdot t_{32} & t_{22}\cdot t_{34} + t_{24}\cdot t_{32} & t_{23}\cdot t_{33} & t_{23}\cdot t_{34} + t_{24}\cdot t_{33} & t_{24}\cdot t_{34} \\ t_{22}\cdot t_{42} & t_{22}\cdot t_{43} + t_{23}\cdot t_{42} & t_{22}\cdot t_{44} + t_{24}\cdot t_{42} & t_{23}\cdot t_{43} & t_{23}\cdot t_{44} + t_{24}\cdot t_{43} & t_{24}\cdot t_{44} \end{pmatrix}.$$

We show that the rank of this matrix is only 5, i.e. the last column of $\tilde{A}_{\mathrm{UOV}}$ is a linear combination of the first 5 columns. Using the row operations of the Gauss Algorithm (without swapping rows), we can bring $\tilde{A}_{\mathrm{UOV}}$ to the form

$$\tilde{\tilde{A}}_{\mathrm{UOV}} = \begin{pmatrix} \star & 0 & 0 & \star & 0 & \star \\ 0 & \star & \star & \star & \star & \star \\ 0 & 0 & \star & 0 & \star & \star \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \star & \star & \star \\ 0 & 0 & 0 & 0 & \star & \star \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

The computations in this step were performed with MAGMA and the values of "$\star$" can be very complicated. For example, $\tilde{\tilde{A}}_{\mathrm{UOV}}[6][6]$ consists of 92 terms of degree up to 32. □

The claim 7.1 seems to be true for arbitrary values of o and v. In fact, we have proven it for $1 \leq o \leq 50$, $v = 2 \cdot o$ (using MAGMA).
Therefore the selection strategy induced by the set I_3 is not suitable for our purposes.

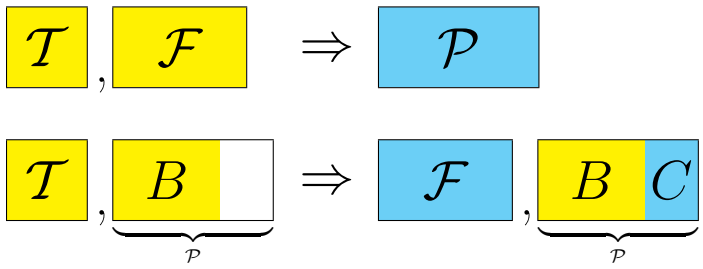


Figure 7.3: Standard key generation (above) and alternative key generation for UOV. The yellow parts are chosen by the user, the blue parts are computed during the key generation process.

Remark 7.2. *Due to the symmetry between $\mathcal{F}$ and $\mathcal{P}$ (c.f. equations* (3.10) *and* (3.11)*) we get a relation between the coefficients $f_{ij}^{(k)}$ and $p_{ij}^{(k)}$ of the form*

$$f_{ij}^{(k)} = \sum_{r=1}^{n} \sum_{s=r}^{n} \widetilde{\alpha}_{ij}^{rs} \cdot p_{rs}^{(k)} \tag{7.12}$$

with

$$\widetilde{\alpha}_{ij}^{rs} = \begin{cases} \tilde{t}_{ri} \cdot \tilde{t}_{si} & (i = j) \\ \tilde{t}_{ri} \cdot \tilde{t}_{sj} + \tilde{t}_{rj} \cdot \tilde{t}_{si} & \text{otherwise} \end{cases} . \tag{7.13}$$

where $\tilde{t}_{ij}$ $(i, j = 1, \ldots, n)$ are the elements of the matrix $\widetilde{M}_T = M_T^{-1}$.
However, since some of the coefficients $f_{ij}^{(k)}$ are fixed to zero (c.f. equation (7.2)*), some of the relations of equation* (7.12) *lead, after fixing the coefficients $p_{ij}^{(k)}$, to contradictions. Therefore, we can not use equation* (7.12) *for our purposes.*
On the other hand, equation (7.12) *is used in cryptanalysis as a starting point for the UOV-Reconciliation attack (see Subsection 3.3.2).*

By applying Algorithm 7.1 we can insert arbitrary matrices B into the Macauley matrix of the public key. In Chapter 8 we will discuss some specific choices of this matrix and their effects on the public key size of UOV.

Remark 7.3. *For affine maps $\mathcal{T}$ which lead, via a given selection strategy I, to invertible matrices A_{UOV} the standard key generation process of UOV as described in Section 3.1 and the alternative key generation as described by Algorithm 7.1 are equivalent: Let a selection strategy I be fixed and let $\mathcal{T}$ be an affine map which leads (together with the selection strategy I) to an invertible matrix A_{UOV}. Then we have, due to equation* (7.7) *a one to one relation between the matrices B and Q. In the standard construction, the matrix Q is fixed (as part of the central map $\mathcal{F}$) and leads via the relation $\mathcal{P} = \mathcal{F} \circ \mathcal{T}$ to a uniquely determined public key $\mathcal{P}$. In our construction, we fix the matrix B (as part of the public key) and get due to equation* (7.8) *a uniquely determined matrix Q. After choosing the linear and constant terms of $\mathcal{F}$, we get due to the relation $\mathcal{P} = \mathcal{F} \circ \mathcal{T}$ a uniquely determined public key. Figure 7.3 shows a graphical illustration of the two possibilities of the key generation process.*

7.2 Rainbow

In this section we describe how to insert a given matrix B into the Macauley matrix of a Rainbow public key. We do this by introducing a new map $\mathcal{Q} = \mathcal{F} \circ \mathcal{T}$ and applying the technique presented in the previous section to each Rainbow layer separately.

7.2.1 Preliminaries

To simplify the notation in this section, we introduce some constants and a specially designed ordering of monomials. We denote

- $D_0 := 0$
- $D_1 := \frac{v_1 \cdot (v_1+1)}{2} + o_1 \cdot v_1$ is the number of non zero quadratic terms in the central polynomials of the first layer.
- $D_2 := \frac{v_2 \cdot (v_2+1)}{2} + o_2 \cdot v_2$ is the number of non zero quadratic terms in the central polynomials of the second layer.
  ...
- $D_\ell := \frac{v_\ell \cdot (v_\ell+1)}{2} + o_\ell \cdot v_\ell$ is the number of non zero quadratic terms in the central polynomials of the ℓ-th layer.
  ...
- $D_u := \frac{v_u \cdot (v_u+1)}{2} + o_u \cdot v_u$ is the number of non zero quadratic terms in the central polynomials of the u-th layer.
- $D := \frac{n \cdot (n+1)}{2}$ is the number of quadratic terms in the public polynomials.

The monomial ordering is defined as follows:

- The first block contains the D_1 quadratic monomials $x_i x_j$ appearing in the central polynomials of the first Rainbow layer (i.e. the monomials $x_i x_j$ $(1 \leq i \leq v_1,\ i \leq j \leq v_2)$).
- The second block contains the $D_2 - D_1$ quadratic monomials $x_i x_j$ appearing in the central polynomials of the second, but not in those of the first Rainbow layer
  (i.e. the monomials $x_i x_j$ $(1 \leq i \leq v_1,\ v_2 < j \leq v_3 \vee v_1 < i \leq v_2, i \leq j \leq v_3)$).
  ...
- The ℓ-th block contains the $D_\ell - D_{\ell-1}$ quadratic monomials $x_i x_j$ appearing in the central polynomials of the ℓ-th but not in those of the $(\ell-1)$-th Rainbow layer
  (i.e. the monomials $x_i x_j$ $(1 \leq i \leq v_{\ell-1},\ v_\ell < j \leq v_{\ell+1} \vee v_{\ell-1} < i \leq v_\ell, i \leq j \leq v_{\ell+1})$).
  ...
- The u-th block contains the $D_u - D_{u-1}$ quadratic monomials $x_i x_j$ appearing in the central polynomials of the u-th but not in those of the $(u-1)$-th Rainbow layer
  (i.e. the monomials $x_i x_j$ $(1 \leq i \leq v_{u-1},\ v_u < j \leq n \vee v_{u-1} < i \leq v_u, i \leq j \leq v_n)$).
- The $(u+1)$-th block contains the remaining $D - D_u$ quadratic monomials (i.e. the monomials $x_i x_j$ $(v_u \leq i \leq j \leq n)$.
- The $(u+2)$-th block contains the linear and constant monomials.

Inside the blocks we use the lexicographic order.

Example 7.1. *For* $(v_1, o_1, o_2) = (2, 2, 2)$ *we get*

- $D_1 = \frac{v_1 \cdot (v_1+1)}{2} + o_1 \cdot v_1 = 7$,
- $D_2 = \frac{v_2 \cdot (v_2+1)}{2} + v_2 \cdot o_2 = 18$,
- $D = \frac{n \cdot (n+1)}{2} = 21$

and the following ordering of monomials on $\mathbb{F}[x_1, x_2, x_3, x_4, x_5, x_6]$*:*
$x_1^2 > x_1x_2 > x_1x_3 > x_1x_4 > x_2^2 > x_2x_3 > x_2x_4 > x_1x_5 > x_1x_6 > x_2x_5 > x_2x_6 > x_3^2 > x_3x_4 > x_3x_5 > x_3x_6 > x_4^2 > x_4x_5 > x_4x_6 > x_5^2 > x_5x_6 > x_6^2 > x_1 > x_2 > x_3 > x_4 > x_5 > x_6 > 1$. □

7.2.2 Properties of the Rainbow Public Key

For the Rainbow signature scheme (see Section 3.4) the public key $\mathcal{P}$ is given as the composition of three maps

$$\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T} \tag{7.14}$$

with affine maps $\mathcal{S} = (M_S, c_S)$ and $\mathcal{T} = (M_T, c_T)$ and a Rainbow central map $\mathcal{F}$. We define

$$\mathcal{Q} = \mathcal{F} \circ \mathcal{T} \tag{7.15}$$

and get

$$\mathcal{P} = \mathcal{S} \circ \mathcal{Q}. \tag{7.16}$$

Note that the relation between the maps $\mathcal{Q}$ and $\mathcal{F}$ has the same form as the relation between a public key and a central map in the UOV case. Therefore we get exactly the same equations as in Section 7.1. We have

$$q_{ij}^{(k)} = \sum_{r=1}^{n} \sum_{s=r}^{n} \alpha_{ij}^{rs} \cdot f_{rs}^{(k)} \ (v_1 + 1 \leq k \leq n, 1 \leq i \leq j \leq n), \tag{7.17}$$

where $q_{ij}^{(k)}$ and $f_{ij}^{(k)}$ are the coefficients of the monomial x_ix_j in the k-th component of $\mathcal{Q}$ and $\mathcal{F}$ respectively and the coefficients α_{ij}^{rs} are given as

$$\alpha_{ij}^{rs} = \begin{cases} t_{ri} \cdot t_{si} & (i = j) \\ t_{ri} \cdot t_{sj} + t_{rj} \cdot t_{si} & \text{otherwise} \end{cases} \tag{7.18}$$

with t_{ij} $(1 \leq i, j \leq n)$ being the elements of the matrix M_T.
Between the coefficients of the public key $\mathcal{P}$ and the map $\mathcal{Q}$ we get

$$p_{ij}^{(k)} = \sum_{l=v_1+1}^{n} s_{k-v_1, l-v_1} \cdot q_{ij}^{(l)} \ (1 \leq i \leq j \leq n, \ v_1 + 1 \leq k \leq n) \tag{7.19}$$

with s_{kl} $(1 \leq k, l \leq m)$ being the elements of the matrix M_S.
Due to the special structure of the central map $\mathcal{F}$ (see Figure 7.5), we can reduce the number of terms in equation (7.17). We get

$$\begin{aligned} q_{ij}^{(k)} &= \sum_{r=1}^{v_1} \sum_{s=r}^{v_2} \alpha_{ij}^{rs} \cdot f_{rs}^{(k)} \ (k \in O_1) \\ q_{ij}^{(k)} &= \sum_{r=1}^{v_2} \sum_{s=r}^{v_3} \alpha_{ij}^{rs} \cdot f_{rs}^{(k)} \ (k \in O_2) \\ &\vdots \\ q_{ij}^{(k)} &= \sum_{r=1}^{v_u} \sum_{s=r}^{n} \alpha_{ij}^{rs} \cdot f_{rs}^{(k)} \ (k \in O_u). \end{aligned} \tag{7.20}$$

$$
A_{\text{Rainbow}} = \begin{array}{c}
\begin{array}{cccccccccc} & & & & D_1 & & & D_2 & \dots & D \end{array} \\
\left(\begin{array}{cccccccccc}
\alpha_{11}^{11} & \alpha_{12}^{11} & \dots & \alpha_{v_1v_2}^{11} & \alpha_{1,v_2+1}^{11} & \dots & \alpha_{v_2,v_3}^{11} & \alpha_{1,v_3+1}^{11} & \dots & \alpha_{n,n}^{11} \\
\alpha_{11}^{12} & \alpha_{12}^{12} & \dots & \alpha_{v_1v_2}^{12} & \alpha_{1,v_2+1}^{12} & \dots & \alpha_{v_2,v_3}^{12} & \alpha_{1,v_3+1}^{12} & \dots & \alpha_{n,n}^{12} \\
\vdots & & & & & & & & & \vdots \\
\alpha_{11}^{v_1v_2} & \alpha_{12}^{v_1v_2} & \dots & \alpha_{v_1v_2}^{v_1v_2} & \alpha_{1,v_2+1}^{v_1v_2} & \dots & \alpha_{v_2,v_3}^{v_1v_2} & \alpha_{1,v_3+1}^{v_1v_2} & \dots & \alpha_{n,n}^{v_1v_2} \\
\hline
\alpha_{11}^{1,v_2+1} & \alpha_{12}^{1,v_2+1} & \dots & \alpha_{v_1v_2}^{1,v_2+1} & \alpha_{1,v_2+1}^{1,v_2+1} & \dots & \alpha_{v_2,v_3}^{1,v_2+1} & \alpha_{1,v_3+1}^{1,v_2+1} & \dots & \alpha_{n,n}^{1,v_2+1} \\
\vdots & & & & & & & & & \vdots \\
\alpha_{11}^{v_2v_3} & \alpha_{12}^{v_2v_3} & \dots & \alpha_{v_1v_2}^{v_2v_3} & \alpha_{1,v_2+1}^{v_2v_3} & \dots & \alpha_{v_2,v_3}^{v_2,v_3} & \alpha_{1,v_3+1}^{v_2,v_3} & \dots & \alpha_{n,n}^{v_2,v_3} \\
\hline
\alpha_{11}^{1,v_3+1} & \alpha_{12}^{1,v_3+1} & \dots & \alpha_{v_1v_2}^{1,v_3+1} & \alpha_{1,v_2+1}^{1,v_3+1} & \dots & \alpha_{v_2,v_3}^{1,v_3+1} & \alpha_{1,v_3+1}^{1,v_3+1} & \dots & \alpha_{n,n}^{1,v_3+1} \\
\vdots & & & & & & & & & \vdots \\
\alpha_{11}^{n,n} & \alpha_{12}^{n,n} & \dots & \alpha_{v_1v_2}^{n,n} & \alpha_{1,v_2+1}^{n,n} & \dots & \alpha_{v_2,v_3}^{n,n} & \alpha_{1,v_3+1}^{n,n} & \dots & \alpha_{n,n}^{n,n}
\end{array}\right)
\begin{array}{l} \\ \\ \\ D_1 \\ \\ \\ D_2 \\ \\ \\ \vdots \\ \\ \\ D \end{array}
\end{array}
$$

Figure 7.4: Structure of the matrix A_{Rb}

To write equation (7.20) in a compact form, we define a quadratic $D \times D$ matrix A_{Rainbow} containing the coefficients α_{ij}^{rs} of equation (7.20) by

$$A_{\text{Rainbow}} = \left(\alpha_{ij}^{rs}\right) \ (1 \leq r \leq s \leq n \text{ for the rows, } 1 \leq i \leq j \leq n \text{ for the columns}) \tag{7.21}$$

(see Figure 7.4). The order in which the α_{ij}^{rs} appear in the matrix A_{Rainbow}, is thereby given by the monomial ordering defined in Subsection 7.2.1 (for both rows and columns).

Let M_P, M_Q and M_F be the Macauley matrices of $\mathcal{P}$, $\mathcal{Q}$ and $\mathcal{F}$ respectively (with respect to the monomial ordering defined in Subsection 7.2.1). As in the case of UOV we denote the submatrices containing the coefficients of quadratic terms by P_{quad}, Q_{quad} and F_{quad} respectively.

With this notation, equations (7.19) and (7.20) yield

$$P_{\text{quad}} = M_S \cdot Q_{\text{quad}} \tag{7.22}$$

and

$$Q_{\text{quad}} = F_{\text{quad}} \cdot A_{\text{Rainbow}}. \tag{7.23}$$

7.2.3 Construction

Similarly to the case of UOV, the equations (7.22) and (7.23) allow us to fix the coefficients of some of the quadratic terms of the public polynomials. In particular, we can fix

- D_1 coefficients of the quadratic terms of each polynomial of the first layer
- D_2 coefficients of the quadratic terms of each polynomial of the second layer

  ...
- D_u coefficients of the quadratic terms of each polynomial of the u-th layer.

In the following, we select the coefficients to be fixed in such a way that they correspond to the non zero coefficients of the central map $\mathcal{F}$.
We divide the matrices M_P, M_Q and M_F into submatrices as shown in Figure 7.5. The selected coefficients of M_P are located in the grey parts of the matrix. The grey parts in the matrix M_F mark the non zero coefficients of quadratic monomials in the map $\mathcal{F}$.

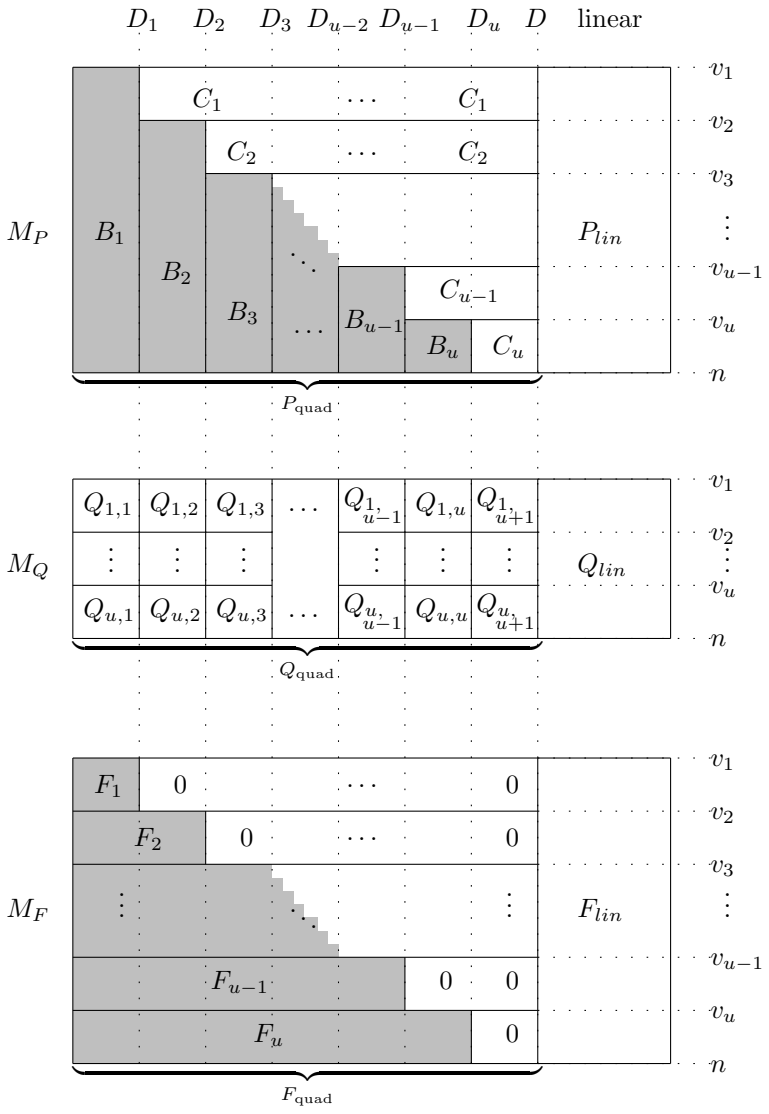


Figure 7.5: Layout of the matrices M_P, M_Q and M_F

Similarly to M_P, M_Q and M_F, we divide the matrices M_S and A_{Rainbow} into submatrices. We write

$$M_S = \begin{pmatrix} S_{1,1} & S_{1,2} & \dots & S_{1,u} \\ S_{2,1} & S_{2,2} & & S_{2,u} \\ \vdots & & & \vdots \\ S_{u,1} & S_{u,2} & \dots & S_{u,u} \end{pmatrix}, \tag{7.24}$$

where $S_{i,j}$ is an $o_i \times o_j$ matrix $(i, j = 1, \dots, u)$ and

$$A_{\text{Rainbowb}} = \begin{pmatrix} A_{1,1} & A_{1,2} & \dots & A_{1,u+1} \\ A_{2,1} & A_{2,2} & & A_{2,u+1} \\ \vdots & & & \vdots \\ A_{u+1,1} & A_{u+1,2} & \dots & A_{u+1,u+1} \end{pmatrix}, \tag{7.25}$$

where $A_{i,j}$ is a $(D_i - D_{i-1}) \times (D_j - D_{j-1})$ matrix $(i, j = 1, \dots, u+1)$. Here we set $D_{u+1} := D$. Therewith, equation (7.22) yields

$$B_i = \begin{pmatrix} S_{i,1} & \dots & S_{i,u} \\ \vdots & & \vdots \\ S_{u,1} & \dots & S_{u,u} \end{pmatrix} \cdot \begin{pmatrix} Q_{1,i} \\ \vdots \\ Q_{u,i} \end{pmatrix} = \underbrace{\begin{pmatrix} S_{i,i} & \dots & S_{i,u} \\ \vdots & & \vdots \\ S_{u,i} & \dots & S_{u,u} \end{pmatrix}}_{\tilde{S}_i} \cdot \begin{pmatrix} Q_{i,i} \\ \vdots \\ Q_{u,i} \end{pmatrix} + \sum_{j=1}^{i-1} \begin{pmatrix} S_{i,j} \\ \vdots \\ S_{u,j} \end{pmatrix} \cdot Q_{j,i} \tag{7.26}$$

Using the relation (7.23), the i-th row of Q_{quad} [1] can be computed as

$$Q_{\text{quad}}[i] = F_{\text{quad}}[i] \cdot A_{\text{Rb}} = F_i \cdot \begin{pmatrix} A_{1,1} & \dots & A_{1,u+1} \\ \vdots & & \vdots \\ A_{i,1} & \dots & A_{i,u+1} \end{pmatrix}. \tag{7.27}$$

We obtain

$$(Q_{i1} || \dots || Q_{ii}) = F_i \cdot \underbrace{\begin{pmatrix} A_{1,1} & \dots & A_{1,i} \\ \vdots & & \vdots \\ A_{i,1} & \dots & A_{i,i} \end{pmatrix}}_{\tilde{A}_i} \tag{7.28}$$

and

$$(Q_{i,i+1} || \dots || Q_{i,u+1}) = F_i \cdot \begin{pmatrix} A_{1,i+1} & \dots & A_{1,u+1} \\ \vdots & & \vdots \\ A_{i,i+1} & \dots & A_{i,u+1} \end{pmatrix}. \tag{7.29}$$

If all the matrices $\tilde{S}_i$ and $\tilde{A}_i$ $(i = 1, \dots, u)$ are invertible, we can use equations (7.26), (7.28) and (7.29) to generate a key pair for Rainbow (see Algorithm 7.2).

[1] Here we denote $Q_{\text{quad}}[i] = (Q_{i,1} || \dots || Q_{i,u+1})$ (see Figure 7.5). $F_{\text{quad}}[i]$ is defined analogously.

Algorithm 7.2 Alternative Key Generation for Rainbow

Input: parameters $(\mathbb{F}, v_1, o_1, \ldots, o_u)$, matrices $B_i \in \mathbb{F}^{(n-v_i)\times(D_i-D_{i-1})}$ $(i = 1, \ldots, u)$
Output: Rainbow key pair $((\mathcal{S}, \mathcal{F}, \mathcal{T}), \mathcal{P})$

1: Choose randomly an affine map $\mathcal{S} : \mathbb{F}^m \to \mathbb{F}^m$ (given as a matrix $M_S \in \mathbb{F}^{m\times m}$ and a vector $c_S \in \mathbb{F}^m$). If one of the submatrices $\tilde{S}_i$ $(i = 1, \ldots, u)$ of M_S (see equation (7.26)) is singular, choose again.
2: Choose randomly an affine map $\mathcal{T} : \mathbb{F}^n \to \mathbb{F}^n$ (given as a matrix $M_T \in \mathbb{F}^{n\times n}$ and a vector $c_T \in \mathbb{F}^n$). If the matrix M_T is singular, choose again.
3: Compute for M_T the corresponding transformation matrix A_{Rainbow} (using equations (7.18) and (7.21)). If one of the submatrices $\tilde{A}_i$ $(i = 1, \ldots, u)$ (see equation 7.28) is singular, go back to step 2.
4: **for** $i = 1$ to u **do**
5: Compute $\begin{pmatrix} Q_{i,i} \\ \vdots \\ Q_{u,i} \end{pmatrix} = \begin{pmatrix} S_{i,i} & \ldots & S_{i,u} \\ \vdots & & \vdots \\ S_{u,i} & \ldots & S_{u,u} \end{pmatrix}^{-1} \cdot \left(B_i - \sum_{j=1}^{i-1} \begin{pmatrix} S_{i,j} \\ \vdots \\ S_{u,j} \end{pmatrix} \cdot Q_{j,i} \right)$.
6: Compute $F_i = (Q_{i,1}|| \ldots ||Q_{i,i}) \cdot \begin{pmatrix} A_{1,1} & \ldots & A_{i,1} \\ \vdots & & \vdots \\ A_{1,i} & \ldots & A_{i,i} \end{pmatrix}^{-1}$.
7: **if** $i < u$ **then**
8: Compute $(Q_{i,i+1}|| \ldots ||Q_{i,u+1}) = F_i \cdot \begin{pmatrix} A_{1,i+1} & \ldots & A_{1,u+1} \\ \vdots & & \vdots \\ A_{i,i+1} & \ldots & A_{i,u+1} \end{pmatrix}$.
9: **end if**
10: **end for**
11: Choose the linear and constant terms of the central polynomials at random.
12: Compute the public key by $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$.
13: **return** $((\mathcal{S}, \mathcal{F}, \mathcal{T}), \mathcal{P})$

For the functioning of Algorithm 7.2 it is important that the tests in line 1, 2 and 3 of the algorithm are fulfilled with high probability. Therefore we have to ensure that the matrices M_T, $\tilde{S}_i$ and $\tilde{A}_i$ $(i = 1, \ldots, u)$ are invertible in a large number of cases.
As the elements of the matrix M_S are randomly chosen field elements, the probability of the $(n - v_{i-1}) \times (n - v_{i-1})$ matrix $\tilde{S}_i$ being invertible is given by

$$\prod_{j=0}^{n-v_{i-1}-1} \left(1 - q^{j-n+v_{i-1}}\right) \tag{7.30}$$

with q being the cardinality of the underlying field. Here we set $v_0 = 0$.
The probability of all matrices $\tilde{S}_i$ $(i = 1, \ldots, u)$ being invertible is therefore given by

$$\prod_{i=1}^{u} \prod_{j=0}^{n-v_{i-1}-1} \left(1 - q^{j-n+v_{i-1}}\right). \tag{7.31}$$

For the parameters $(\mathbb{F}, v_1, o_1, o_2)$ = (GF(256),17,13,13) this probability is 98.9 %.
Similarly, the probability of the $n \times n$ matrix M_T being invertible is given by

$$\prod_{j=0}^{n-1} (1 - q^{j-n}), \tag{7.32}$$

which for the parameters (GF(256),17,13,13) is 99.6 %.

	GF(16)		GF(31)		GF(256)	
parameters $(v_1, o_1, \ldots, o_u)$	(17,23,17)	(17,12,11,17)	(14,19,14)	(14,10,9,14)	(17,13,13)	(17,7,6,13)
% of cases in which all the matrices $\tilde{A}_i$ $(i = 1, \ldots, u)$ are invertible	87.9	82.5	93.7	90.7	99.2	98.8

Table 7.2: percentage of matrices $\tilde{A}_i$ being invertible

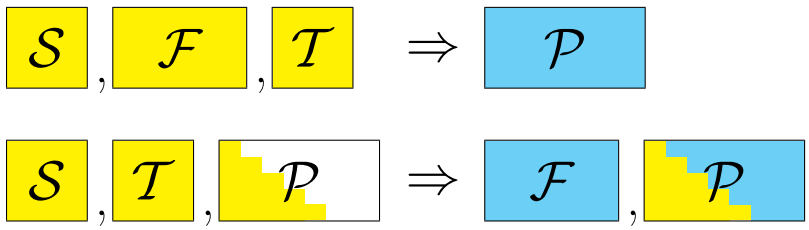


Figure 7.6: Standard key generation (above) and alternative key generation for Rainbow. The yellow parts are chosen by the user, the blue parts are computed during the key generation process.

To estimate the percentage of the test in line 3 of Algorithm 7.2 being fulfilled, we carried out a number of experiments (see Table 7.2). As the table shows, in a large part of all cases all the matrices $\tilde{A}_i$ $(i = 1, \ldots, u)$ are invertible. Again, the values are close to the theoretical results.

As we have seen, the probability that all the tests in line 1 to 3 of Algorithm 7.2 are fulfilled is (at least for big fields) very large. This guarantees the functioning of the algorithm.

By using Algorithm 7.2 we can insert arbitrary matrices $B_1, \ldots, B_u$ into the Macauley matrix of the public key. In Chapter 9 we will discuss some possibilities for the choice of these matrices and their effect on the public key size of Rainbow.

Remark 7.4. *If the affine maps $\mathcal{S}$ and $\mathcal{T}$ lead to invertible matrices $\tilde{S}_i$ and $\tilde{A}_i$ $(i = 1, \ldots, u)$, the standard key generation process of Rainbow (see Section 3.4) and the alternative key generation as described by Algorithm 7.2 are equivalent. This means that we have a one to one relation between the matrices $B_1, \ldots, B_u$ and central maps $\mathcal{F}$ (due to equations* (7.27) - (7.29)*): After fixing the entries of the matrices $B_1, \ldots, B_u$ we get by using Algorithm 7.2 a uniquely determined central map $\mathcal{F}$ and every map $\mathcal{F}$ can be generated in this way. Figure 7.6 shows a graphical illustration of the two possibilities of the key generation process.*

Chapter 8

Improved versions of UOV

In this chapter we present different improved versions of the Unbalanced Oil and Vinegar (UOV) signature scheme which reduce the public key size of the original UOV scheme (see Section 3.1) by large factors. To do this, we follow the approach of Section 7.1 to create a UOV key pair $((\mathcal{F}, \mathcal{T}), \mathcal{P})$ with a structured public key. As shown in the last chapter, the Macauley matrix of $\mathcal{P}$ has the form $M_P = (B|C)$ with a structured matrix B and a matrix C without visible structure. In the following 6 sections we present different possibilities of choosing the matrix B and appropriate selection strategies to generate UOV key pairs with reduced public key size. Section 8.7 discusses the security of these schemes and in Section 8.8 we compare our improved schemes with the original UOV scheme in terms of the public key size. Finally, Section 8.9 gives details about the implementation of our schemes.

As in Section 7.1 we define $D := \frac{v \cdot (v+1)}{2} + o \cdot v$ to be the number of non zero quadratic terms in the polynomials of the central map $\mathcal{F}$.

8.1 CyclicUOV

The first possibility we consider here is to use a partially circulant matrix B. In this case we need to store only the first row of the matrix B and can compute the remaining rows out of this first one. To create a partially circulant $o \times D$ matrix B, we start with a randomly chosen vector $\mathbf{b} = (b_1, \ldots, b_D) \in \mathbb{F}^D$. The i-th row of the matrix B is given by

$$B[i] = \mathcal{R}^{i-1}(\mathbf{b}) \ (i = 1, \ldots, o), \tag{8.1}$$

where $\mathcal{R}^j(\mathbf{b})$ denotes the cyclic right shift of the vector $\mathbf{b}$ by j positions. Algorithm 8.1 shows this generation process in a compact form.

Algorithm 8.1 Generation of a matrix B for cyclicUOV

Input: parameters $(\mathbb{F}, o, v)$
Output: matrix $B \in \mathbb{F}^{o \times D}$ for cyclicUOV

1: Choose randomly a vector $\mathbf{b} \in \mathbb{F}^D$.
2: $B[1] \leftarrow \mathbf{b}$
3: **for** $i = 2$ to o **do**
4: $\quad \mathbf{b} \leftarrow \mathcal{R}(\mathbf{b})$
5: $\quad B[i] \leftarrow \mathbf{b}$
6: **end for**
7: **return** B

To generate a key pair of cyclicUOV, we use this matrix B together with selection strategy 1 (see Subsection 7.1.1) as input for Algorithm 7.1.

To represent a partially circulant $o \times D$ matrix B, we need D field elements. The size of the public key of cyclicUOV is therefore given by

$$\text{size}_{\text{pk cyclicUOV}} = D + o \cdot \left(\frac{o \cdot (o+1)}{2} + n + 1 \right) \tag{8.2}$$

field elements. The size of the private key is the same as for the standard UOV scheme (see equation (3.3)).
Additionally to the key size reduction, the structure in the public key of cyclicUOV enables us to speed up the verification process of the scheme by a large factor (see Section 10.2).

8.2 UOVLRS

In this section we consider the case, where the matrix B is generated by a linear recurring sequence. This approach is motivated by the fact that a partially circulant matrix does not have good statistical properties. Therefore we get a public key which is easily distinguishable from a random public key which might make it possible to develop dedicated attacks against the scheme. Before we start with the description of the UOVLRS scheme, we need some basic facts about randomness properties of sequences and linear recurring sequences (LRS's).

8.2.1 Golomb's Randomness Postulates

In this subsection we look at sequences over a finite field $\mathbb{F}_q$. We cite from [29] some criteria a sequence Σ must fulfill to be considered a random sequence.

Definition 8.1. *Let $\lambda, \eta, \zeta \in \mathbb{F}_q$ with $\lambda \neq \eta$ and $\lambda \neq \zeta$. A subsequence $\bar{\sigma}$ of $\Sigma = \{\sigma_1, \sigma_2, \dots\}$ of the form*

$$\eta, \underbrace{\lambda, \dots, \lambda}_{k \text{ times}}, \zeta$$

is called a run of λ of length k.

Definition 8.2. *Let K be a finite field and L be an extension field of K. Then the function*

$$\text{Tr} : L \to K, \text{Tr}(\alpha) = \sum_{g \in \text{Gal}(L/K)} g(\alpha)$$

is called the trace function from L to K. Here, $\text{Gal}(L/K)$ is the Galois group of the field extension L/K, i.e. the group of all L-automorphisms $g : L \to L$ with $g|_K = id$.

Definition 8.3. *The auto-correlation function of a sequence $\Sigma = \{\sigma_1, \sigma_2, \dots\}$ with period $q^n - 1$ is defined as*

$$\text{AC}_\Sigma(\tau) = \sum_{i=1}^{q^n - 1} \chi(\sigma_i) \cdot \overline{\chi(\sigma_{i+\tau})} \ (0 \leq \tau \leq q^n - 2),$$

where χ is given by

$$\chi(x) = e^{2\pi i \, \text{Tr}(x)}$$

with Tr *being the trace function between $\mathbb{F}_q$ and its prime field $\mathbb{F}_p$ (see Definition 8.2).*

Golomb formulated in [28] three postulates a sequence has to fulfill to be considered a random sequence. Let Σ be a sequence with period $q^n - 1$.

R-1 In every period, every non zero element of $\mathbb{F}_q$ occurs exactly q^{n-1} times and the zero element occurs exactly $q^{n-1} - 1$ times.

R-2 In every period,

1. for $1 \leq k \leq n-2$, the runs of each element $\lambda \in \mathbb{F}_q$ of length k occur exactly $(q-1)^2 \cdot q^{n-k-2}$ times.
2. the runs of each non zero element of $\mathbb{F}_q$ of length $n-1$ occur exactly $q-2$ times.
3. the runs of the zero element of length $n-1$ occur exactly $q-1$ times.
4. the run of every non zero element of length n occurs exactly once.

R-3 The auto-correlation function AC_Σ is two valued with

$$\mathrm{AC}_\Sigma(\tau) = \begin{cases} q^n - 1 & \text{if } \tau \equiv 0 \bmod (q^n - 1) \\ -1 & \text{if } \tau \not\equiv 0 \bmod (q^n - 1) \end{cases}$$

Remark 8.1. *The auto-correlation function AC_Σ measures the amount of similarity between the sequence Σ and its shift by τ positions. Postulate R-3 states that for $\tau \geq 1$ the value $AC_\Sigma(\tau)$ should be quite small.*

Postulate R-1 can be extended as follows

R-4 In every period, each n-tuple $(\lambda_1, \ldots, \lambda_n) \in \mathbb{F}_q^n \setminus \{\mathbf{0}\}$ appears exactly once.

Remark 8.2. *In the partially cyclic approach shown in the previous section, the rows of the matrix B are given as $B[i] = \mathcal{R}^{i-1}(\mathbf{b})$ $(i = 1, \ldots, o)$, where $\mathcal{R}^j$ is the cyclic right shift by j positions and $\mathbf{b}$ is a randomly chosen vector. The sequence obtained by this construction clearly does not fulfill Golomb's postulates. For example, for most of the $\lambda \in \mathbb{F}_q$ the 2-run (λ, λ) does not appear in such a sequence (contradiction to postulate R-2).*

8.2.2 Linear Recurring Sequences

In this subsection we discuss some basic facts about linear recurring sequences (LRS's). For more information about LRS's we refer to [37].

Definition 8.4. *Let L be a positive integer and $\gamma_1, \ldots, \gamma_L$ be given elements of a finite field $\mathbb{F}_q$. A linear recurring sequence (LRS) of length L is a sequence $S = (s_1, s_2, \ldots)$ of $\mathbb{F}_q$-elements satisfying the relation*

$$s_j = \gamma_1 \cdot s_{j-1} + \gamma_2 \cdot s_{j-2} + \cdots + \gamma_L \cdot s_{j-L} = \sum_{i=1}^{L} \gamma_i \cdot s_{j-i} \quad (j > L). \tag{8.3}$$

The values $s_1, \ldots, s_L$ are called the initial values of the LRS.

Definition 8.5. *The connection polynomial of the linear recurring sequence S is defined as*

$$C(X) = \gamma_L X^L + \gamma_{L-1} X^{L-1} + \ldots + \gamma_1 \cdot X + 1 = \sum_{i=1}^{L} \gamma_i X^i + 1.$$

The degree of the connection polynomial C corresponds to the length of the LRS.

The LRS S is uniquely determined by its initial values $s_1, \ldots, s_L$ and the connection polynomial $C(X)$ (due to equation (8.3)). Therefore we denote the sequence S by $S = \mathrm{LRS}(s_1, \ldots, s_L, C(X))$.

Definition 8.6. *A univariate polynomial $p(X) \in \mathbb{F}_q[X]$ is called primitive, if $p(X)$ is the minimal polynomial of a generator α of the extension field $\mathbb{F}_{q^n}$, i.e. $p(X)$ is a polynomial with $\deg(p(X)) = n$ in $\mathbb{F}_q[X]$, $p(\alpha) = 0$ and $\langle\alpha\rangle = \mathbb{F}_{q^n}^{\star}$.*

Definition 8.7. *A linear recurring sequence with non trivial initial state $(s_1, \ldots s_L) \in \mathbb{F}_q^L \setminus \{\mathbf{0}\}$ and primitive connection polynomial C of degree L is called an (L-th order) m-sequence.*

Lemma 8.1. *Any L-th order m-sequence over $\mathbb{F}_q$ is periodic with minimal period $q^L - 1$ and the minimal period of every LRS of length L is upper bounded by this number.*

Proof. See [37], Theorem 6.33, page 205. □

Definition 8.8. *The linear complexity $\mathrm{LC}(S)$ of a sequence S is the degree of the connection polynomial $C(X)$ of lowest degree L for which there exist $s_1, \ldots s_L \in \mathbb{F}_q$ such that $S = \mathrm{LRS}(s_1, \ldots, s_L, C(X))$.*

Lemma 8.2. *Let $S = \mathrm{LRS}(s_1, \ldots, s_L, C(X))$ be an L-th order m-sequence. Then we have $\mathrm{LC}(S) = L$.*

Proof. Follows directly from Definition 8.8 and Lemma 8.1. □

Lemma 8.2 states that there exists no linear recurring sequence of length $L' < L$ which generates S.

Lemma 8.3. *Any L-th order m-sequence fulfills the postulates R-1 to R-4 of Subsection 8.2.1 (for $n = L$).*

Proof. See [29], Property 5.2 - 5.5 (pages 128 ff.). □

Remark 8.3. *Because of the good statistical properties of m-sequences, linear recurring sequences are used to bring randomness into a large number of areas, for example digital broadcasting and the Global Positioning System (GPS). However, an m-sequence can not be said to be a truly random sequence. For example, the linear complexity of an L-th order m-sequence of length $N \gg L$ is L, whereas for a random sequence it should be about $N/2$. Therefore, the elements of an m-sequence are easily predictable. Hence, for cryptographic applications like stream ciphers, one has to add some non-linearity features.*

8.2.3 Construction of the Matrix B

In this subsection we describe how to generate the matrix B of Algorithm 7.1 for UOVLRS. To get a public key with good statistical properties, we use an L-th order m-sequence to compute the elements of the matrix B. We therefore start with a primitive polynomial $C(X)$ of degree L and a non trivial initial state $(s_1, \ldots, s_L) \in \mathbb{F}^L \setminus \{\mathbf{0}\}$. We then calculate the first $o \cdot D$ elements $s_1, \ldots, s_{o \cdot D}$ of the m-sequence $S = \mathrm{LRS}(s_1, \ldots, s_L, C(X))$ and define the i-th row of the matrix B by

$$B[i] = (s_{(i-1)\cdot D+1}, \ldots, s_{i \cdot D}) \quad (i = 1, \ldots, o). \tag{8.4}$$

Algorithm 8.2 shows this generation process in a compact form.

To generate a key pair of UOVLRS, we first use Algorithm 8.2 to generate a structured matrix B. Then we use Algorithm 7.1 (with selection strategy 1; see Subsection 7.1.1) to generate the private and public key of the scheme.

To represent the matrix B of UOVLRS we have to store the initial state $(s_1, \ldots, s_L)$ and the connection polynomial $C(X)$.
For the following we assume the connection polynomial $C(X)$ to be primitive. So, whenever we speak of a linear recurring sequence, we refer to an m-sequence.

Algorithm 8.2 Generation of a matrix B for UOVLRS

Input: parameters $(\mathbb{F}, o, v)$, L
Output: matrix $B \in \mathbb{F}^{o \times D}$ for UOVLRS

1: Choose randomly a vector $\mathbf{s} \in \mathbb{F}^L \setminus \{\mathbf{0}\}$ as initial state and a primitive connection polynomial $C(X)$ of degree L.
2: Compute the first $o \cdot D$ elements $s_1, \ldots, s_{o \cdot D}$ of the sequence $S = \mathrm{LRS}(\mathbf{s}, C(X))$.
3: **for** $i = 1$ to o **do**
4: $\quad B[i] \leftarrow (s_{(i-1) \cdot D+1}, \ldots, s_{i \cdot D})$.
5: **end for**
6: **return** B

Remark 8.4. *The randomness properties of m-sequences guarantee that we get a public key with good statistical properties. In particular, all elements of $\mathbb{F}$ appear in the matrix B approximately the same number of times. This is in contrast to the cyclicUOV scheme (see Section 8.1) where the elements of the vector $\mathbf{b}$ are repeated in every row of B. We therefore believe that the development of dedicated attacks against the UOVLRS scheme is a hard task.*

8.2.4 Choice of L

A natural question in this context is how to choose the length L of the linear recurring sequence in use. A small number L will reduce the size of the public key but possibly weaken the security of the scheme. The following theorem gives a lower bound on L. In the theorem we use the same notation as in Section 7.1 (see page 94).

Theorem 8.1. *Let $((\mathcal{F}, \mathcal{T}), \mathcal{P})$ be a UOV key pair generated by a linear recurring sequence of length L. Then we have* $\mathrm{rank}(P_{\mathrm{quad}}) = \mathrm{rank}(B) = \min(o, L)$.

Proof. Since B is a submatrix of P_{quad}, the rank of P_{quad} can not be less than that of B. But, according to equation (7.6), the rank of P_{quad} can not be larger than $\mathrm{rank}(Q) \overset{(7.8)}{=} \mathrm{rank}(B)$, too. The remainder of the proof is given by Lemma 8.4. □

Lemma 8.4. *Let the $o \times D$ matrix B be generated by an LRS of length L (as described by Algorithm 8.2). Then we have* $\mathrm{rank}(B) = \min(o, L)$.

Proof. As B is an $o \times D$ matrix, the rank of B can not be higher than o. So let's assume $L < o$. We show that $\mathrm{rank}(B) = L$. First assume that we have $\mathrm{rank}(B) > L$. Then there exist $r > L$ linear independent columns of B. W.l.o.g. we can assume that the first L columns of B, namely $B_1, \ldots, B_L$ are linearly independent. Let B_i $(i > L)$ be a column of B which is linearly independent of the first L columns. But, since the sequence S (see Algorithm 8.2) has linear complexity L, there exists a linear combination of the form $B_i = \sum_{j=1}^{L} \beta_j \cdot B_j$. which contradicts our assumption.
On the other hand let's assume that we have $\mathrm{rank}(B) = r < L$. Then there exist r columns $\hat{B}_1, \ldots, \hat{B}_r$ in B such that every other column B_k of B can be written in the form $B_k = \sum_{i=1}^{r} \hat{\beta}_i \cdot \hat{B}_i$. This implies that S can be generated by a linear recurring sequence of length $r < L$. But, this is a contradiction to S being an L-th order m-sequence and the primitiveness of the connection polynomial (see Lemma 8.2). □

Theorem 8.1 states that for $L < o$ the homogeneous quadratic parts of the public polynomials will be linearly dependent. In particular, of the o quadratic polynomials of a public key generated by an LRS of length $L < o$, only L will have linear independent homogeneous quadratic parts.
Therefore, an attacker can find linear relations between the public polynomials which transform the last $o - L$ components of the system $\mathcal{P}(\mathbf{x}) = \mathbf{h}$ into linear equations. By doing so, he can remove $o - L$ equations and variables from the system which leaves him with a system of only L quadratic equations. As a consequence of this, to achieve the maximal possible security, we have to choose the length of the LRS at least o.

(GF(256),o,v)	$(9, 18)$	$(10, 20)$	$(11, 22)$	$(12, 24)$	$(13, 26)$	$(14, 28)$
$L = o$	5.3 s	40.6 s	293.8 s	2,388 s	19,052 s	169,312 s
$L = o - 1$	-	5.8 s	42.2 s	303 s	2,311 s	20,952 s
$L = o - 2$	-	-	6.0 s	43.6 s	309 s	2,560 s

Table 8.1: Running time of the direct attack against UOVLRS for different values of L

To check the correctness of these theoretical considerations, we carried out a number of experiments with MAGMA [8]. For different parameter sets $(\mathrm{GF}(256), o, v, L)$ we created instances of the UOVLRS scheme and solved the corresponding systems, after fixing v variables to get a determined system, with MAGMA v.2.13-10 using the command `Variety`. Table 8.1 shows the results.

As the table shows, solving a UOV system with o equations generated by a linear recurring sequence of length $L < o$ is only as difficult as solving a determined system of L quadratic equations. We therefore fix the length of the LRS in our scheme to o. So we need $2 \cdot o$ field elements to represent the matrix B.
The size of the public key of UOVLRS is therefore given by

$$\text{size}_{\text{ pk UOVLRS}} = 2 \cdot o + o \cdot \left(\frac{o \cdot (o+1)}{2} + n + 1 \right) \tag{8.5}$$

field elements, the size of the private key is the same as for the standard UOV scheme (see equation (3.3)).

8.3 Combination of LRS and Cyclic Techniques

It is also possible to combine linear recurring sequences with the technique shown in Section 8.1. For cycUOVLRS, we generate the vector $\mathbf{b}$ used in Algorithm 8.1 by a linear recurring sequence of length L. To get the optimal security, we use for this step a primitive polynomial $C(X)$ of degree L and a non trivial initial state $(s_1, \ldots s_L) \in \mathbb{F}^L \setminus \{\mathbf{0}\}$. We then compute the first D elements $s_1, \ldots, s_D$ of the m-sequence $S = \mathrm{LRS}(s_1, \ldots, s_L, C(x))$, put them into a vector $\mathbf{b}$ and define the i-th row of the matrix B by

$$B[i] = \mathcal{R}^{i-1}(\mathbf{b}) \ (i = 1, \ldots, o). \tag{8.6}$$

As in Section 8.1, $\mathcal{R}^i(\mathbf{b})$ denotes the cyclic right shift of the vector $\mathbf{b}$ by i positions. Algorithm 8.3 shows this generation process in a compact form.

Algorithm 8.3 Generation of a matrix B for cycUOVLRS

Input: parameters $(\mathbb{F}, o, v)$, L
Output: matrix $B \in \mathbb{F}^{o \times D}$ for cycUOVLRS

1: Choose randomly a vector $\mathbf{s} \in \mathbb{F}^L \setminus \{\mathbf{0}\}$ and a primitive connection polynomial $C(X)$ of degree L.
2: Compute the first D elements of the m-sequence $S = \mathrm{LRS}(\mathbf{s}, C(x))$ and put them into a vector $\mathbf{b} \in \mathbb{F}^D$.
3: $B[1] \leftarrow \mathbf{b}$
4: **for** $i = 2$ to o **do**
5: $\quad \mathbf{b} \leftarrow \mathcal{R}(\mathbf{b})$
6: $\quad B[i] \leftarrow \mathbf{b}$
7: **end for**
8: **return** B

(GF(256),o,v)	(9, 18)	(10, 20)	(11, 22)	(12, 24)	(13, 26)	(14, 28)
$L = o$	5.4 s	340.5 s	292.9 s	2,384 s	19,065 s	169,412 s
$L = o - 1$	5.1 s	338.3 s	286.4 s	2,371 s	19,012 s	168,283 s
$L = o - 2$	2.9 s	18.1 s	136.3 s	1,035 s	8,833 s	82,473 s

Table 8.2: Running time of the direct attack against cycUOVLRS for different values of L

We can use this matrix B as input for Algorithm 7.1 (with selection strategy 1; see Subsection 7.1.1) to generate a key pair for cycUOVLRS.

As in the case of UOVLRS, we need $2 \cdot L$ field elements to represent the matrix B.

8.3.1 Choice of L

As for UOVLRS (see Section 8.2) we have to answer the question how to choose the length L of the linear recurring sequence in use. In the case of cycUOVLRS, this question can not be answered as directly as in the previous section. The reason for this is the cyclic right shift $\mathcal{R}$, which destroys some of the structure of the LRS. To get an impression of the influence of L on the security of our scheme, we created for different parameter sets instances of cycUOVLRS and solved the public systems, after fixing v variables to get a determined system, with MAGMA v.2.13-10 using the command `Variety`. Table 8.2 shows the results. As the table shows, cycUOVLRS systems with $L < o$ can be solved significantly faster than systems with $L = o$ (at least for $L \leq o - 2$). The reason for this is that the cyclic right shift $\mathcal{R}$ destroys the linear structure only in the left part of the matrix B. So there remain linear dependencies between the coefficients of the public polynomials which can be used during the elimination step of a Gröbner Basis method. By using these dependencies in a systematic way, it should be possible to reduce the numbers in Table 8.2 even further. We therefore recommend to use for cycUOVLRS a linear recurring sequence of length o.
So, in order to represent the matrix B, we need $2 \cdot o = v$ field elements. The size of the public key is the same as for the UOVLRS scheme, i.e.

$$\text{size}_{\text{pk cycUOVLRS}} = 2 \cdot o + o \cdot \left(\frac{o \cdot (o+1)}{2} + n + 1 \right) \tag{8.7}$$

field elements, the size of the private key is the same as for the standard UOV scheme (see equation (3.3)).

8.4 UOVLRS2

To speed up the verification process (see Section 10.3) it is useful to use linear recurring sequences of short length. As we have seen in the previous sections, neither UOVLRS nor cycUOVLRS are a candidate for this. Therefore we come to another approach. In contrast to the schemes presented in the previous sections, we use not only one, but o different linear recurring sequences. The goal of this strategy is to reduce the lengths of the single LRS's, which will later help us to speed up the verification process of the scheme (see Chapter 10). In fact, we use linear recurring sequences of length 1.
We choose two vectors α and $\gamma \in \mathbb{F}^o$ and define for each $i = 1, \ldots, o$ a univariate polynomial $C_i(X)$ by $C_i(X) = \gamma_i \cdot X + 1$. For $i = 1, \ldots, o$ we compute the first D elements $s_1^{(i)}, \ldots, s_D^{(i)}$ of the linear recurring sequence

$$S^{(i)} = \text{LRS}(\alpha_i, C_i(X)) \tag{8.8}$$

and put this sequence into the i-th row of the matrix B. Therefore, the matrix B will have the following structure:

$$B = \begin{pmatrix} s_1^{(1)} & s_2^{(1)} & \dots & s_D^{(1)} \\ s_1^{(2)} & s_2^{(2)} & \dots & s_D^{(2)} \\ \vdots & & & \vdots \\ s_1^{(o)} & s_2^{(o)} & \dots & s_D^{(o)} \end{pmatrix}. \tag{8.9}$$

Algorithm 8.4 presents the generation of a such a matrix B in a compact form.

Algorithm 8.4 Generation of a matrix B for UOVLRS2

Input: parameters $(\mathbb{F}, o, v)$
Output: matrix $B \in \mathbb{F}^{o \times D}$ for UOVLRS2
1: Choose two vectors α and $\gamma \in \mathbb{F}^o$ (see Subsection 8.4.1).
2: **for** $i = 1$ to o **do**
3: Compute the first D elements $s_1^{(i)}, \dots, s_D^{(i)}$ of the linear recurring sequence $S^{(i)} = \mathrm{LRS}(a_i, \gamma_i \cdot X + 1)$ and put them into a vector $\mathbf{b} \in \mathbb{F}^D$.
4: $B[i] \leftarrow \mathbf{b}$
5: **end for**
6: **return** B

To create a key pair of UOVLRS2, we use the matrix B computed by Algorithm 8.4 as input for Algorithm 7.1 (with selection strategy 1; see Subsection 7.1.1).

8.4.1 Choice of α and γ

In this subsection we consider the question how the two vectors α and γ in line 1 of Algorithm 8.4 should be chosen.

First, we look at the question what happens if two elements of the vector γ, say γ_i and γ_j $(1 \leq i < j \leq o)$ are equal.

Theorem 8.2. *If $\gamma_i = \gamma_j$ for $i \neq j \in \{1, \dots, o\}$, the homogeneous quadratic parts of the public polynomials $p^{(i)}$ and $p^{(j)}$ are linearly dependent.*

Proof. If $\gamma_i = \gamma_j$ for $i \neq j \in \{1, \dots, o\}$, the two rows $B[i]$ and $B[j]$ are linearly dependent. Since we have $Q = B \cdot A^{-1}$ (c.f. equation (7.8)), the same holds for the rows $Q[i]$ and $Q[j]$ (see Figure 7.1). Note that the matrix Q contains all the coefficients of quadratic terms of the map $\mathcal{F}$, which means that the homogeneous quadratic parts of the i-th and j-th central polynomials are linearly dependent. Since during the key generation of UOV the rows of the central map $\mathcal{F}$ are not mixed, the same holds for the homogeneous quadratic part of the i-th and j-th public polynomial. □

Theorem 8.2 states that by computing $p^{(i)} - \frac{\alpha_i}{\alpha_j} \cdot p^{(j)}$ the attacker gets a linear equation in the system variables, which means that he can reduce the number of variables in the quadratic system $\mathcal{P}(\mathbf{x}) = \mathbf{h}$ by 1. We can conclude

Corollary 8.1. *Attacking an instance of UOVLRS2 with o equations and $t < o$ different values in the vector γ is only as hard as solving a (UOVLRS2) system of t equations.*

To check this theoretical result, we created instances of UOVLRS2 for different parameters o and v and different types of vectors γ and solved the resulting public systems with MAGMA v.2.13-10 (see Table 8.3). Before applying the MAGMA command `Variety` we fixed v of the n variables to create determined systems.

t [1]/ (GF(256),o,v)	(9,18)	(10,20)	(11,22)	(12,24)	(13,26)	(14,28)
9	5.4 s	5.7 s	5.7 s	5.8 s	5.8 s	5.9 s
10	———	40.8 s	41.2 s	41.8 s	43.0 s	44.6 s
11	———	———	288.3 s	301.4 s	309.8 s	315.2 s

[1] number of different values in γ

Table 8.3: Running time of the direct attack against UOVLRS2 for different types of γ

To achieve the optimal security level, the elements of the vector γ have to be chosen pairwise distinct. Furthermore, all the elements have to be $\neq 0$.

Remark 8.5. *The above condition gives a lower bound on the cardinality of the underlying field. In particular, we can not define our scheme over $GF(16)$ and $GF(31)$.*

On the contrary, there seem to be no major conditions for the choice of the vector α.[2] We have to ensure only that $\alpha_i \in \mathbb{F} \setminus \{0\} \;\; \forall i = 1, \ldots, o$. For simplicity we choose $\alpha = \underbrace{(1, \ldots, 1)}_{o}$. Therefore, we get a matrix B of the Vandermonde-type:

$$B = \begin{pmatrix} 1 & \gamma_1 & \gamma_1^2 & \ldots & \gamma_1^{D-1} \\ 1 & \gamma_2 & \gamma_2^2 & \ldots & \gamma_2^{D-1} \\ \vdots & & & & \vdots \\ 1 & \gamma_o & \gamma_o^2 & \ldots & \gamma_o^{D-1} \end{pmatrix}. \tag{8.10}$$

To represent the matrix B we therefore need to store only the vector $\gamma \in \mathbb{F}^o$. So, the public key size of UOVLRS2 is given by

$$\text{size}_{\text{pk UOVLRS2}} = o + o \cdot \left(\frac{o \cdot (o+1)}{2} + n + 1 \right) \tag{8.11}$$

field elements, the size of the private key is the same as for the standard UOV scheme (see equation (3.3)).

In Section 10.3 we show how the structure in the public key of UOVLRS2 can be used to speed up the verification process of the scheme.

8.5 UOVrand

In this section we present an improved version of UOV called UOVrand for which it is possible to achieve some results in the direction of provable security. For the following we set $\alpha = \frac{v}{o}$.
For UOVrand, the elements of the $o \times D$ matrix B are chosen uniformly at random from $\mathbb{F}$. We can then use Algorithm 7.1 (with selection strategy 1; see Subsection 7.1.1) to generate a UOV key pair $((\mathcal{F}, \mathcal{T}), \mathcal{P})$. After performing this algorithm, we use a secret permutation $\Sigma \in S_n$ on the set of variables to compute $\mathcal{T}' = \mathcal{T} \circ \Sigma$ and $\mathcal{P}' = \mathcal{P} \circ \Sigma$. Since

$$\mathcal{F} \circ \mathcal{T}' = \mathcal{F} \circ \mathcal{T} \circ \Sigma = \mathcal{P} \circ \Sigma = \mathcal{P}', \tag{8.12}$$

$((\mathcal{F}, \mathcal{T}'), \mathcal{P}')$ is a valid UOV key pair. We call $(\mathcal{F}, \mathcal{T}'), \mathcal{P}')$ a key pair of UOVrand. Let us denote the input variables of $\mathcal{P}$ by $x_1, \ldots, x_n$ and the input variables of $\mathcal{P}'$ by $u_1, \ldots, u_n$. Note that we have $x_i = u_{\Sigma^{-1}(i)} \;\forall i = 1, \ldots, n$.
We denote the set of all variables u_i corresponding to Vinegar variables x_i $(i \in V)$ by X, the set $\{u_1, \ldots, u_n\} \setminus X$ by Y. We have $|X| = \frac{\alpha}{\alpha+1} \cdot n$ and $|Y| = \frac{n}{\alpha+1}$.

[2] In fact, the attacker is allowed to multiply each public polynomial $p^{(i)}$ $(i = 1, \ldots, o)$ by an arbitrary element $a_i \in \mathbb{F} \setminus \{0\}$. By doing so, he can produce a vector α' of his choice.

8.5.1 Security Reduction

Due to the above construction the major part of the coefficients in the public polynomials (both of $\mathcal{P}$ and $\mathcal{P}'$) is chosen uniformly at random. In $\mathcal{P}$, these are the coefficients of the monomials $x_i x_j$ which contain at least one Vinegar variable, i.e.

$$p_{ij}^{(k)} \text{ is chosen uniformly at random from } \mathbb{F} \Leftrightarrow i \in V \vee j \in V. \tag{8.13}$$

For $\mathcal{P}'$ we get

$$p_{ij}^{\prime(k)} \text{ is chosen uniformly at random from } \mathbb{F} \Leftrightarrow u_i \in X \vee u_j \in X. \tag{8.14}$$

But, without the knowledge of the permutation Σ, an attacker is not able to detect which variables u_i are contained in the set X.
When attacking the system $\mathcal{P}'$ directly, an attacker usually fixes a number of variables u_i to create an (over-)determined system. When the attacker randomly chooses a variable u_i to be fixed, he meets with probability $\frac{\alpha}{\alpha+1}$ a variable $u_i \in X$ and with probability $\frac{1}{\alpha+1}$ a variable from Y. When fixing v of the variables u_i, the attacker fixes therefore on average $\frac{\alpha}{\alpha+1} \cdot v$ variables from X and $\frac{v}{\alpha+1}$ variables from Y. We can use this observation to prove the following theorem:

Theorem 8.3. *A direct attack against a UOVrand scheme with o equations and $v+o = (\alpha+1)\cdot o$ variables is on average at least as hard as solving a quadratic random system of o equations in $\frac{\alpha}{\alpha+1} \cdot o$ variables.*

Proof. Let $\mathcal{A}$ be an attacker who wants to solve the system $\mathcal{P}'(u_1, \ldots, u_n) = \mathbf{h}$ directly. Let us assume that, before applying XL or a Gröbner Basis algorithm, $\mathcal{A}$ fixes v of the variables u_i to create a determined system. Since he has no means to detect which of the variables u_i are contained in the set X (and therefore correspond to Vinegar variables x_j), he chooses the indices $i_1, \ldots, i_v$ of variables to be fixed uniformly at random from the set $\{1, \ldots, n\}$. As we have seen above, he meets by this strategy on average $\frac{\alpha}{\alpha+1} \cdot v$ variables from X and $\frac{v}{\alpha+1}$ variables from Y. Let us denote the set of fixed variables from X and Y by X_F and Y_F respectively. After this fixing part, he therefore gets a system $\tilde{\mathcal{P}}$ of o equations in the o variables contained in $X \setminus X_F$ and $Y \setminus Y_F$.
Now let's assume that $\mathcal{A}$ has access to an oracle which gives him the values of all variables from $Y \setminus Y_F$. He therefore ends up with a system $\tilde{\tilde{\mathcal{P}}}$ of o equations in the variables $u_i \in X \setminus X_F$. As we have seen, this set contains on average $v - \frac{\alpha}{\alpha+1} \cdot v = \frac{\alpha}{\alpha+1} \cdot o$ variables and according to (8.14) the homogeneous quadratic part of the system $\tilde{\tilde{\mathcal{P}}}$ is chosen uniformly at random from $\mathbb{F}$. □

The security reduction as given by Theorem 8.3 is quite bad. For example, to reach a (provable) security of 80 bit (under direct attacks) for a UOVrand scheme with o equations and $3 \cdot o$ variables, we would need $o = 66$ equations over GF(256). The reason for this is the oracle used in the proof of Theorem 8.3, which gives the attacker the values of (on average) $\frac{o}{3}$ variables $u_i \in Y \setminus Y_F$. By doing so, the oracle reduces the complexity of the attack by a huge factor. In practice we expect that solving the system $\tilde{P}$ is as hard as solving an $o \times o$ system with randomly chosen coefficients.

In a practical setting, one uses a cryptographic secure pseudo random number generator to generate the elements of the matrix B of UOVrand. For example, we can use a block cipher such as AES in the OFB mode or a stream cipher like Salsa20 [3] for this step. In this case, in order to represent the matrix B, we have to store only a small random seed of e.g. 128 bit. Therefore, the public key size of the scheme is given by

$$\text{size}_{\text{pk UOVrand AES}} = 128 \text{ bit} + o \cdot \left(\frac{o \cdot (o+1)}{2} + n + 1 \right) \tag{8.15}$$

field elements, the size of the private key is the same as for the standard UOV scheme (see equation (3.3)).

8.6 0/1UOV

Let $\mathbb{F}$ be a finite field of cardinality $q > 2$. In this section we describe an approach to generate a UOV key pair defined over $\mathbb{F}$, whose public coefficients are mainly in the field GF(2). Hereby we can not follow the strategy used in the previous sections (see Subsection 8.6.2). Before we can present our new approach, we first need a result from graph theory.

8.6.1 The Turán graph

In this subsection we introduce the Turán graph which is the basis of our construction presented in the next subsection. Before we come to the Turán graph itself, we first need some basic definitions.

Definition 8.9. *An (undirected) graph is an ordered pair $G = (V, E)$ consisting of a set V of vertices and a set E of edges, which are 2-element subsets of V. We denote an edge connecting the two vertices x and $y \in V$ by (x, y). In this case, the two vertices x and y are called adjacent.*

Definition 8.10. *The complement of a graph $G = (V, E)$ is a graph $\overline{G}$ (called complementary graph) on the same set of vertices such that two vertices of $\overline{G}$ are adjacent if and only if they are not adjacent in G, i.e. $\overline{G} = (V, K \setminus E)$ where K is the set of all 2-element subsets of V.*

Definition 8.11. *A k-independent set in an undirected graph $G = (V, E)$ is a set $I \subset V$ of k vertices, no two of which are connected by an edge, i.e. $(x, y) \notin E \ \forall \ x, y \in I$.*

Definition 8.12. *A k-clique in an undirected graph $G = (V, E)$ is a k-subset of the vertex set $C \subset V$, such that for every two vertices in C, there exists an edge connecting the two, i.e. $(x, y) \in E \ \forall \ x, y \in C$.*

The Turán graph $\mathrm{T}(n, k)$ (named after the Hungarian mathematician Pal Turán) is defined as follows [57]: The set V of n vertices is partitioned into k subsets $A_1, \ldots, A_k$, whose sizes are as equal as possible, i.e. $\bigcup_{i=1}^{k} A_i = V$, $A_i \cap A_j = \emptyset$, $||A_i| - |A_j|| \leq 1 \ \forall \ i \neq j$. Two vertices are connected by an edge if and only if they belong to different subsets, i.e. $(v_i, v_j) \in E \Leftrightarrow v_i \in A_r, \ v_j \in A_s$ with $r \neq s$.

The number of edges in $\mathrm{T}(n, k)$ is given by $\frac{1}{2} \cdot \sum_{i=1}^{k} |A_i| \cdot (n - |A_i|)$ and is upper bounded by $\left(1 - \frac{1}{k}\right) \cdot \frac{n^2}{2}$. Since every set of $(k + 1)$ vertices contains at least two vertices in the same subset A_i, the Turán graph does not contain a $(k + 1)$ clique.

Theorem 8.4. *[57] The Turán graph is the graph with the highest possible number of edges with this property.*

The complementary graph of the Turán graph $\mathrm{T}(n, k)$ is denoted by $\mathrm{CT}(n, k)$. Here, two vertices are connected by an edge if and only if they belong to the same subset,
i.e. $(v_i, v_j) \in E \Leftrightarrow \exists r \in \{1, \ldots, k\}$ s.t. $v_i \in A_r \wedge v_j \in A_r$.
The number of edges in the graph $CT(n, k)$ is given by $|E| = \sum_{i=1}^{k} \binom{|A_i|}{2}$, which is bounded from below by $\frac{n}{2} \cdot \left(\frac{n}{k} - 1\right)$.
Since every set of $(k+1)$ vertices contains at least two vertices from the same subset A_i, the graph $\mathrm{CT}(n, k)$ does not contain a $(k + 1)$ independent set. From Theorem 8.4 it follows that $\mathrm{CT}(n, k)$ is the graph with the minimal number of edges with this property.
Figure 8.1 shows for $n = 8$ and $k = 3$ the Turán graph T(8,3) and its complementary graph CT(8,3).

8.6.2 The Selection Strategy

When using a matrix $B \in \mathrm{GF}(2)^{o \times D}$, the coefficients of the quadratic monomials $x_i x_j$ corresponding to the columns of B are elements of GF(2). Only the $\frac{o \cdot (o+1)}{2}$ remaining quadratic monomials will have coefficients in $\mathbb{F} \setminus \mathrm{GF}(2)$. We denote the set of these monomials by $\overline{C}$. When using a

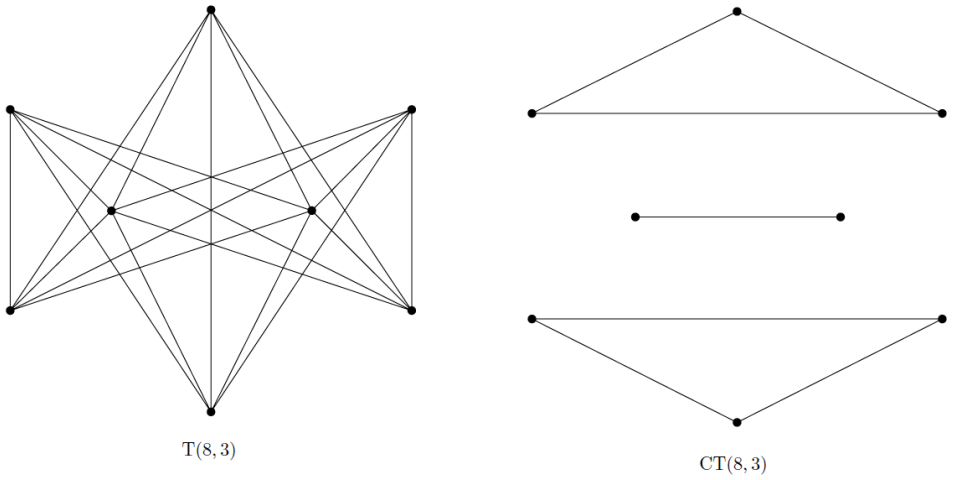


Figure 8.1: Turán graph T(8, 3) and complementary graph CT(8, 3)

matrix $B \in \mathrm{GF}(2)^{o \times D}$ together with selection strategy 1 (see Subsection 7.1.1), this set $\overline{C}$ has the form

$$\overline{C} = \{x_i x_j | v+1 \leq i \leq j \leq n\}. \tag{8.16}$$

Before applying an algorithm like XL or a Gröbner Basis method, the attacker usually fixes some of the variables to create an (over-)determined system. By fixing the o variables x_i $(v+1 \leq i \leq n)$ an attacker can turn all the monomials of $\overline{C}$ into constants and therefore create a system whose quadratic terms have coefficients from GF(2). As Table 8.4 shows, MAGMA can solve systems of this type significantly faster than the public systems of the standard UOV scheme. Therefore we have to follow a new approach.

(GF(256),o,v)	(9,18)	(10,20)	(11,22)	(12,24)	(13,26)	(14,28)
UOV system	5.5 s	38.3 s	289.2 s	2,383 s	18,928 s	169,038 s
GF(2) - system	3.7 s	32.4 s	253.4 s	2,214 s	17,742 s	143,749 s

[1] system obtained by using a matrix B over GF(2) as input for Algorithm 7.1 (with selection strategy 1)

Table 8.4: Running time of the direct attack against UOV and GF(2) systems

The goal of our new approach is to choose the elements of $\overline{C}$ in such a way that, no matter which variables the attacker fixes before applying an algorithm like F_4, he is not able to turn all the monomials of $\overline{C}$ to linear or constant terms. Our new idea can be described as follows:

The set $\overline{C}$ of quadratic monomials with coefficients in $\mathbb{F} \setminus \mathrm{GF}(2)$ can be seen as a graph $G(V, E)$ with vertices $V = \{x_1, \ldots, x_n\}$ and edges $E = \{(x_i, x_j) | x_i x_j \in \overline{C}\}$. By construction we have $|E| = \frac{o \cdot (o+1)}{2}$. By fixing/guessing k variables, the attacker creates a new graph $G^{(k)} = (V^{(k)}, E^{(k)})$ with $n-k$ vertices and $E^{(k)} = \{(x_i, x_j) \in E | x_i \in V^{(k)} \wedge x_j \in V^{(k)}\}$.
In the following we create the graph G in such a way, that the graph $G^{(k)}$, no matter which variables the attacker fixes before applying a Gröbner basis algorithm, contains at least one edge (for all $k \leq \bar{k}$ and a maximal upper bound $\bar{k}$). This means that the attacker is forced to solve the system over $\mathbb{F}$ and can not restrict to the easier case of computing a Gröbner Basis over GF(2). We observe the following:

Theorem 8.5. *By fixing/guessing k variables, an attacker is able to create a system whose quadratic terms all have coefficients in $GF(2)$ if and only if the graph G contains an $(n-k)$-independent set (see Definition 8.11).*

Proof. $'\Leftarrow'$: Assume that G contains an $(n-k)$-independent set $I \subset V$. Set $V' = V \setminus I$. Note that $|V'| = k$. Then E contains $\frac{o\cdot(o+1)}{2}$ edges of the form $(x_i, x_j) : i \in V' \vee j \in V'$. By fixing k of the variables the attacker is able to remove all the vertices $x_i \in V'$ from the graph G. Therefore, we have $E^{(k)} = \emptyset = \overline{C}$, which means that there remains a system whose quadratic terms all have coefficients over GF(2).
$'\Rightarrow'$: Assume that G does not contain an $(n-k)$-independent set. Therefore, for each $(n-k)$-subset of vertices V' there exists at least one edge of the form (x_i, x_j) with $x_i \in V'$ and $x_j \in V'$. When the attacker removes k vertices from V there remain $n-k$ vertices $\tilde{x}_1, \ldots, \tilde{x}_{n-k}$ in the graph $G^{(k)}$. Since $|V^{(k)}| = n-k$, there remains at least one edge $(\tilde{x}_i, \tilde{x}_j)$ with $\tilde{x}_i \in V^{(k)}$ and $\tilde{x}_j \in V^{(k)}$. Therefore, each polynomial in $\mathcal{P}$ contains at least one quadratic monomial with coefficient in $\mathbb{F} \setminus$ GF(2). □

According to Theorem 8.5 and our discussion before, we have to create a graph $G = (V, E)$ with the following properties:

- G contains n vertices $x_1, \ldots, x_n$, i.e. $|V| = n$.
- G contains $\frac{o\cdot(o+1)}{2}$ edges, i.e. $|E| = \frac{o\cdot(o+1)}{2}$.
- G does not contain an ℓ-independent set (for all $\ell \geq \bar{\ell}$ and minimal $\bar{\ell}$).

For fixed ℓ, the problem of finding the graph with n vertices, without ℓ-independent set and minimal number of edges is solved by the complementary Turán graph $CT(n, \ell-1)$. So, to find the optimal graph G for our purposes, we apply the following strategy:

We start with $\ell = 1$ and construct the complementary Turán graph $CT(n, 1)$. We then increase ℓ until the number of edges in $CT(n, \ell)$ is less or equal to $\frac{o\cdot(o+1)}{2}$.
Let $\hat{\ell}$ be this minimal ℓ such that the number $|\hat{E}|$ of edges in $CT(n, \hat{\ell})$ is less or equal to $\frac{o\cdot(o+1)}{2}$. If we have $\frac{o\cdot(o+1)}{2} - |\hat{E}| = t > 0$, we add t edges to $\hat{E}$ (beginning with the edges $(x_i, x_j) \notin \hat{E}$ corresponding to the first quadratic cross terms in the graded lexicographic order). By doing so, we obtain a graph $G(V, E)$ with $CT(n, \hat{\ell}) \subset G$, $|V| = n$ and $|E| = \frac{o\cdot(o+1)}{2}$. Since G contains $CT(n, \hat{\ell})$ as a subgraph, it does not contain an $\hat{\ell}+1$ independent set.

We now define a selection strategy suitable for our purposes as follows. The set I containing the monomials appearing in the matrix B (see Section 7.1) is given as

$$I = \{(i, j) : (i, j) \in \overline{G} \vee i = j\}, \tag{8.17}$$

where $\overline{G}$ is the complementary graph of G which is actually a subgraph of the Turán graph $T(n, \hat{\ell})$. Note that $\overline{G}$ contains $\binom{n}{2} - \frac{o\cdot(o+1)}{2} = D - n$ edges. Therefore, I contains exactly D 2-tuples (i, j) $(1 \leq i \leq j \leq n)$ (including all the 2-tuples (i, i) $(1 \leq i \leq n)$).
We now choose the elements of the $o \times D$ matrix B uniformly at random from GF(2). We can then generate a key pair of 0/1 UOV by following Algorithm 7.1 (using the selection strategy defined above). By doing so we ensure that an attacker who fixes up to $n - \hat{\ell} - 1$ variables is not able to remove all quadratic terms with coefficients in $\mathbb{F} \setminus$ GF(2) from the system.

Example 8.1. *For the parameters* $(\mathbb{F}, o, v) = (\text{GF}(256), 28, 56)$ *we have* $n = 84$,
$D = \frac{v\cdot(v+1)}{2} + o \cdot v = 3164$ *and* $D_2 = \frac{o\cdot(o+1)}{2} = 406$.
To find the number $\hat{\ell}$, *we compute for* $\ell = 1, \ldots$ *the number of edges in the graph* $\text{CT}(84, \ell)$.

ℓ	1	2	3	4	5	6	7	8
# edges in $\mathrm{CT}(84,\ell)$	3486	1722	1134	840	664	546	462	400
	$> D_2$	$> D_2$	$> D_2$	$> D_2$	$> D_2$	$> D_2$	$> D_2$	$\leq D_2$

So we have $\hat{\ell} = 8$. To construct the complementary Turán graph $CT(84,8)$ we divide the set of vertices $x_1, \ldots, x_{84}$ into 8 subsets A_i $(i = 1, \ldots 8)$ as follows

i	1	2	3	4	5	6	7	8
$\lvert A_i \rvert$	11	11	11	11	10	10	10	10
A_i	$x_1, \ldots, x_{11}$	$x_{12}, \ldots, x_{22}$	$x_{23}, \ldots, x_{33}$	$x_{34}, \ldots, x_{44}$	$x_{45}, \ldots, x_{54}$	$x_{55}, \ldots, x_{64}$	$x_{65}, \ldots, x_{74}$	$x_{75}, \ldots, x_{84}$

The graph $CT(84,8)$ contains 400 edges (x_i, x_j) with $i \neq j$ and x_i and x_j belonging to the same subset A_i $(i = 1, \ldots, 8)$.
To get the full number of 406 edges in the graph G we have to add 6 edges to $CT(84,8)$. In our example these are the edges $(x_1, x_{12}), \ldots, (x_1, x_{17})$ (Note that the edges $(x_1, x_2), \ldots, (x_1, x_{11})$ are already contained in $CT(84,8)$). By doing so we get a graph G with $n = 84$ vertices and $\frac{o \cdot (o+1)}{2} = 406$ edges which contains the graph $CT(84,8)$ as a subgraph. The complementary graph $\overline{G}$ therefore contains $\binom{84}{2} - 406 = 3080$ edges.

Fixing of variables*:*
When attacking the scheme directly, an attacker usually fixes a certain number of variables before applying XL or a Gröbner basis method. The attacker tries to fix the variables in such a way, that the maximal number of edges in G (and therefore monomials in $\overline{C}$) vanishes. For this step we neglect the edges added in the previous step and assume that $G = CT(84,8)$. Since the number of edges in the complementary Turán graph CT(n,k) is given by

$$\sum_{i=1}^{k} \binom{|A_i|}{2},$$

it is obviously the best strategy for the attacker to remove from each subset A_i approximately the same number of vertices. Note that by removing r vertices from $\mathrm{CT}(n,k)$ using the above strategy the attacker creates the graph $\mathrm{CT}(n-r,k)$.

When fixing v of the n variables, the attacker creates the graph $\mathrm{CT}(o,k)$. In our example, we get the graph $\mathrm{CT}(28,8)$ whose 28 vertices are divided into subsets A'_i $(i = 1, \ldots, 8)$ as follows:

i	1	2	3	4	5	6	7	8
$\lvert A'_i \rvert$	4	4	4	4	3	3	3	3

The number of edges in this graph and therefore the number of quadratic terms with coefficients in $GF(256) \setminus GF(2)$ remaining in each component of the public system is $\sum_{i=1}^{8} \binom{|A'_i|}{2} = 36$.

When fixing/guessing $v + 2 = 30$ variables, the attacker implicitly creates the graph $\mathrm{CT}(26,8)$, whose vertices are divided into 8 groups A''_i as follows:

i	1	2	3	4	5	6	7	8
$\lvert A''_i \rvert$	4	4	3	3	3	3	3	3

The number of edges in this graph and therefore the number of quadratic terms with coefficients in $GF(256) \setminus GF(2)$ remaining in each component of the public system is $\sum_{i=1}^{8} \binom{|A''_i|}{2} = 30$.
To remove all edges in the graph G (i.e. all quadratic monomials with coefficients in $GF(256) \setminus GF(2)$ from the system $\mathcal{P}$) the attacker would have to fix/guess $n - \hat{\ell} = 84 - 8 = 76$ variables. To find a solution of the original system by this strategy, he would have to solve $256^{76-v} = 2^{8 \cdot 20} = 2^{160}$ final systems, which is far out of reach. □

(GF(256), o, v)	(9,18)	(10,20)	(11,22)	(12,24)	(13,26)	(14,28)
UOV	5.5 s	40.5 s	299.7 s	2,389 s	19,045 s	169,261 s
cyclicUOV	5.5 s	40.4 s	299.6 s	2,388 s	19,023 s	169,127 s
UOVLRS ($L = o$)	5.4 s	40.3 s	299.2 s	2,386 s	19,031 s	169,173 s
UOVLRScyc ($L = o$)	5.4 s	40.4 s	299.5 s	2,383 s	19,028 s	169,018 s
UOVLRS2	5.3 s	40.3 s	299.4 s	2,385 s	19,035 s	169,214 s
UOVrand	5.5 s	40.5 s	299.8 s	2,388 s	19,047 s	169,227 s
0/1 UOV	5.4 s	40.2 s	299.2 s	2,385 s	19,031 s	168,724 s

Table 8.5: Running time of the direct attack against UOV like schemes

The following table shows for some values of o the optimal number $\hat{\ell}$, as well as the number of monomials in $\overline{C}$ and the minimal number of monomials remaining in $\overline{C}$ after fixing/guessing v and $v+2$ variables (denoted by $|\overline{C}|_v$ and $(|\overline{C}|_{v+2})$ respectively. As it is usually done for UOV, we set here $v = 2 \cdot o$.

o	10	18	26	28	30	32	36	40	44	48
$\hat{\ell}$	7	8	8	8	8	8	9	9	9	9
$\lvert\overline{C}\rvert$	55	171	351	406	465	528	666	820	990	1080
$\lvert\overline{C}\rvert_v$	3	12	30	36	42	48	54	70	86	105
$\lvert\overline{C}\rvert_{v+2}$	1	8	24	30	36	42	48	62	78	95

The public key size of 0/1UOV is given by

$$\text{size}_{\text{pk 0/1UOV}} = o \cdot \left(\frac{v \cdot (v+1)}{2} + o \cdot v \right) \text{ bits } + o \cdot \left(\frac{o \cdot (o+1)}{2} + n + 1 \right) \tag{8.18}$$

field elements, the size of the private key is the same as for the standard UOV scheme (see equation (3.3)).

8.7 Security

In this section we analyze the security of our improved versions of the UOV signature scheme against known attacks. These attacks include (see Section 3.3)

- direct attacks,
- the UOV attack and
- the UOV-Reconciliation attack.

Direct attacks

The most straightforward way to attack a multivariate signature scheme like UOV is by solving the public system $\mathcal{P}(\mathbf{x}) = \mathbf{h}$ by XL or a Gröbner Basis method (signature forgery attack; see Section 2.5). To analyze the security of our improved schemes against these attacks, we carried out a number of experiments with MAGMA [8] v.2.13-10, which contains an efficient implementation of Faugère's F_4 algorithm [23]. For each of the improved versions of UOV presented in this chapter and each parameter set listed in Table 8.5 we created 100 of the public systems $\mathcal{P}(\mathbf{x}) = \mathbf{h}$ and solved them using the MAGMA command `Variety`. Table 8.5 shows the results.

(GF(256), o, v)	(9,18)	(10,20)	(11,22)	(12,24)	(13,26)	(14,28)
UOV	5.6 s	40.8 s	298.9 s	2,391 s	19,071 s	169,642 s
cyclicUOV	5.5 s	40.6 s	298.6 s	2,390 s	18,987 s	169,454 s
UOVLRS ($L = o$)	5.5 s	40.5 s	299.1 s	2,389 s	18,519 s	169,421 s
UOVLRScyc ($L = o$)	5.4 s	40.5 s	299.0 s	2,389 s	18,918 s	169,443 s
UOVLRS2	5.5 s	40.4 s	299.1 s	2,391 s	18,924 s	169,541 s
UOVrand	5.6 s	40.5 s	299.2 s	2,392 s	19,031 s	169,637 s
0/1 UOV	5.4 s	40.4 s	299.1 s	2,391 s	18,904 s	168,961 s

Table 8.6: Running time of the UOV-Reconciliation attack against UOV like schemes

(GF(256), o, v)	(2,4)	(3,6)	(4,8)	(5,10)
UOV	16.1	24.3	32.2	40.0
cyclicUOV	16.1	24.2	32.0	39.9
UOVLRS ($L = o$)	16.0	24.1	31.9	39.9
UOVLRScyc ($L = o$)	16.1	24.2	32.1	40.0
UOVLRS2	16.0	24.1	32.0	39.9
UOVrand	16.1	24.2	32.2	40.0
0/1 UOV	16.0	24.1	32.1	39.9

Table 8.7: Results of the experiments with the UOV attack

UOV-Reconciliation attack

The goal of the UOV-Reconciliation attack (see Subsection 3.3.2) is to find a linear transformation which brings the public polynomials into the form of a UOV central map. To do this, one has to solve several systems of multivariate quadratic equations. The complexity of the attack is mainly determined by the complexity of solving the first and largest of these systems. Table 8.6 shows the time, MAGMA needs to solve this system which consists of o equations in v variables. For each of our schemes and each of the parameter sets listed in the table we attacked 100 instances.

UOV attack

The goal of the UOV attack (see Subsection 3.3.3) is to compute an equivalent affine map $\mathcal{T}$ by finding the preimage of the Oil subspace $\mathcal{O} = \{(x_1, \ldots, x_n) \in \mathbb{F}^n : x_1 = \ldots = x_v = 0\}$ under $\mathcal{T}$. To do this, one computes the invariant subspaces of matrices $W = (P^{(i)})^{-1} \cdot \sum_{j=1}^{o} \lambda_j \cdot P^{(j)}$ which, with a certain probability, are also subspaces of $\mathcal{T}^{-1}(\mathcal{O})$. Here, $P^{(i)}$ is the symmetric matrix associated to the homogeneous quadratic part of the i-th component of the public key (see equation 2.14). Table 8.7 shows the base 2 logarithm of the number of matrices W we had to test until finding a basis of $\mathcal{T}^{-1}(\mathcal{O})$.

8.8 Parameters and Comparison

Based on our security analysis (see previous section) we propose for our improved schemes the same parameters as for the standard UOV scheme (see Table 5.6), namely (for 80 bit security)

- $(o, v) = (40, 80)$ for UOV like schemes over GF(16),
- $(o, v) = (33, 66)$ for UOV like schemes over GF(31) and
- $(o, v) = (28, 56)$ for UOV like schemes over GF(256).

security level (bit)			hash size (bit)	signature size (bit)	private key size (kB)	public key size (kB)	reduction factor
80	GF(16)	UOV(40,80)	160	480	135.2	144.2	-
		cyclicUOV(40,80)	160	480	135.2	21.5	6.7
		UOVLRS(40,80)	160	480	135.2	18.4	7.8
		UOVrand(40,80) [1]	160	480	135.2	18.4	7.8
		0/1UOV(40,80)	160	480	135.2	49.8	2.9
	GF(31)	UOV(33,66)	160	528	102.9	108.5	-
		cyclicUOV(33,36)	160	528	102.9	17.1	6.4
		UOVLRS(33,66)	160	528	102.9	14.2	7.6
		UOVrand(33,66) [1]	160	528	102.9	14.2	7.6
		0/1UOV(33,66)	160	528	102.9	31.9	3.4
	GF(256)	UOV(28,56)	224	672	95.8	99.9	-
		cyclicUOV(28,56)	224	672	95.8	16.5	6.0
		UOVLRS(28,56)	224	672	95.8	13.5	7.4
		UOVLRS2(28,56)	224	672	95.8	13.5	7.4
		UOVrand(28,56) [1]	224	672	95.8	13.4	7.4
		0/1 UOV(28,56)	224	672	95.8	24.1	4.1
100	GF(256)	UOV(35,70)	280	840	183.2	193.8	-
		cyclicUOV(35,70)	280	840	183.2	30.0	6.5
		UOVLRS(35,70)	280	840	183.2	25.2	7.7
		UOVLRS2(35,70)	280	840	183.2	25.2	7.7
		UOVrand (35,70) [1]	280	840	183.2	25.1	7.7
		0/1 UOV(35,70)	280	840	183.2	46.1	4.2
128	GF(256)	UOV(45,90)	360	1080	381.8	409.4	-
		cyclicUOV(45,90)	360	1080	381.8	59.4	6.9
		UOVLRS(45,90)	360	1080	381.8	51.5	7.9
		UOVLRS2(45,90)	360	1080	381.8	51.4	8.0
		UOVrand(45,90) [1]	360	1080	381.8	51.3	8.0
		0/1 UOV(45,90)	360	1080	381.8	96.0	4.3

[1] using a PRNG with seed of 128 bit (e.g. AES in OFB mode)

Table 8.8: Key sizes for improved versions of UOV

For a security level of 100 bits we need

- $(o, v) = (50, 100)$ for UOV like schemes over GF(16),
- $(o, v) = (41, 82)$ for UOV like schemes over GF(31) and
- $(o, v) = (35, 70)$ for UOV like schemes over GF(256).

For 128 bit security we propose

- $(o, v) = (64, 128)$ for UOV like schemes over GF(16),
- $(o, v) = (52, 104)$ for UOV like schemes over GF(31) and
- $(o, v) = (45, 90)$ for UOV like schemes over GF(256).

Table 8.8 shows the key sizes of our improved schemes and compares them with those of the standard UOV scheme. Note that the UOVLRS2 scheme does not work over GF(16) and GF(31) (see Remark 8.5). Furthermore we omit the UOVLRScyc scheme here, since the results are the same as for UOVLRS.

security level (bit)	parameters $(\mathbb{F}, o, v)$	key generation time (ms)	key generation cycles (10^6)	signature generation time (ms)	signature generation cycles (10^6)	signature verification	signature verification time (ms)	signature verification cycles (10^6)
80	(GF(256),28,56)	37,152	93,473	4.521	11.201	cyclicUOV	0.23	0.58
						UOVLRS2	0.20	0.50
100	(GF(256),35,70)	111,249	279,780	7.818	18.818	cyclicUOV	0.40	1.00
						UOVLRS2	0.36	0.90
128	(GF(256),45,90)	378,564	952,362	14.01	35.22	cyclicUOV	0.80	2.01
						UOVLRS2	0.72	1.82

Table 8.9: Running time of improved versions of UOV

8.9 Implementation

In this section we give some details about the implementation of our improved versions of UOV. The implementation was done in C and runs on a Lenovo ThinkPad with one Intel Core 2Duo processor with 2.53 GHz and 4 GB of main memory.

Key Generation: The key generation process is the most expensive part of our implementation. During this step we have to perform the inversion of the large matrix $A_{\text{UOV}} \in \mathbb{F}^{D \times D}$ (for the parameters $(o, v) = (28, 56)$ this matrix consists of 3146 rows and columns). To do this efficiently, we used the M4RIE library [1], which performs the inversion using Gauss-Newton-John Elimination. The main idea behind this can be described as follows: For each row $A[i]$ of the matrix A_{UOV} the multiples $c \cdot A[i]$ ($c \in \mathbb{F}$) are precomputed and stored in the Newton-John table. The elimination step can therefore be done by adding one row of A_{UOV} with one row of the Newton-John table. By doing so, the matrix inversion can be performed using $\mathcal{O}(D^2)$ multiplications and $\mathcal{O}(D^3)$ additions.
To speed up the signature generation process, we furthermore store the inverted map $\mathcal{T}^{-1}$ (instead of $\mathcal{T}$) in the private key.

Signature Generation: The signature generation process is done as for the standard UOV scheme. To sign a message d with hash value $\mathbf{h}$, we first choose the values of the Vinegar variables at random and solve the resulting system for the Oil variables by Gaussian Elimination. After that we use the map $\mathcal{T}^{-1}$ to compute a signature for the message d. As we find, for this step it is not suitable to use the M4RIE library, as the matrices to be inverted are quite small.

Signature verification: The signature verification step is the most interesting part of the implementation. For two of our improved versions of UOV (namely cyclicUOV and UOVLRS2) we can use the structure in the public key to speed up the computations (more information on this can be found in Chapter 10 of this thesis).

Table 8.9 shows the running time of our implementation. As the table shows, the key generation process of our schemes is quite expensive and takes a lot of time. However, since this process has to be performed only once during the lifetime of a key, we do not think that this is a major problem.

Chapter 9

Improved Versions of Rainbow

In this chapter we present two different improved versions of the Rainbow signature scheme, which reduce the public key size of the original Rainbow scheme by a factor of up to 3.1. To achieve this, we follow the approach described in Section 7.2. We show how to choose the matrices B_i $(i = 1 \ldots, u)$ used in Algorithm 7.2 in order to get a structured public key.

As we find, our description will be much easier if we first define a large matrix $B_{\text{Rainbow}} \in \mathbb{F}^{m \times D_u}$ and then divide it into submatrices as shown in Figure 9.1. We denote the entry of the i-th row of B_{Rainbow} corresponding to the monomial $x_j x_k$ by $B_{jk}^{(i)}$. Note that, analogously to the polynomials of the Rainbow public key, we enumerate the rows of B_{Rainbow} by $B[v_1 + 1], \ldots, B[n]$.

As in Section 7.2 we define $D_0 := 0$ and

$$D_\ell := \frac{v_\ell \cdot (v_\ell + 1)}{2} + o_\ell \cdot v_\ell \quad (\ell = 1, \ldots, u). \tag{9.1}$$

Furthermore we use (unless otherwise stated) the monomial ordering defined in Subsection 7.2.1.

In Section 9.1 and 9.2 we describe two improved versions of the Rainbow signature scheme called cyclicRainbow and RainbowLRS2. Section 9.3 discusses the security of these improved schemes and in Section 9.4 we compare our two schemes with the original Rainbow signature scheme in terms of the public key size. Finally, Section 9.5 gives details about the implementation of our improved schemes.

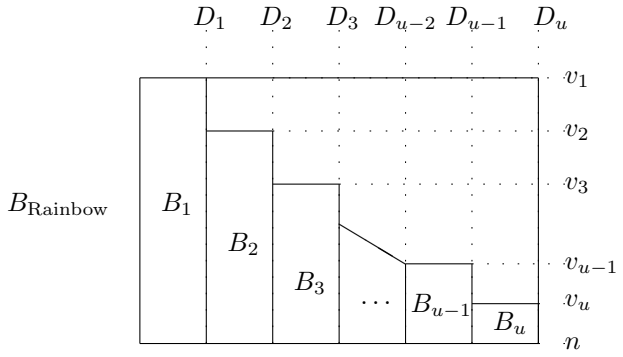


Figure 9.1: Layout of the matrix B_{Rainbow}

9.1 CyclicRainbow

The first improved version of the Rainbow signature scheme we describe here is a variant of Rainbow with a partially circulant public key.
As stated above, we define a large matrix $B_{\text{Rainbow}} \in \mathbb{F}^{m \times D_u}$ and divide it into submatrices as shown in Figure 9.1. In the concrete setting of cyclicRainbow, we choose randomly a vector $\mathbf{b} \in \mathbb{F}^{D_u}$, whose elements are repeated in each row of the matrix B_{Rainbow} (after applying a cyclic shift). In terms of key size reduction, the details of this process play no major role. However, to speed up the verification process (see Chapter 10), it is better to fill the rows of B_{Rainbow} in lexicographic order. Algorithm 9.1 shows the process of generating the matrices $B_1, \ldots, B_u$ in a compact form.

Algorithm 9.1 Generation of the matrices $B_1, \ldots, B_u$ for cyclicRainbow

Input: parameters $(\mathbb{F}, v_1, o_1, \ldots, o_u)$
Output: matrices $B_i \in \mathbb{F}^{(n-v_i)\times(D_i - D_{i-1})}$ $(i = 1, \ldots, u)$ for cyclicRainbow

1: Choose randomly a vector $\mathbf{b} \in \mathbb{F}^{D_u}$.
2: **for** $i = v_1 + 1$ to n **do**
3: $\quad c \leftarrow 1$
4: $\quad$ **for** $j = 1$ to v_u **do**
5: $\quad\quad$ **for** $k = j$ to n **do**
6: $\quad\quad\quad B_{jk}^{(i)} = b_c$
7: $\quad\quad\quad c \leftarrow c + 1$
8: $\quad\quad$ **end for**
9: $\quad$ **end for**
10: $\quad \mathbf{b} \leftarrow \mathcal{R}(\mathbf{b})$ with $\mathcal{R}(\mathbf{b})$ being the cyclic right shift of the vector $\mathbf{b}$ by 1 position.
11: **end for**
12: Divide B into submatrices B_i as shown in Figure 9.1.
13: **return** $B_1, \ldots, B_u$

We use the matrices $B_1, \ldots B_u$ generated by Algorithm 9.1 as input to Algorithm 7.2 to generate a key pair of cyclicRainbow.

Example 9.1. *For* $\mathbb{F} = GF(19)$, $(v_1, o_1, o_2) = (2, 2, 2)$ *we get*

- $D_1 = \frac{v_1 \cdot (v_1+1)}{2} + v_1 \cdot o_1 = 7$ *and*
- $D_2 = \frac{v_2 \cdot (v_2+1)}{2} + v_2 \cdot o_2 = 18$.

We choose the vector $\mathbf{b} \in \mathbb{F}^{18}$ *as* $\mathbf{b} = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18)$ *and get*

$$B_{\text{Rainbow}} = \begin{pmatrix} 1 & 2 & 3 & 4 & 7 & 8 & 9 & 5 & 6 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 \\ 18 & 1 & 2 & 3 & 6 & 7 & 8 & 4 & 5 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 \\ 17 & 18 & 1 & 2 & 5 & 6 & 7 & 3 & 4 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\ 16 & 17 & 18 & 1 & 4 & 5 & 6 & 2 & 3 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \end{pmatrix}.$$

Such we get

$$B_1 = \begin{pmatrix} 1 & 2 & 3 & 4 & 7 & 8 & 9 \\ 18 & 1 & 2 & 3 & 6 & 7 & 8 \\ 17 & 18 & 1 & 2 & 5 & 6 & 7 \\ 16 & 17 & 18 & 1 & 4 & 5 & 6 \end{pmatrix} \in \mathbb{F}^{4\times 7}$$

and

$$B_2 = \begin{pmatrix} 3 & 4 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\ 2 & 3 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \end{pmatrix} \in \mathbb{F}^{2\times 11}$$

which can be used in Algorithm 7.2. We choose

$$\mathcal{S}(\mathbf{x}) = \begin{pmatrix} 5 & 0 & 6 & 0 \\ 13 & 12 & 14 & 15 \\ 4 & 7 & 11 & 17 \\ 13 & 1 & 11 & 14 \end{pmatrix} \cdot \mathbf{x} + \begin{pmatrix} 9 \\ 0 \\ 14 \\ 10 \end{pmatrix}$$

and

$$\mathcal{T}(\mathbf{x}) = \begin{pmatrix} 12 & 9 & 16 & 1 & 12 & 7 \\ 2 & 3 & 13 & 10 & 16 & 11 \\ 13 & 8 & 2 & 18 & 17 & 10 \\ 8 & 8 & 15 & 12 & 8 & 17 \\ 15 & 4 & 3 & 12 & 1 & 4 \\ 16 & 16 & 14 & 16 & 2 & 12 \end{pmatrix} \cdot \mathbf{x} + \begin{pmatrix} 7 \\ 14 \\ 10 \\ 16 \\ 5 \\ 11 \end{pmatrix}$$

and get the public key $\mathcal{P}$ (in matrix form (see equation (2.10))

$$p^{(3)} : (x_1, x_2, x_3, x_4, x_5, x_6, 1) \cdot \begin{pmatrix} 1 & 2 & 3 & 4 & 10 & 3 & 18 \\ 0 & 7 & 8 & 9 & 15 & 14 & 13 \\ 0 & 0 & 11 & 5 & 9 & 15 & 5 \\ 0 & 0 & 0 & 7 & 12 & 0 & 2 \\ 0 & 0 & 0 & 0 & 6 & 8 & 14 \\ 0 & 0 & 0 & 0 & 0 & 12 & 18 \\ 0 & 0 & 0 & 0 & 0 & 0 & 16 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ 1 \end{pmatrix},$$

$$p^{(4)} : (x_1, x_2, x_3, x_4, x_5, x_6, 1) \cdot \begin{pmatrix} 18 & 1 & 2 & 3 & 14 & 9 & 9 \\ 0 & 6 & 7 & 8 & 17 & 18 & 12 \\ 0 & 0 & 7 & 12 & 14 & 8 & 5 \\ 0 & 0 & 0 & 5 & 8 & 0 & 16 \\ 0 & 0 & 0 & 0 & 7 & 4 & 17 \\ 0 & 0 & 0 & 0 & 0 & 6 & 13 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ 1 \end{pmatrix},$$

$$p^{(5)} : (x_1, x_2, x_3, x_4, x_5, x_6, 1) \cdot \begin{pmatrix} 17 & 18 & 1 & 2 & 3 & 4 & 17 \\ 0 & 5 & 6 & 7 & 8 & 9 & 1 \\ 0 & 0 & 10 & 11 & 12 & 13 & 11 \\ 0 & 0 & 0 & 14 & 15 & 16 & 8 \\ 0 & 0 & 0 & 0 & 15 & 3 & 12 \\ 0 & 0 & 0 & 0 & 0 & 15 & 16 \\ 0 & 0 & 0 & 0 & 0 & 0 & 8 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ 1 \end{pmatrix} \text{ and}$$

$$p^{(6)} : (x_1, x_2, x_3, x_4, x_5, x_6, 1) \cdot \begin{pmatrix} 16 & 17 & 18 & 1 & 2 & 3 & 0 \\ 0 & 4 & 5 & 6 & 7 & 8 & 1 \\ 0 & 0 & 9 & 10 & 11 & 12 & 3 \\ 0 & 0 & 0 & 13 & 14 & 15 & 5 \\ 0 & 0 & 0 & 0 & 4 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 15 & 8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 11 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ 1 \end{pmatrix}. \quad \square$$

To store the matrices $B_1, \ldots, B_u$, we have to store only the vector $\mathbf{b} \in \mathbb{F}^{D_u}$. The size of the public key of cyclicRainbow is therefore given by

$$\text{size}_{\text{pk cyclicRainbow}} = D_u + \sum_{\ell=1}^{u} o_\ell \cdot \left(\frac{(n+1)\cdot(n+2)}{2} - D_\ell \right) = D_u + m \cdot \frac{(n+1)\cdot(n+2)}{2} - \sum_{\ell=1}^{u} o_\ell \cdot D_\ell \tag{9.2}$$

field elements, the size of the private key is the same as for the standard Rainbow scheme (see equation (3.18)).

In Section 10.4 we show how the structure of the public key of cyclicRainbow can be used to speed up the verification process of the scheme.

9.2 RainbowLRS2

In this section we show how to use linear recurring sequences (LRS's) to reduce the public key size of the Rainbow signature scheme. As in the case of cyclicRainbow (see previous section) we define a matrix $B_{\text{Rainbow}} \in \mathbb{F}^{m \times D_u}$ and divide it into submatrices to get the matrices B_i $(i = 1, \ldots, u)$ used in Algorithm 7.2. In the concrete setting of RainbowLRS2, we choose two vectors $\alpha = (\alpha_{v_1+1}, \ldots, \alpha_n)$ and $\gamma = (\gamma_{v_1+1}, \ldots, \gamma_n) \in \mathbb{F}^m$ and define m linear recurring sequences $S^{(i)}$ of length 1 by

$$S^{(i)} = \text{LRS}(\alpha_i, \gamma_i \cdot X + 1) \ (i = v_1 + 1, \ldots, n). \tag{9.3}$$

Finally, we put the first D_u elements of $S^{(i)}$ into the i-th row of the matrix B (in graded lexicographic order).
Algorithm 9.2 shows this generation process in a compact form.

Algorithm 9.2 Generation of the matrices $B_1, \ldots, B_u$ for RainbowLRS2

Input: parameters $(\mathbb{F}, v_1, o_1, \ldots, o_u)$
Output: matrices $B_i \in \mathbb{F}^{(n-v_i) \times D_i - D_{i-1}}$ $(i = 1, \ldots, u)$ for RainbowLRS2.

```
Choose two vectors α = (α_{v_1+1}, ..., α_n) and γ = (γ_{v_1+1}, ..., γ_n) ∈ F^m (see Subsection 9.2.1).
for i = v_1 + 1 to n do
    Compute the first D_u elements s_1 ..., s_{D_u} of the LRS S^(i) = LRS(a_i, γ_i · X + 1).
    c ← 1
    for j = 1 to v_u do
        for k = j to n do
            B^(i)_jk ← s_c
            c ← c + 1
        end for
    end for
end for
Divide B into submatrices B_i as shown in Figure 9.1.
return B_1, ..., B_u
```

We use the matrices $B_1, \ldots, B_u$ generated by Algorithm 9.2 as input to Algorithm 7.2 to generate a key pair of RainbowLRS2. In order to store the matrices $B_1, \ldots, B_u$ we have to store only the vectors α and γ.

9.2.1 Choice of α and γ

For RainbowLRS2, the question of choosing the vector γ is a bit more complicated than for the simple case of UOVLRS2 (see Section 8.4). The reason for this is the second affine map $\mathcal{S}$ which mixes the polynomials. So, if two elements of the vector γ are equal, not the whole homogeneous quadratic part of the public polynomials will be linearly dependent. However, there still remain linear dependencies which can be used by Gröbner Basis algorithms (see Table 9.1). Therefore we recommend to choose all the elements of the vector γ to be pairwise distinct. Furthermore, all the elements of the vector γ must be different from zero.
In contrast to this, there are no major conditions on the choice of the vector α. We just need $\alpha_i \in \mathbb{F} \setminus \{0\}$ $\forall i = 1, \ldots, m$. So α can be fixed to $\alpha = (1, \ldots, 1) \in \mathbb{F}^n$.

Remark 9.1. *As in the case of UOVLRS2 (see Section 8.4), the above condition on γ prevents us from using RainbowLRS2 over $GF(16)$ and $GF(31)$.*

$(GF(256), v_1, o_1, o_2)$	$(6,4,5)$	$(7,5,5)$	$(8,5,6)$	$(9,6,6)$	$(10,6,7)$	$(11,7,7)$
$L = m$	5.2 s	40.6 s	291.9 s	2,381 s	19,027 s	168,562 s
$L = m - 1$	5.1 s	39.1 s	287.8 s	2,355 s	18,876 s	145,885 s
$L = m - 2$	2.8 s	18.4 s	135.7 s	1,034 s	8,836 s	82,492 s

Table 9.1: Running time of the direct attack against RainbowLRS2 for different values of L

$(GF(256), v_1, o_1, o_2)$	(6,4,5)	(7,5,5)	(7,5,6)	(8,6,6)	(9,6,7)	(10,7,7)
Rainbow	5.3 s	40.6 s	292.5 s	2,382 s	19,031 s	168,578 s
cyclicRainbow	5.3 s	40.5 s	292.3 s	2,379 s	19,025 s	168,521 s
RainbowLRS2	5.2 s	40.6 s	291.9 s	2,381 s	19,027 s	168,562 s

Table 9.2: Running time of the direct attack against Rainbow like schemes

Therefore, in order to store the matrices $B_1, \ldots, B_u$, we have to store only the m elements of the vector γ.
The size of the public key of RainbowLRS2 is therefore given by

$$\text{size}_{\text{pk RainbowLRS2}} = m + \sum_{\ell=1}^{u} o_\ell \cdot \left(\frac{(n+1)\cdot(n+2)}{2} - D_\ell \right) = m \cdot \frac{(n+1)\cdot(n+2)+2}{2} - \sum_{\ell=1}^{u} o_\ell \cdot D_\ell \tag{9.4}$$

field elements, the size of the private key is the same as for the standard Rainbow scheme (see equation (3.18)).

In Section 10.5 we show how the structure of the RainbowLRS2 public key can be used to speed up the verification process of the scheme.

9.3 Security

We analyzed the security of our improved versions of the Rainbow signature scheme against known attacks. These include (see Section 3.6):

- direct attacks
- Rank attacks (MinRank and HighRank)
- the Rainbow-Band-Separation attack and
- attacks against UOV (UOV-Reconciliation and UOV attack).

Direct attacks

The most straightforward way to attack multivariate schemes like Rainbow is by solving the public system $\mathcal{P}(\mathbf{x}) = \mathbf{h}$ directly by the XL-Algorithm or a Gröbner Basis method (see Section 2.5). These attacks belong therefore to the area of signature forgery attacks. To analyze the security of our improved schemes against these attacks, we carried out a number of experiments with MAGMA [8] v. 2.13-10, which contains an efficient implementation of Faugère's F_4 algorithm [23]. For each of the improved versions of Rainbow presented in this chapter and each parameter set we created 100 public systems $\mathcal{P}(\mathbf{x}) = \mathbf{h}$ and solved them, after fixing $n - m$ variables to get a determined system, using the MAGMA command `Variety`. Table 9.2 shows the results.

$(GF(256), v_1, o_1, o_2)$	(3,2,2)	(4,2,3)	(5,3,3)	(6,3,4)
Rainbow	24.3	32.5	40.1	48.3
cyclicRainbow	24.2	32.1	40.0	48.2
RainbowLRS2	24.2	32.3	40.1	48.1

Table 9.3: Results of our experiments with the MinRank attack against Rainbow like schemes

$(GF(256), v_1, o_1, o_2)$	(3,2,2)	(4,2,3)	(5,3,3)	(6,3,4)
Rainbow	16.7	24.6	24.7	32.4
cyclicRainbow	16.5	24.5	24.5	32.3
RainbowLRS2	16.4	24.4	24.6	32.2

Table 9.4: Results of our experiments with the HighRank attack against Rainbow like schemes

MinRank attack

The goal of the MinRank attack (see Subsection 3.6.2) is to find linear combinations of the matrices $P^{(k)}$ associated to the homogeneous quadratic part of the k-th component of the public key (see equation (2.14)) of rank $r \leq v_2$. These linear combinations correspond to linear combinations of the central polynomial of the first Rainbow layer. Finding o_1 of these linear combinations therefore enables the attacker to separate the first Rainbow layer from the other polynomials. We implemented the MinRank attack as presented in [6] in MAGMA and used it to attack our improved Rainbow schemes. Table 9.3 shows the base 2 logarithm of the number of linear combinations we had to test until finding a matrix of rank $\leq v_2$.

HighRank attack

The goal of the HighRank attack (see Subsection 3.6.3) is to find the space $\mathcal{T}^{-1}(\mathcal{O}_u)$ for $\mathcal{O}_u = \{x \in \mathbb{F}^n : x_1 = \ldots = x_{v_u} = 0\}$ by looking at the kernel of linear combinations of the matrices $P^{(k)}$. Table 9.4 shows the base 2 logarithm of the number of linear combinations we had to test to find a basis of $\mathcal{T}^{-1}(\mathcal{O}_u)$.

Rainbow-Band-Separation attack

The goal of the Rainbow-Band-Separation (RBS) attack (see Subsection 3.6.4) is to find an equivalent Rainbow private key by stepwise creating a linear transformation which brings the matrices $P^{(k)}$ into the form of a Rainbow central map. To do this, the attacker has to solve several systems of multivariate equations. Table 9.5 shows the time MAGMA needs to solve the first and largest of these systems for the standard Rainbow scheme as well as for our improved versions.

UOV-Reconciliation attack

Since Rainbow can be seen as a UOV scheme with v_u Vinegar variables and o_u Oil variables, it can be attacked by the UOV-Reconciliation attack (see Subsection 3.3.2). The goal of this attack

$(GF(256), v_1, o_1, o_2)$	(6,4,5)	(7,5,5)	(7,5,6)	(8,6,6)	(9,6,7)	(10,7,7)
Rainbow	6.9 s	41.2 s	299.2 s	4,206 s	21,302 s	415,458 s
cyclicRainbow	6.8 s	41.0 s	298.3 s	4,207 s	21,319 s	414,962 s
RainbowLRS2	6.7 s	40.9 s	297.9 s	4,202 s	21,278 s	415,243 s

Table 9.5: Running time of the RBS attack against Rainbow like schemes

$(GF(256), v_1, o_1, o_2)$	(6,4,5)	(7,5,5)	(7,5,6)	(8,6,6)	(9,6,7)	(10,7,7)
Rainbow	5.4 s	38.3 s	289.7 s	2,370 s	19,235 s	147,641 s
cyclicRainbow	5.3 s	38.2 s	289.5 s	2,367 s	19,231 s	147,484 s
RainbowLRS2	5.3 s	38.1 s	289.3 s	2,362 s	19,227 s	147,521 s

Table 9.6: Running time of UOV-Reconciliation attack against Rainbow like schemes

$(256, v_1, o_1, o_2)$	(3,2,2)	(4,2,3)	(5,3,3)	(6,3,4)
Rainbow	24.4	24.5	40.4	40.6
cyclicRainbow	24.3	24.5	40.4	40.5
RainbowLRS2	24.3	24.4	40.3	40.4

Table 9.7: Results of the experiments with the UOV attack against Rainbow like schemes

is to find a linear transformation of the variables which brings the matrices $P^{(k)}$ into the form of a UOV(v_u, o_u) central map. To find this transformation, one has to solve a number of multivariate quadratic systems. Table 9.6 shows the time MAGMA needs to solve the first and largest of these systems (m equations in v_u variables).

UOV attack

As in the case of the UOV-Reconciliation attack we here consider Rainbow as an UOV scheme with v_u Vinegar variables and o_u Oil variables. The goal of the UOV attack (see Subsection 3.3.3) is to find an equivalent private key by finding the preimage of the Oil-subspace $\mathcal{O}_u = \{x \in \mathbb{F}^n : x_1 = \ldots = x_{v_u} = 0\}$ under the affine map $\mathcal{T}$. We do this by looking at the invariant subspaces of matrices of the form $W = (P^{(i)})^{-1} \cdot \sum_{j=v_1+1}^{n} \lambda_j \cdot P^{(j)}$. Table 9.7 shows the base 2 logarithm of the number of matrices W we had to test until finding a basis of $\mathcal{T}^{-1}(\mathcal{O}_u)$.

Summary

As the Tables 9.2 to 9.7 show, none of the known attacks against the Rainbow signature scheme is significantly more efficient for any of our improved versions. The reason for this is that these attacks do not look too closely at the inner structure of the polynomials and therefore can not use the special structure of our public keys. However, it remains an open question if one can develop dedicated attacks which use this structure.

security level (bit)			hash size (bit)	signature size (bit)	private key size (kB)	public key size (kB)	reduction factor
80	GF(16)	Rainbow(17,23,17)	160	228	21.9	33.4	-
		cyclicRainbow(17,23,17)	160	228	21.9	15.6	2.1
	GF(31)	Rainbow(14,19,14)	160	256	17.1	25.3	-
		cyclicRainbow(14,19,14)	160	256	17.1	12.0	2.1
	GF(256)	Rainbow(17,13,13)	208	344	19.1	25.1	-
		cyclicRainbow(17,13,13)	208	344	19.1	10.4	2.4
		RainbowLRS2(17,13,13)	208	344	19.1	9.6	2.6
100	GF(256)	Rainbow(26,16,17)	264	472	45.0	59.0	-
		cyclicRainbow(26,16,17)	264	472	45.0	21.7	2.7
		RainbowLRS2(26,16,17)	264	472	45.0	20.2	2.9
128	GF(256)	Rainbow(36,21,22)	344	632	101.5	136.1	-
		cyclicRainbow(36,21,22)	344	632	102.5	47.3	2.9
		RainbowLRS2(36,21,22)	344	632	102.5	44.5	3.1

Table 9.8: Key sizes of Rainbow like schemes

9.4 Parameters and Comparison

Based on our security analysis presented in the previous section we can use for our improved schemes the same parameters as for the standard Rainbow signature scheme (see Chapter 6). However, to find a good compromise between public and private key size, we choose the sizes of the middle and the last layer to be as equal as possible. Note that this is not possible for Rainbow schemes over GF(16) and GF(31), since we need $v_1 \geq o_u$ to defend the scheme against the UOV-Reconciliation attack.
So we propose (for 80 bit security)

- $(v_1, o_1, o_2) = (17, 23, 17)$ for Rainbow like schemes over GF(16),
- $(v_1, o_1, o_2) = (14, 19, 14)$ for Rainbow like schemes over GF(31) and
- $(v_1, o_1, o_2) = (17, 13, 13)$ for Rainbow like schemes over GF(256).

For 100 bit security we propose the parameters

- $(v_1, o_1, o_2) = (22, 28, 22)$ for Rainbow like schemes over GF(16),
- $(v_1, o_1, o_2) = (18, 23, 18)$ for Rainbow like schemes over GF(31) and
- $(v_1, o_1, o_2) = (26, 16, 17)$ for Rainbow like schemes over GF(256).

For a security level of 128 bits we get the parameters

- $(v_1, o_1, o_2) = (29, 35, 29)$ for Rainbow like schemes over GF(16),
- $(v_1, o_1, o_2) = (23, 29, 23)$ for Rainbow like schemes over GF(31) and
- $(v_1, o_1, o_2) = (36, 21, 22)$ for Rainbow like schemes over GF(256).

Table 9.8 shows the key sizes of our improved schemes and compares them with the original Rainbow scheme. Note that RainbowLRS2 can not be used over the fields GF(16) and GF(31).

security level (bit)	parameters $(\mathbb{F}, v_1, o_1, o_2)$	key generation time (ms)	key generation cycles (10^6)	signature generation time (ms)	signature generation cycles (10^6)	signature verification	signature verification time (ms)	signature verification cycles (10^6)
80	(GF(256),17,13,13)	2,377	6,133	2.01	4.74	cyclicRainbow	0.14	0.35
						RainbowLRS2	0.15	0.38
100	(GF(256),26,16,17)	9,957	25,693	2.99	7.40	cyclicRainbow	0.29	0.72
						RainbowLRS2	0.31	0.78
128	(GF(256),36,21,22)	39,008	100,660	4.95	12.53	cyclicRainbow	0.63	1.59
						RainbowLRS2	0.67	1.69

Table 9.9: Running times of Rainbow like schemes

9.5 Implementation

In this section we give some details about the implementation of our improved versions of Rainbow. The implementation was done in C and runs on a Lenovo ThinkPad with one Intel Core 2Duo processor with 2.53 GHz and 4 GB of main memory.

Key Generation: The key generation process is the most expensive part of our implementation. Similarly to the case of UOV we have to invert the large matrices $\tilde{A}_1, \ldots, \tilde{A}_u$. For the parameters $(v_1, o_1, o_2) = (17, 13, 13)$ the biggest of these matrices (the matrix $\tilde{A}_2$) consists of 855 rows and columns. As in the case of UOV (see Section 8.9) we use for this step the M4RIE library [1]. Furthermore, we store the inverted maps $\mathcal{S}^{-1}$ and $\mathcal{T}^{-1}$ (instead of $\mathcal{S}$ and $\mathcal{T}$) in the private key to speed up the signature generation process.

Signature Generation: The signature generation process is done as for the standard Rainbow scheme. To sign a message d with hash value $\mathbf{h}$, we first use the map $\mathcal{S}^{-1}$ to compute $\mathbf{x} = \mathcal{S}^{-1}(\mathbf{h})$. Then we choose the values of the Vinegar variables of the first layer at random and solve the resulting system for the variables x_i $(i \in O_1)$ by Gaussian Elimination. We substitute these values into the remaining polynomials and solve the resulting system for the variables x_i $(i \in O_2)$. Finally, we use the map $\mathcal{T}^{-1}$ to compute a signature for the message d. As in Section 8.9, we do not use the M4RIE library during this step.

Signature verification: The signature verification step is the most interesting part of the implementation. For both of the proposed schemes we can use the structure of the public key to speed up the computations (more information on this step can be found in Chapter 10 of this thesis).

Table 9.9 shows the running time of our implementation. As for the improved versions of UOV, the key generation process of our schemes is quite expensive and takes a lot of time. However, since this process has to be performed only once during the lifetime of a key, we do not think that this is a major problem.

Chapter 10

Speeding up the Verification Process

In this chapter we show how the structure of the public keys of our improved versions of UOV and Rainbow can be used to speed up the verification process of the schemes. After the description of the generic verification process of multivariate signature schemes (Section 10.1), we show in Sections 10.2 - 10.5 how this step can be improved for the improved schemes cyclicUOV, UOVLRS2, cyclicRainbow and RainbowLRS2 (see Sections 8.1, 8.4, 9.1 and 9.2). Finally, Section 10.6 presents the results of our computer experiments and compares the running time of the verification process of our improved versions with that of the original schemes.

10.1 Verification Process of Multivariate Signature Schemes

Let $\mathcal{P} = (p^{(1)}, \ldots, p^{(m)})$ be the public key of a multivariate signature scheme. Let $\mathbf{z} = (z_1, \ldots, z_n) \in \mathbb{F}^n$ be a (valid or invalid) signature of a message d with hash value $\mathbf{h} = \mathcal{H}(d) = (h_1, \ldots, h_m) \in \mathbb{F}^m$. The standard verification process looks as shown in Algorithm 10.1.

Algorithm 10.1 Generic verification process of multivariate signature schemes

Input: public key $\mathcal{P} = (p^{(1)}, \ldots, p^{(m)})$ of a multivariate signature scheme, signature $\mathbf{z} \in \mathbb{F}^n$, hash value $\mathbf{h} \in \mathbb{F}^m$
Output: boolean value TRUE or FALSE
1: **for** $k = 1$ to m **do**
2: $\quad h'_k \leftarrow \text{Evaluate}(p^{(k)}, \mathbf{z})$
3: $\quad$ **if** $h'_k \neq h_k$ **then return** FALSE
4: $\quad$ **end if**
5: **end for**
6: **return** TRUE

As can be seen from Algorithm 10.1, invalid signatures can be detected usually very fast. As soon as the test in line 3 is fulfilled, the algorithm stops and the signature is rejected. On the contrary, for valid signatures, the test in line 3 has to be performed m times and we have to evaluate all the polynomials $p^{(1)}, \ldots, p^{(m)}$. Therefore, the running time of the verification algorithm is quite asymmetric for valid and invalid signatures. This is in contrast to signature schemes like RSA and DSA, where, regardless of the validity of a signature, the same operations have to be performed. Furthermore, we see that the central step of the verification process is the evaluation of the public polynomials $p^{(1)}, \ldots, p^{(m)}$ (line 2). To perform this step, there are basically two possibilities:

In the first way (later referred to as the standard approach), one computes for a given (valid or invalid) signature $\mathbf{z} = (z_1, \ldots, z_n) \in \mathbb{F}^n$ an $\frac{(n+1)\cdot(n+2)}{2}$ vector mon, which contains the values of all monomials of degree ≤ 2, i.e.

$$\text{mon} = (z_1^2, z_1 z_2, \ldots, z_n^2, z_1, \ldots, z_n, 1). \tag{10.1}$$

Then we have

$$p^{(k)}(\mathbf{z}) = M_P[k] \cdot \text{mon}^T \quad (k = 1, \ldots, m), \tag{10.2}$$

with $M_P[k]$ being the k-th row of the Macauley matrix M_P (see Definition 2.6) and $\cdot$ being the standard scalar product.
Note that for both the vector mon and the matrix M_P we use the graded lexicographic order.
To evaluate the system $\mathcal{P} = (p^{(1)}, \ldots, p^{(m)})$ in this way, one needs

- $\frac{n\cdot(n+1)}{2}$ field multiplications to compute the vector mon and
- $m \cdot \left(\frac{n\cdot(n+1)}{2} + n\right)$ field multiplications to compute the scalar products in equation (10.2).

Altogether, to verify a valid signature in this way, one needs

$$\frac{n}{2} \cdot ((n+1) \cdot (m+1) + 2 \cdot m) \tag{10.3}$$

field multiplications.

Remark 10.1. *As mentioned above, the verification algorithm stops as soon as it has found an $k \in \{1, \ldots, m\}$ with $p^{(k)}(\mathbf{z}) \neq h_k$. Therefore, for invalid signatures, the number of multiplications is usually less than stated by formula* (10.3).

The second strategy (later referred to as the alternative approach) can be described as follows:
We write the public polynomials in matrix form as shown in equation (2.10), i.e.

$$MP^{(k)} = \begin{pmatrix} p_{11}^{(k)} & p_{12}^{(k)} & p_{13}^{(k)} & \ldots & p_{1n}^{(k)} & p_1^{(k)} \\ 0 & p_{22}^{(k)} & p_{23}^{(k)} & \ldots & p_{2n}^{(k)} & p_2^{(k)} \\ 0 & 0 & p_{33}^{(k)} & & p_{3n}^{(k)} & p_3^{(k)} \\ \vdots & & \ddots & & \vdots & \vdots \\ 0 & 0 & \ldots & 0 & p_{nn}^{(k)} & p_n^{(k)} \\ 0 & 0 & \ldots & 0 & 0 & p_0^{(k)} \end{pmatrix} \quad (k = 1, \ldots, m). \tag{10.4}$$

For a (valid or invalid) signature $\mathbf{z} = (z_1, \ldots, z_n)$ we define the extended signature vector

$$\text{sign} = (z_1, \ldots, z_n, 1). \tag{10.5}$$

With this notation we can evaluate the polynomial $p^{(k)}$ $(k = 1, \ldots, m)$ by

$$p^{(k)}(\mathbf{z}) = \text{sign} \cdot MP^{(k)} \cdot \text{sign}^T \quad (k = 1, \ldots, m). \tag{10.6}$$

To evaluate a single polynomial $p(x_1, \ldots, x_n)$ with associated matrix MP in this way, one needs

- $\frac{(n+1)\cdot(n+2)}{2}$ field multiplications to compute the product $\text{temp} = \text{sign} \cdot MP$ and
- $n + 1$ field multiplications to compute the scalar product $\text{temp} \cdot \text{sign}^T$.

		standard approach		alternative approach	
		running time (ms)	CPU cycles (10^6)	running time (ms)	CPU cycles (10^6)
GF(16)	UOV(40,80)	2.76	6.97	3.08	7.78
	Rainbow(17,23,17)	0.65	1.63	0.72	1.84
GF(31)	UOV(33,66)	2.01	5.08	2.12	5.21
	Rainbow(14,19,14)	0.49	1.23	0.55	1.34
GF(256)	UOV(28,56)	1.13	2.87	1.18	2.93
	Rainbow(17,13,13)	0.29	0.72	0.31	0.83

Table 10.1: Comparison of our two evaluation strategies for UOV and Rainbow schemes (80 bit security)

Therefore, to verify a valid signature in this way, we need

$$m \cdot \frac{(n+1)\cdot(n+4)}{2} \tag{10.7}$$

field multiplications. As mentioned above, for an invalid signature, this number is usually smaller.

Table 10.1 compares the two strategies for the verification process regarding their efficiency. For this we created a straightforward C implementation of the UOV and Rainbow signature schemes and verified 100,000 valid signatures using both of the above strategies. The table shows the average running time as well as the number of CPU cycles needed during this process.

As Table 10.1 shows, for the standard UOV and Rainbow schemes, evaluating the polynomials by the standard approach is slightly more efficient. But, as we will see in the following sections, for schemes with structured public key we can do much better with the alternative approach.

10.2 CyclicUOV

In the case of cyclicUOV (see Section 8.1), the matrices $MP^{(k)}$ are of the form shown in Figure 10.1. We have

$$MP^{(k)}_{ij} = MP^{(k-1)}_{i,j-1} \;\forall\; i \in \{1,\ldots,v\},\; j \in \{i+1,\ldots,n\},\; k \in \{2,\ldots,o\}. \tag{10.8}$$

Therefore we get

$$(\text{sign}_1,\ldots,\text{sign}_i) \cdot \begin{pmatrix} MP^{(k)}_{1j} \\ MP^{(k)}_{2j} \\ \vdots \\ MP^{(k)}_{ij} \end{pmatrix} = (\text{sign}_1,\ldots,\text{sign}_i) \cdot \begin{pmatrix} MP^{(k-1)}_{1,j-1} \\ MP^{(k-1)}_{2,j-1} \\ \vdots \\ MP^{(k-1)}_{i,j-1} \end{pmatrix}$$
$$\forall i \in \{1,\ldots v\},\; j \in \{i+1,\ldots,n\},\; k \in \{2,\ldots,o\}. \tag{10.9}$$

The boxes in Figure 10.1 illustrate this equation. The black boxes show the vector $(MP^{(k-1)}_{1,j-1},\ldots,MP^{(k-1)}_{i,j-1})^T$ on the right hand side of the equation, whereas the blue boxes represent the vector $(MP^{(k)}_{1,j},\ldots,MP^{(k)}_{i,j})^T$ on the left hand side. As one can see, the blue boxes in the matrix $MP^{(k)}$ are exactly the same as the black boxes in the matrix $MP^{(k-1)}$ $(k = 2,\ldots,o)$. By using this relation, we can speed up the verification process of cyclicUOV by a large factor (see Algorithm 10.2).

$$
MP^{(1)} = \begin{pmatrix}
\boxed{b_1} & \boxed{b_2} & \boxed{b_3} & \dots & \boxed{b_{v-1}} & \boxed{b_v} & b_{v+1} & \dots & b_{n-1} & b_n & \star \\
0 & b_{n+1} & b_{n+2} & \dots & b_{n+v-2} & b_{n+v-1} & b_{n+v} & \dots & b_{2n-2} & b_{2n-1} & \star \\
0 & 0 & b_{2n} & \dots & b_{2n+v-4} & b_{2n+v-3} & b_{2n+v-2} & \dots & b_{3n-4} & b_{3n-3} & \star \\
\vdots & & \ddots & & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\
0 & & \dots & 0 & b_{D-2o-1} & b_{D-2o} & b_{D-2o+1} & \dots & b_{D-o-2} & b_{D-o-1} & \star \\
0 & & \dots & & 0 & b_{D-o} & b_{D-o+1} & \dots & b_{D-1} & b_D & \star \\
0 & & \dots & & & 0 & \star & \dots & \star & \star & \star \\
\vdots & & & & & & \ddots & & \vdots & \vdots & \vdots \\
\vdots & & & & & & & \ddots & \vdots & \vdots & \vdots \\
0 & & \dots & & \dots & & \dots & & 0 & \star & \star \\
0 & & \dots & & \dots & & \dots & & & 0 & \star
\end{pmatrix}
$$

$$
MP^{(2)} = \begin{pmatrix}
\boxed{b_D} & \boxed{b_1} & \boxed{b_2} & \dots & \boxed{b_{v-2}} & \boxed{b_{v-1}} & \boxed{b_v} & \dots & \boxed{b_{n-2}} & \boxed{b_{n-1}} & \star \\
0 & b_n & b_{n+1} & \dots & b_{n+v-3} & b_{n+v-2} & b_{n+v-1} & \dots & b_{2n-3} & b_{2n-2} & \star \\
0 & 0 & b_{2n-1} & \dots & b_{2n+v-5} & b_{2n+v-4} & b_{2n+v-3} & \dots & b_{3n-5} & b_{3n-4} & \star \\
\vdots & & \ddots & & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\
0 & & \dots & 0 & b_{D-2o-2} & b_{D-2o-1} & b_{D-2o} & \dots & b_{D-o-3} & b_{D-o-2} & \star \\
0 & & \dots & & 0 & b_{D-o-1} & b_{D-o} & \dots & b_{D-2} & b_{D-1} & \star \\
0 & & \dots & & & 0 & \star & \dots & \star & \star & \star \\
\vdots & & & & & & \ddots & & \vdots & \vdots & \vdots \\
\vdots & & & & & & & \ddots & \vdots & \vdots & \vdots \\
0 & & \dots & & \dots & & \dots & & 0 & \star & \star \\
0 & & \dots & & \dots & & \dots & & & 0 & \star
\end{pmatrix}
$$

$$
MP^{(o-1)} = \begin{pmatrix}
 & & & & \vdots & & & & & & \\
\boxed{b_{D-o+3}} & \boxed{b_{D-o+4}} & \boxed{b_{D-o+5}} & \dots & \boxed{b_{o+1}} & \boxed{b_{o+2}} & \boxed{b_{o+3}} & \dots & \boxed{b_{v+1}} & \boxed{b_{v+2}} & \star \\
0 & b_{v+3} & b_{v+4} & \dots & b_{n+o} & b_{n+o+1} & b_{n+o+2} & \dots & b_{n+v} & b_{n+v+1} & \star \\
0 & 0 & b_{n+v+2} & \dots & b_{2n+o-2} & b_{2n+o-1} & b_{2n+o} & \dots & b_{2n+v-2} & b_{2n+v-1} & \star \\
\vdots & & \ddots & & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\
0 & & \dots & 0 & b_{D-3o+1} & b_{D-3o+2} & b_{D-3o+3} & \dots & b_{D-2o} & b_{D-2o+1} & \star \\
0 & & \dots & & 0 & b_{D-2o+2} & b_{D-2o+3} & \dots & b_{D-o+1} & b_{D-o+2} & \star \\
0 & & \dots & & & 0 & \star & \dots & \star & \star & \star \\
\vdots & & & & & & \ddots & & \vdots & \vdots & \vdots \\
\vdots & & & & & & & \ddots & \vdots & \vdots & \vdots \\
0 & & \dots & & \dots & & \dots & & 0 & \star & \star \\
0 & & \dots & & \dots & & \dots & & & 0 & \star
\end{pmatrix}
$$

$$
MP^{(o)} = \begin{pmatrix}
b_{D-o+2} & \boxed{b_{D-o+3}} & \boxed{b_{D-o+4}} & \dots & \boxed{b_o} & \boxed{b_{o+1}} & \boxed{b_{o+2}} & \dots & \boxed{b_v} & \boxed{b_{v+1}} & \star \\
0 & b_{v+2} & b_{v+3} & \dots & b_{n+o-1} & b_{n+o} & b_{n+o+1} & \dots & b_{n+v-1} & b_{n+v} & \star \\
0 & 0 & b_{n+v+1} & \dots & b_{2n+o-3} & b_{2n+o-2} & b_{2n+o-1} & \dots & b_{2n+v-3} & b_{2n+v-2} & \star \\
\vdots & & \ddots & & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\
0 & & \dots & 0 & b_{D-3o} & b_{D-3o+1} & b_{D-3o+2} & \dots & b_{D-2o-1} & b_{D-2o} & \star \\
0 & & \dots & & 0 & b_{D-2o+1} & b_{D-2o+2} & \dots & b_{D-o} & b_{D-o+1} & \star \\
0 & & \dots & & & 0 & \star & \dots & \star & \star & \star \\
\vdots & & & & & & \ddots & & \vdots & \vdots & \vdots \\
\vdots & & & & & & & \ddots & \vdots & \vdots & \vdots \\
0 & & \dots & & \dots & & \dots & & 0 & \star & \star \\
0 & & \dots & & \dots & & \dots & & & 0 & \star
\end{pmatrix}
$$

Figure 10.1: Matrices $MP^{(i)}$ for cyclicUOV

Algorithm 10.2 Verification process of cyclicUOV

Input: public key of cyclicUOV (given in matrix form), signature $\mathbf{z} \in \mathbb{F}^n$, hash value $\mathbf{h} \in \mathbb{F}^m$
Output: boolean value TRUE or FALSE

1: $\text{sign} \leftarrow (\mathbf{z}, 1)$
2: **for** $i = 1$ to $n-1$ **do** ▷ first polynomial ($p^{(1)}$)
3: $\quad a_i \leftarrow \sum_{j=1}^{\min(i,v)} MP_{ji}^{(1)} \cdot \text{sign}_j$
4: $\quad \text{temp}_i \leftarrow a_i$
5: **end for**
6: **for** $i = v+1$ to $n-1$ **do**
7: $\quad \text{temp}_i \leftarrow a_i + \sum_{j=v+1}^{i} MP_{ji}^{(1)} \cdot \text{sign}_j$
8: **end for**
9: $\text{temp}_n \leftarrow \sum_{j=1}^{n} MP_{jn}^{(1)} \cdot \text{sign}_j$
10: $\text{temp}_{n+1} \leftarrow \sum_{j=1}^{n+1} MP_{j,n+1}^{(1)} \cdot \text{sign}_\text{j}$
11: $h'_1 \leftarrow \sum_{j=1}^{n+1} \text{temp}_j \cdot \text{sign}_j$
12: **if** $h_1 \neq h'_1$ **then return** FALSE
13: **end if**
14: **for** $k = 2$ to o **do** ▷ polynomials $p^{(2)}, \ldots, p^{(o)}$
15: $\quad \text{temp}_{n+1} \leftarrow \sum_{j=1}^{n+1} MP_{j,n+1}^{(k)} \cdot \text{sign}_j$
16: $\quad$ **for** $i = n$ to $v+1$ by -1 **do**
17: $\quad\quad a_i \leftarrow a_{i-1}$
18: $\quad\quad \text{temp}_i \leftarrow a_i + \sum_{j=v+1}^{i} MP_{ji}^{(k)} \cdot \text{sign}_j$
19: $\quad$ **end for**
20: $\quad$ **for** $i = v$ to 2 by -1 **do**
21: $\quad\quad a_i \leftarrow a_{i-1} + MP_{ii}^{(k)} \cdot \text{sign}_i$
22: $\quad\quad \text{temp}_i \leftarrow a_i$
23: $\quad$ **end for**
24: $\quad a_1 \leftarrow MP_{11}^{(k)} \cdot \text{sign}_1$
25: $\quad \text{temp}_1 \leftarrow a_1$
26: $\quad h'_k \leftarrow \sum_{j=1}^{n+1} \text{temp}_j \cdot \text{sign}_j$
27: $\quad$ **if** $h_k \neq h'_k$ **then return** FALSE
28: $\quad$ **end if**
29: **end for**
30: **return** TRUE

Algorithm 10.2 works as follows. The first matrix vector product $\text{sign} \cdot MP^{(1)} \cdot \text{sign}^T$ is computed in the same way as for random polynomials: From line 2 to line 10 we compute the product $\text{sign} \cdot MP^{(1)}$ (the result is written into the vector temp) and line 11 computes the scalar product of temp and sign. In line 12 we check if the result is equal to the first component of the hash value. If this test is not fulfilled, the algorithm returns FALSE and stops. Furthermore we compute the vector $a = (a_1, \ldots, a_{n-1})$ which we can use during the computation of $\text{sign} \cdot MP^{(2)}$ (line 3). In the loop (line 14 to line 29 of the algorithm) we evaluate the remaining polynomials $(p^{(2)}, \ldots, p^{(o)})$ and test if the result is equal to the corresponding component of the hash value. From line 15 to 25 the algorithm computes the vector $\text{temp} = \text{sign} \cdot MP^{(k)}$. We begin with temp_{n+1} and go back to temp_1. During the computation of temp_i $(i = 2, \ldots, n)$ we use the values a_i computed during the evaluation of the previous polynomial, which helps us to save a large number of multiplications and therefore to speed up the evaluaton process. Furthermore (line 18 and 21) we update the values of a_i $(i = 1, \ldots, n-1)$ for the use in the next iteration of the loop.
Line 26 computes the scalar product of the vectors temp and sign. Finally, in line 27, we check if this result is equal to the corresponding component of the hash value. If the test is not fulfilled, the algorithm returns FALSE and stops.

Computational effort

Algorithm 10.2 needs

- in line 3 $\frac{v\cdot(v+1)}{2} + (o-1)\cdot v$ field multiplications,
- in line 7 $\frac{(o-1)\cdot o}{2}$ field multiplications,
- in line 9 n field multiplications,
- in line 10 $n+1$ field multiplications and
- in line 11 again $n+1$ field multiplications.

Therefore, to compute the value of h'_1, the algorithm needs $\frac{(n+1)\cdot(n+4)}{2}$ field multiplications (just as for the straightforward evaluation of a random polynomial using the alternative approach). In the loop (line 14 to 29) Algorithm 10.2 needs

- in line 15 $n+1$ field multiplications,
- in line 18 $\frac{o\cdot(o+1)}{2}$ field multiplications,
- in line 21 $v-1$ field multiplications,
- in line 24 1 field multiplication and
- in line 26 $n+1$ field multiplications.

So, for every iteration of the loop the algorithm needs $2\cdot(n+1)+v+\frac{o\cdot(o+1)}{2}$ field multiplications. During the verification process of a valid signature, we need therefore

$$(o-1)\cdot\left(2\cdot(n+1)+v+\frac{o\cdot(o+1)}{2}\right)+\frac{(n+1)\cdot(n+4)}{2} \tag{10.10}$$

field multiplications.

For $(\mathbb{F}, o, v) = (\mathrm{GF}(256), 28, 56)$ this means a reduction of the number of field multiplications needed during the verification process by a factor of 5.9 compared to the evaluation of the system with the standard approach. For UOV schemes over GF(16), $(o, v) = (40, 80)$, the reduction factor is 6.1.

The results presented in this section can also be used to speed up the verification process of UOVLRScyc (see Section 8.3). However, the large length of the linear recurring sequence in use prevents us from using the additional structure efficiently.

10.3 UOVLRS2

In the case of UOVLRS2 (see Section 8.4), the matrices $MP^{(k)}$ are of the form shown in Figure 10.2.

We have

$$MP^{(k)}_{i,j} = \gamma_k \cdot MP^{(k)}_{i,j-1} \;\forall i \in \{1,\ldots,v\},\; j \in \{i+1,\ldots,n\},\; k \in \{1,\ldots,o\}. \tag{10.11}$$

Therefore we get

$$(\mathrm{sign}_1,\ldots,\mathrm{sign}_i)\cdot\begin{pmatrix} MP^{(k)}_{1j} \\ MP^{(k)}_{2j} \\ \vdots \\ MP^{(k)}_{ij} \end{pmatrix} = \gamma_k\cdot(\mathrm{sign}_1,\ldots,\mathrm{sign}_i)\cdot\begin{pmatrix} MP^{(k)}_{1,j-1} \\ MP^{(k)}_{2,j-1} \\ \vdots \\ MP^{(k)}_{i,j-1} \end{pmatrix} \tag{10.12}$$

$$\forall i \in \{1,\ldots v\},\; j \in \{i+1,\ldots,n\},\; k \in \{1,\ldots,o\}.$$

$$MP^{(k)} = \begin{pmatrix} \boxed{1} & \boxed{\gamma_k} & \boxed{\gamma_k^2} & \cdots & \boxed{\gamma_k^{v-2}} & \boxed{\gamma_k^{v-1}} & \boxed{\gamma_k^{v}} & \cdots & \boxed{\gamma_k^{n-2}} & \boxed{\gamma_k^{n-1}} & \star \\ 0 & \boxed{\gamma_k^{n}} & \boxed{\gamma_k^{n+1}} & \cdots & \boxed{\gamma_k^{n+v-3}} & \boxed{\gamma_k^{n+v-2}} & \boxed{\gamma_k^{n+v-1}} & \cdots & \boxed{\gamma_k^{2n-3}} & \boxed{\gamma_k^{2n-2}} & \star \\ 0 & 0 & \boxed{\gamma_k^{2n-1}} & \cdots & \boxed{\gamma_k^{2n+v-5}} & \boxed{\gamma_k^{2n+v-4}} & \boxed{\gamma_k^{2n+v-3}} & \cdots & \boxed{\gamma_k^{3n-5}} & \boxed{\gamma_k^{3n-4}} & \star \\ \vdots & & \ddots & & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & & \cdots & 0 & \boxed{\gamma_k^{D-2o-2}} & \boxed{\gamma_k^{D-2o-1}} & \boxed{\gamma_k^{D-2o}} & \cdots & \boxed{\gamma_k^{D-o-4}} & \boxed{\gamma_k^{D-o-3}} & \star \\ 0 & & \cdots & & 0 & \boxed{\gamma_k^{D-o-1}} & \boxed{\gamma_k^{D-o}} & \cdots & \boxed{\gamma_k^{D-2}} & \boxed{\gamma_k^{D-1}} & \star \\ 0 & & \cdots & & & 0 & \star & \cdots & \star & \star & \star \\ \vdots & & & & & & 0 & \ddots & \vdots & \vdots & \vdots \\ \vdots & & & & & & & \ddots & \star & \vdots & \vdots \\ 0 & & \cdots & & \cdots & & \cdots & & 0 & \star & \star \\ 0 & & \cdots & & \cdots & & \cdots & & & 0 & \star \end{pmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} v$$

$$(k \in \{1, \ldots, o\})$$

Figure 10.2: Matrices $MP^{(k)}$ for UOVLRS2

The boxes in Figure 10.2 illustrate this equation: The black boxes show the vector $(MP^{(k)}_{1,j-1}, \ldots, MP^{(k)}_{i,j-1})^T$ on the right hand side of equation (10.12), whereas the blue boxes represent the vector $(MP^{(k)}_{1,j}, \ldots, MP^{(k)}_{i,j})^T$ on the left hand side. Any blue box can be computed by multiplying the corresponding black box by γ_k.

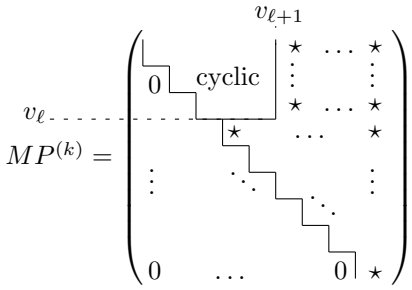
We can use this fact to speed up the verification process of UOVLRS2 by a large factor (see Algorithm 10.3).

Algorithm 10.3 Verification process of UOVLRS2

Input: public key of UOVLRS2 (given in matrix form), signature $\mathbf{z} \in \mathbb{F}^n$, hash value $\mathbf{h} \in \mathbb{F}^m$
Output: boolean value TRUE or FALSE

1: sign $\leftarrow (\mathbf{z}, 1)$
2: **for** $k = 1$ to o **do**
3: $\quad \text{temp}_1 \leftarrow \text{sign}_1$
4: $\quad$ **for** $i = 2$ to v **do**
5: $\quad\quad \text{temp}_i \leftarrow \gamma_k \cdot \text{temp}_{i-1} + MP^{(k)}_{ii} \cdot \text{sign}_i$
6: $\quad$ **end for**
7: $\quad a \leftarrow \text{temp}_v$
8: $\quad$ **for** $i = v + 1$ to n **do**
9: $\quad\quad a \leftarrow \gamma_k \cdot a$
10: $\quad\quad \text{temp}_i \leftarrow a + \sum_{j=v+1}^{i} MP^{(k)}_{ji} \cdot \text{sign}_j$
11: $\quad$ **end for**
12: $\quad \text{temp}_{n+1} \leftarrow \sum_{j=1}^{n+1} MP^{(k)}_{j,n+1} \cdot \text{sign}_j$
13: $\quad h'_k \leftarrow \sum_{j=1}^{n+1} \text{temp}_j \cdot \text{sign}_j$
14: $\quad$ **if** $h_k \neq h'_k$ **then return** FALSE
15: $\quad$ **end if**
16: **end for**
17: **return** TRUE

Algorithm 10.3 works as follows. Every iteration of the main loop (line 2 to 16 of the algorithm) evaluates one polynomial and checks if the result is equal to the corresponding component of the hash value. From line 3 to 13 the public polynomials are evaluated. From line 3 to 12 we hereby compute the matrix vector product $\text{sign} \cdot MP^{(k)}$ whose result is stored in the vector temp. In line



$$v_\ell + 1 \leq k \leq v_{\ell+1} \quad (\ell \in \{1, \ldots, u\})$$

Figure 10.3: Matrices $MP^{(k)}$ for cyclicRainbow

5 and line 9/10 we hereby use the special structure of the UOVLRS2 public key, which allows us save a large number of multiplications and therefore speeds up the computations. Finally, in line 13 of the algorithm, we compute the scalar product of the vectors temp and sign. In line 14 we test, if the result of the evaluation process is equal to the corresponding component of the hash value. If this test is not fulfilled, the algorithm returns FALSE and stops.

Computational effort

Algorithm 10.3 needs (for each iteration of the main loop)

- in the first loop (line 4 to 6) $2 \cdot (v-1)$ field multiplications,
- in the second loop (line 8 to 11) $o + \frac{o \cdot (o+1)}{2}$ field multiplications,
- in line 12 $n+1$ field multiplications and
- in line 13 again $n+1$ field multiplications.

Therefore, to verify a valid signature (o iterations of the main loop), Algorithm 10.3 needs

$$o \cdot \left(3 \cdot n + v + \frac{o \cdot (o+1)}{2} \right) \tag{10.13}$$

field multiplications.

For $\mathbb{F} = \mathrm{GF}(256)$, $(o, v) = (28, 56)$ this means a reduction of the number of field multiplications needed during the verification process by a factor of 5.9 compared to the evaluation of the polynomials using the standard approach. For UOV schemes over GF(16), $(o, v) = (40, 80)$, the reduction factor is 6.2.

10.4 CyclicRainbow

The verification process of cyclicRainbow is mainly done as that of cyclicUOV (see Section 10.2). However we have to consider the different layers of the Rainbow public key. For cyclicRainbow, the matrices $MP^{(k)}$ look as shown in Figure 10.3.
For the polynomials of the ℓ-th Rainbow layer ($\ell \in \{1, \ldots, u\}$) we get

$$MP^{(k)}_{ij} = MP^{(k-1)}_{i,j-1} \ \forall i \in \{1, \ldots, v_\ell\},\ j \in \{i+1, \ldots, v_{\ell+1}\},\ k \in \{v_\ell + 1, \ldots, v_{\ell+1}\} \tag{10.14}$$

or

$$(\text{sign}_1,\ldots,\text{sign}_i)\cdot\begin{pmatrix} MP_{1j}^{(k)} \\ MP_{2j}^{(k)} \\ \vdots \\ MP_{ij}^{(k)} \end{pmatrix} = (\text{sign}_1,\ldots,\text{sign}_i)\cdot\begin{pmatrix} MP_{1,j-1}^{(k-1)} \\ MP_{2,j-1}^{(k-1)} \\ \vdots \\ MP_{i,j-1}^{(k-1)} \end{pmatrix} \quad \begin{array}{c}\forall i \in \{1,\ldots v_\ell\},\ j \in \{i+1,\ldots,v_{\ell+1}\},\\ k \in \{v_\ell+1,\ldots,v_{\ell+1}\}.\end{array} \tag{10.15}$$

We can therefore use Algorithm 10.2 for each Rainbow layer separately (see Algorithm 10.4).

Algorithm 10.4 Verification process of cyclicRainbow

Input: public key of cyclic Rainbow (given in matrix form), signature $\mathbf{z} \in \mathbb{F}^n$, hash value $\mathbf{h} \in \mathbb{F}^m$
Output: boolean value TRUE or FALSE

1: sign $\leftarrow (\mathbf{z}, 1)$
2: **for** $i = 1$ to v_2 **do** ▷ First polynomial ($p^{(v_1+1)}$)
3: $\quad a_i \leftarrow \sum_{j=1}^{\min(i,v_1)} MP_{ji}^{(v_1+1)} \cdot \text{sign}_j$
4: $\quad \text{temp}_i \leftarrow a_i$
5: **end for**
6: **for** $i = v_1 + 1$ to v_2 **do**
7: $\quad \text{temp}_i \leftarrow a_i + \sum_{j=v_1+1}^{i} MP_{ji}^{(v_1+1)} \cdot \text{sign}_j$
8: **end for**
9: **for** $i = v_2 + 1$ to $n + 1$ **do**
10: $\quad \text{temp}_i \leftarrow \sum_{j=1}^{i} MP_{ji}^{(v_1+1)} \cdot \text{sign}_j$
11: **end for**
12: $h'_{v_1+1} \leftarrow \sum_{j=1}^{n+1} \text{temp}_j \cdot \text{sign}_j$
13: **if** $h_{v_1+1} \neq h'_{v_1+1}$ **then return** FALSE
14: **end if**
15: **for** $k = v_1 + 2$ to v_2 **do** ▷ Remaining polynomials of the first layer ($p^{(v_1+2)},\ldots,p^{(v_2)}$)
16: $\quad$ **for** $i = v_2 + 1$ to $n + 1$ **do**
17: $\quad\quad \text{temp}_i \leftarrow \sum_{j=1}^{i} MP_{ji}^{(k)} \cdot \text{sign}_j$
18: $\quad$ **end for**
19: $\quad$ **for** $i = v_2$ to $v_1 + 1$ by -1 **do**
20: $\quad\quad a_i \leftarrow a_{i-1}$
21: $\quad\quad \text{temp}_i \leftarrow a_i + \sum_{j=v_1+1}^{i} MP_{ji}^{(k)} \cdot \text{sign}_j$
22: $\quad$ **end for**
23: $\quad$ **for** $i = v_1$ to 2 by -1 **do**
24: $\quad\quad a_i \leftarrow a_{i-1} + MP_{ii}^{(k)} \cdot \text{sign}_i$
25: $\quad\quad \text{temp}_i \leftarrow a_i$
26: $\quad$ **end for**
27: $\quad a_1 \leftarrow MP_{11}^{(k)} \cdot \text{sign}_1$
28: $\quad \text{temp}_1 \leftarrow a_1$
29: $\quad h'_k \leftarrow \sum_{j=1}^{n+1} \text{temp}_j \cdot \text{sign}_j$
30: $\quad$ **if** $h_k \neq h'_k$ **then return** FALSE
31: $\quad$ **end if**
32: **end for** ▷ algorithm continues on the next page

Algorithm 10.4 works as follows: From line 2 to 14 we evaluate the first polynomial $p^{(v_1+1)}$ and check if the result is equal to the first component of the hash value [1]. Hereby, line 2 to 11 compute the vector temp = sign $\cdot\, MP^{(1)}$ and the elements of the vector a which can be used during the evaluation of the second polynomial $p^{(v_1+2)}$. Line 12 computes the scalar product temp $\cdot$ sign^T and line 13 checks whether the result is equal to the first component of the hash value. If this test is not fulfilled, the algorithm returns FALSE and stops.

[1] Analogously to the public polynomials, we enumerate the components of the hash value by $h_{v_1+1},\ldots,h_n$.

Algorithm 10.4 Verification process of cyclicRainbow (cont.)

for $\ell = 2$ to u **do**
  for $i = v_{\ell+1} + 1$ to $n + 1$ **do** ▷ First polynomial of the ℓ-th layer ($p^{(v_\ell+1)}$)
    $\text{temp}_i \leftarrow \sum_{j=1}^{i} MP_{j,i}^{(v_\ell+1)} \cdot \text{sign}_j$
  end for
  for $i = v_{\ell+1}$ to $v_\ell + 1$ by -1 **do**
    $a_i \leftarrow \sum_{j=1}^{v_\ell} MP_{ji}^{(v_\ell+1)} \cdot \text{sign}_j$
    $\text{temp}_i \leftarrow a_i + \sum_{j=v_\ell+1}^{i} MP_{ji}^{(v_\ell+1)} \cdot \text{sign}_j$
  end for
  for $i = v_\ell$ to $v_{\ell-1} + 1$ by -1 **do**
    $a_i \leftarrow a_{i-1} + \sum_{j=v_{\ell-1}+1}^{i} MP_{ji}^{(v_\ell+1)} \cdot \text{sign}_j$
    $\text{temp}_i \leftarrow a_i$
  end for
  for $i = v_{\ell-1}$ to 2 by -1 **do**
    $a_i \leftarrow a_{i-1} + MP_{ii}^{(v_\ell+1)} \cdot \text{sign}_i$
    $\text{temp}_i \leftarrow a_i$
  end for
  $a_1 \leftarrow MP_{11}^{(v_\ell+1)} \cdot \text{sign}_1$
  $\text{temp}_1 \leftarrow a_1$
  $h'_{v_\ell+1} \leftarrow \sum_{j=1}^{n+1} \text{temp}_j \cdot \text{sign}_j$
  if $h_{v_\ell+1} \neq h'_{v_\ell+1}$ **then return** FALSE
  end if
  for $k = v_\ell + 2$ to $v_{\ell+1}$ **do** ▷ Remaining Polynomials of the ℓ-th layer ($p^{(v_\ell+2)}, \ldots, p^{(v_{\ell+1})}$)
    for $i = v_{\ell+1} + 1$ to $n + 1$ **do**
      $\text{temp}_i \leftarrow \sum_{j=1}^{i} MP_{ji}^{(k)} \cdot \text{sign}_j$
    end for
    for $i = v_{\ell+1}$ to $v_l + 1$ by -1 **do**
      $a_i \leftarrow a_{i-1}$
      $\text{temp}_i \leftarrow a_i + \sum_{j=v_\ell+1}^{i} MP_{ji}^{(k)} \cdot \text{sign}_j$
    end for
    for $i = v_\ell$ to 2 by -1 **do**
      $a_i \leftarrow a_{i-1} + MP_{ii}^{(k)} \cdot \text{sign}_i$
      $\text{temp}_i \leftarrow a_i$
    end for
    $a_1 \leftarrow MP_{11}^{(k)} \cdot \text{sign}_1$
    $\text{temp}_1 \leftarrow a_1$
    $h'_k \leftarrow \sum_{j=1}^{n+1} \text{temp}_j \cdot \text{sign}_j$
    if $h_k \neq h'_k$ **then return** FALSE
    end if
  end for
end for
return TRUE

From line 15 to 32 we deal with the remaining polynomials of the first Rainbow layer ($p^{(v_1+2)}, \ldots, p^{(v_2)}$). From line 16 to 29 the polynomials are evaluated. First (line 16 to 28) we compute again the vector temp $=$ sign $\cdot\, MP^{(k)}$. During this step we can use the vector a to speed up the computations (line 21 and 25). Furthermore, a is updated for use in the next iteration of the loop (line 20, 24 and 27). Finally, line 29 computes the scalar product temp $\cdot$ sign^T and line 30 checks if the result is equal to the corresponding component of the hash value. If this test is not fulfilled, the algorithm returns FALSE and stops.
In each iteration of the big loop (line 33 to 72) we evaluate all the polynomials of the ℓ-th Rainbow layer and test, if the result is equal to the corresponding components of the hash value. Lines 34 to 53 hereby deal with the first polynomial of the ℓ-th layer ($p^{(v_\ell+1)}$). Besides evaluating the polynomial, we invest much work in updating the vector a (line 38, 41, 45 and 48) which helps us during the evaluation of the remaining polynomials. In line 52 we test if the result of the evaluation is equal to the corresponding component of the hash value. If this test is not fulfilled, the algorithm returns FALSE and stops.
From line 54 to 71 we finally deal with the remaining polynomials of the ℓ-th layer ($p^{(v_\ell+2)}, \ldots, p^{(v_{\ell+1})}$). The computations are made less complex by the use of the vector a (line 60 and 63) which helps us to speed up the algorithm. For increasing ℓ more and more of the multiplications can be saved. Finally, in line 69 of the algorithm, we test if the result of the evaluation is equal to the corresponding component of the hash value. As soon as this test is not fulfilled, the algorithm returns FALSE and stops.

Example 10.1. *In Example 9.1 we had* $\mathbb{F} = GF(19)$, $(v_1, o_1, o_2) = (2, 2, 2)$ *and the matrices* $MP^{(k)}$ $\{3 \leq k \leq 6\}$ *were given by*

$$MP^{(3)} = \begin{pmatrix} 1 & 2 & 3 & 4 & 10 & 3 & 18 \\ 0 & 7 & 8 & 9 & 15 & 14 & 13 \\ 0 & 0 & 11 & 5 & 9 & 15 & 5 \\ 0 & 0 & 0 & 7 & 12 & 0 & 2 \\ 0 & 0 & 0 & 0 & 6 & 8 & 14 \\ 0 & 0 & 0 & 0 & 0 & 12 & 18 \\ 0 & 0 & 0 & 0 & 0 & 0 & 16 \end{pmatrix},$$

$$MP^{(4)} = \begin{pmatrix} 18 & 1 & 2 & 3 & 14 & 9 & 9 \\ 0 & 6 & 7 & 8 & 17 & 18 & 12 \\ 0 & 0 & 7 & 12 & 14 & 8 & 5 \\ 0 & 0 & 0 & 5 & 8 & 0 & 16 \\ 0 & 0 & 0 & 0 & 7 & 4 & 17 \\ 0 & 0 & 0 & 0 & 0 & 6 & 13 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{pmatrix},$$

$$MP^{(5)} = \begin{pmatrix} 17 & 18 & 1 & 2 & 3 & 4 & 17 \\ 0 & 5 & 6 & 7 & 8 & 9 & 1 \\ 0 & 0 & 10 & 11 & 12 & 13 & 11 \\ 0 & 0 & 0 & 14 & 15 & 16 & 8 \\ 0 & 0 & 0 & 0 & 15 & 3 & 12 \\ 0 & 0 & 0 & 0 & 0 & 15 & 16 \\ 0 & 0 & 0 & 0 & 0 & 0 & 8 \end{pmatrix} \text{ and}$$

$$MP^{(6)} = \begin{pmatrix} 16 & 17 & 18 & 1 & 2 & 3 & 0 \\ 0 & 4 & 5 & 6 & 7 & 8 & 1 \\ 0 & 0 & 9 & 10 & 11 & 12 & 3 \\ 0 & 0 & 0 & 13 & 14 & 15 & 5 \\ 0 & 0 & 0 & 0 & 4 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 15 & 8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 11 \end{pmatrix}.$$

Let $\mathbf{z} = (2, 4, 3, 8, 16, 5) \in \mathbb{F}^6$ *be a (valid or invalid) signature,* $\text{sign} = (2, 4, 3, 8, 16, 5, 1)$ *be the corresponding extended signature vector and* $\mathbf{h} = (12, 12, 18, 16) \in \mathbb{F}^4$ *be the hash value of the message to be signed.*

1. *Evaluation of the first polynomial* ($p^{(3)}$)
   $\text{temp}_1 = MP_{11}^{(3)} \cdot \text{sign}_1 = 1 \cdot 2 = 2 = a_1$
   $\text{temp}_2 = MP_{12}^{(3)} \cdot \text{sign}_1 + MP_{22}^{(3)} \cdot \text{sign}_2 = 2 \cdot 2 + 7 \cdot 4 = 13 = a_2$
   $\text{temp}_3 = \underbrace{MP_{13}^{(3)} \cdot \text{sign}_1 + MP_{23}^{(3)} \cdot \text{sign}_2}_{a_3 = 3 \cdot 2 + 8 \cdot 4 = 0} + MP_{33}^{3} \cdot \text{sign}_3 = 0 + 11 \cdot 3 = 14$
   $\text{temp}_4 = \underbrace{MP_{14}^{(3)} \cdot \text{sign}_1 + MP_{24}^{(3)} \cdot \text{sign}_2}_{a_4 = 4 \cdot 2 + 9 \cdot 4 = 6} + MP_{34}^{3} \cdot \text{sign}_3 + MP_{44}^{(3)} \cdot \text{sign}_4 = 6 + 5 \cdot 3 + 7 \cdot 8 = 1$
   $\text{temp}_5 = \sum_{i=1}^{5} MP_{i5}^{(3)} \cdot \text{sign}_i = 14$
   $\text{temp}_6 = \sum_{i=1}^{6} MP_{i6}^{(3)} \cdot \text{sign}_i = 10$
   $\text{temp}_7 = \sum_{i=1}^{7} MP_{i7}^{(3)} \cdot \text{sign}_i = 12$
   $h'_3 = \sum_{i=1}^{7} \text{temp}_i \cdot \text{sign}_i = 12.$
   Since we have $h'_3 = 12 = h_3$*, we continue.*

2. *Evaluation of the second polynomial* ($p^{(4)}$)
   $\text{temp}_7 = \sum_{i=1}^{7} MP_{i7}^{(4)} \cdot \text{sign}_i = 16$
   $\text{temp}_6 = \sum_{i=1}^{6} MP_{i6}^{(4)} \cdot \text{sign}_i = 18$
   $\text{temp}_5 = \sum_{i=1}^{5} MP_{i5}^{(4)} \cdot \text{sign}_i = 10$
   $\text{temp}_4 = \sum_{i=1}^{4} MP_{i4}^{(4)} \cdot \text{sign}_i = \underbrace{a_3}_{a_4 = 0} + MP_{34}^{(4)} \cdot \text{sign}_3 + MP_{44}^{(4)} \cdot \text{sign}_4 = 0 + 12 \cdot 3 + 5 \cdot 8 = 0$
   $\text{temp}_3 = \sum_{i=1}^{3} MP_{i3}^{(4)} \cdot \text{sign}_i = \underbrace{a_2}_{a_3 = 13} + MP_{33}^{(4)} \cdot \text{sign}_3 = 13 + 7 \cdot 3 = 15$
   $\text{temp}_2 = \sum_{i=1}^{2} MP_{i2}^{(4)} \cdot \text{sign}_i = a_1 + MP_{22}^{(4)} \cdot \text{sign}_2 = 2 + 6 \cdot 4 = 7 = a_2$
   $\text{temp}_1 = MP_{11}^{(4)} \cdot \text{sign}_1 = 18 \cdot 2 = 17 = a_1$
   $h'_4 = \sum_{i=1}^{7} \text{temp}_i \cdot \text{sign}_i = 12.$
   Since we have $h'_4 = 12 = h_4$*, we continue.*

3. *Evaluation of the third polynomial* ($p^{(5)}$)
   $\text{temp}_7 = \sum_{i=1}^{7} MP_{i7}^{(5)} \cdot \text{sign}_i = 16$
   $\text{temp}_6 = \sum_{i=1}^{6} MP_{i6}^{(5)} \cdot \text{sign}_i = 11$
   $\text{temp}_5 = \underbrace{a_4 + MP_{35}^{(5)} \cdot \text{sign}_3 + MP_{45}^{(5)} \cdot \text{sign}_4}_{a_5 = 4} + MP_{55}^{(5)} \cdot \text{sign}_5 = 16$
   $\text{temp}_4 = a_3 + MP_{34}^{(5)} \cdot \text{sign}_3 + MP_{44}^{(5)} \cdot \text{sign}_4 = 6 = a_4$
   $\text{temp}_3 = a_2 + MP_{33}^{(5)} \cdot \text{sign}_3 = 18 = a_3$
   $\text{temp}_2 = a_1 + MP_{22}^{(5)} \cdot \text{sign}_2 = 18 = a_2$
   $\text{temp}_1 = MP_{11}^{(5)} \cdot \text{sign}_1 = 15 = a_1$
   $h'_5 = \sum_{i=1}^{7} \text{temp}_i \cdot \text{sign}_i = 18.$
   Since we have $h'_5 = 18 = h_5$*, we continue.*

4. *Evaluation of the fourth polynomial* ($p^{(6)}$)
   $\text{temp}_7 = \sum_{i=1}^{7} MP_{i7}^{(6)} \cdot \text{sign}_i = 6$
   $\text{temp}_6 = a_5 + MP_{56}^{(6)} \cdot \text{sign}_5 + MP_{66}^{(6)} \cdot \text{sign}_6 = 16$
   $\text{temp}_5 = \underbrace{a_4}_{a_5 = 6} + MP_{55}^{(6)} \cdot \text{sign}_6 = 13$
   $\text{temp}_4 = a_3 + MP_{44}^{(6)} \cdot \text{sign}_4 = 8 = a_4$
   $\text{temp}_3 = a_2 + MP_{33}^{(6)} \cdot \text{sign}_3 = 7 = a_3$

$\text{temp}_2 = a_1 + MP_{22}^{(6)} \cdot \text{sign}_2 = 12 = a_2$
$\text{temp}_1 = MP_{11}^{(6)} \cdot \text{sign}_1 = 16 \cdot 2 = 13 = a_1$
$h'_6 = \sum_{i=1}^{7} \text{temp}_i \cdot \text{sign}_i = 16.$
Since we have $h'_6 = 16 = h_6$*, the signature is accepted.* □

Computational cost

Our algorithm needs

- $\frac{(n+1)\cdot(n+4)}{2}$ field multiplications to evaluate the first polynomial $p^{(v_1+1)}$ (line 2 - 12) and
- $(o_1 - 1) \cdot \left(\frac{(n+1)\cdot(n+4)}{2} - \frac{v_2\cdot(v_2+1)}{2} + \frac{o_1\cdot(o_1+1)}{2} + v_1 \right)$ field multiplications to evaluate the polynomials $p^{(v_1+2)}, \ldots, p^{(v_2)}$ (line 15 - 32).

To evaluate the polynomials of the ℓ-th layer the algorithm needs

- $\frac{(n+1)\cdot(n+4)}{2} - \frac{v_\ell\cdot(v_\ell+1)}{2} + \frac{o_{\ell-1}\cdot(o_{\ell-1}+1)}{2} + v_{\ell-1}$ field multiplications to evaluate the polynomial $p^{(v_{\ell-1}+1)}$ (line 34 - 51) and
- $(o_\ell - 1) \cdot \left(\frac{(n+1)\cdot(n+4)}{2} - \frac{v_{\ell+1}\cdot(v_{\ell+1}+1)}{2} + \frac{o_\ell\cdot(o_\ell+1)}{2} + v_\ell \right)$ field multiplications to evaluate the polynomials $p^{(v_{\ell-1}+2)}, \ldots, p^{(v_\ell)}$ (line 54 - 71).

So, to verify a valid signature, Algorithm 10.4 needs

$$\begin{aligned} m \quad \cdot \quad & \frac{(n+1)\cdot(n+4)}{2} - (o_1 - 1) \cdot \frac{v_2\cdot(v_2+1) - o_1\cdot(o_1+1) - 2\cdot v_1}{2} \\ - \quad & \sum_{\ell=2}^{u} \left(\frac{v_\ell\cdot(v_\ell+1) - o_{\ell-1}\cdot(o_{\ell-1}+1) - 2\cdot v_{\ell-1}}{2} \right. \\ + \quad & \left. (o_\ell - 1) \cdot \frac{v_{\ell+1}\cdot(v_{\ell+1}+1) - o_\ell\cdot(o_\ell+1) - 2\cdot v_\ell}{2} \right) \end{aligned} \tag{10.16}$$

field multiplications. For the parameters $(\mathbb{F}, v_1, o_1, o_2) = (\text{GF}(256),17,13,13)$, this means a reduction of the number of field multiplications by a factor of 2.5 compared to evaluating the Rainbow public key using the standard approach. For Rainbow schemes over GF(16), $(v_1, o_1, o_2) = (17, 23, 17)$, the reduction factor is 2.4.

10.5 RainbowLRS2

The verification process of RainbowLRS2 is mainly done as that of UOVLRS2 (see Section 10.3). However we have to consider the different layers of the Rainbow public key. For RainbowLRS2, the matrices $MP^{(k)}$ look as shown in Figure 10.4.
For the polynomials of the ℓ-th Rainbow layer ($\ell \in \{1, \ldots, u\}$) we have

$$MP_{ij}^{(k)} = \gamma_k \cdot MP_{i,j-1}^{(k)} \ \forall i \in \{1, \ldots, v_\ell\},\ j \in \{i+1, \ldots, v_{\ell+1}\},\ k \in \{v_\ell + 1, \ldots, v_{\ell+1}\}. \tag{10.17}$$

Such we get

$$(\text{sign}_1, \ldots, \text{sign}_i) \cdot \begin{pmatrix} MP_{1j}^{(k)} \\ \vdots \\ MP_{ij}^{(k)} \end{pmatrix} = \gamma_k \cdot (\text{sign}_1, \ldots, \text{sign}_i) \cdot \begin{pmatrix} MP_{1,j-1}^{(k)} \\ \vdots \\ MP_{i,j-1}^{(k)} \end{pmatrix} \quad \begin{array}{c} \forall i \in \{1, \ldots, v_\ell\},\ j \in \{i+1, \ldots, v_{\ell+1}\}, \\ k \in \{v_\ell + 1, \ldots, v_{\ell+1}\} \end{array} . \tag{10.18}$$

We can therefore use Algorithm 10.3 for each Rainbow layer separately (see Algorithm 10.5).

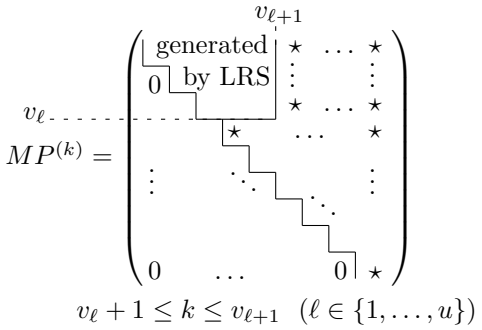


Figure 10.4: Matrices $MP^{(k)}$ for RainbowLRS

Algorithm 10.5 Verification process of RainbowLRS2

Input: public key of RainbowLRS2 (given in matrix form), signature $\mathbf{z} \in \mathbb{F}^n$, hash value $\mathbf{h} \in \mathbb{F}^m$
Output: boolean value TRUE or FALSE

sign $\leftarrow (\mathbf{z}, 1)$
for $\ell = 1$ to u **do**
  for $k = v_\ell + 1$ to $v_{\ell+1}$ **do**
    $\text{temp}_1 \leftarrow \text{sign}_1$
    for $i = 2$ to v_ℓ **do**
      $\text{temp}_i \leftarrow \gamma_k \cdot \text{temp}_{i-1} + MP^{(k)}_{ii} \cdot \text{sign}_i$
    end for
    $a \leftarrow \text{temp}_{v_\ell}$
    for $i = v_\ell + 1$ to $v_{\ell+1}$ **do**
      $a \leftarrow \gamma_k \cdot a$
      $\text{temp}_i \leftarrow a + \sum_{j=v_\ell+1}^{i} MP^{(k)}_{ji} \cdot \text{sign}_j$
    end for
    for $i = v_{\ell+1} + 1$ to $n + 1$ **do**
      $\text{temp}_i \leftarrow \sum_{j=1}^{i} MP^{(k)}_{ji} \cdot \text{sign}_j$
    end for
    $h'_k \leftarrow \sum_{j=1}^{n+1} \text{temp}_j \cdot \text{sign}_j$
    if $h_k \neq h'_k$ **then return** FALSE
    end if
  end for
end for
return TRUE

Algorithm 10.5 works as follows: Each iteration of the loop (line 3 - 19) evaluates one of the public polynomials and checks if the result is equal to the corresponding component of the hash value. From line 3 to 15 we compute the vector temp = sign $\cdot MP^{(k)}$. Due to the special structure of the RainbowLRS2 public key we can save a large number of multiplications and therefore speed up the computations. Line 16 computes the scalar product temp $\cdot$ sign^T and in line 17 we check if the result is equal to the corresponding component of the hash value. If this test is not fulfilled, the algorithm returns FALSE and stops.

Computational Effort

To evaluate a polynomial of the ℓ-th layer, say $p^{(k)}$ ($k \in \{v_\ell + 1, \ldots, v_{\ell+1}\}$), Algorithm 10.5 needs

- $2 \cdot (v_\ell - 1)$ multiplications to compute the elements $\text{temp}_2, \ldots, \text{temp}_{v_\ell}$ (line 6 of the algorithm),
- o_ℓ multiplications to update a (step 10),
- $\frac{o_\ell \cdot (o_\ell + 1)}{2}$ multiplications to compute $\text{temp}_{v_l+1}, \ldots, \text{temp}_{v_{\ell+1}}$ (line 11),
- $\frac{(n+1)\cdot(n+2)}{2} - \frac{v_{\ell+1}\cdot(v_{\ell+1}+1)}{2}$ multiplications to compute $\text{temp}_{v_{\ell+1}+1}, \ldots, \text{temp}_{n+1}$ (line 14)
- and $n + 1$ multiplications to compute the value h_k (line 16).

Therefore, during the verification of a valid signature, Algorithm 10.5 needs

$$\sum_{\ell=1}^{u} o_\ell \cdot \left(\frac{(n+1)\cdot(n+2) - v_{\ell+1}\cdot(v_{\ell+1}+1) + o_\ell \cdot (o_\ell + 1)}{2} + o_\ell + 2 \cdot v_\ell + n - 1 \right) \tag{10.19}$$

field multiplications. For the parameters $(\mathbb{F}, v_1, o_1, o_2)$ = (GF(256),17,13,13) this means a reduction of the number of field multiplications by a factor of 2.5 compared to the evaluation of a Rainbow public key using the standard approach. For Rainbow schemes over GF(16), $(v_1, o_1, o_2) = (17, 23, 17)$, the reduction factor is 2.4.

10.6 Experimental Results

In this section we present our experimental results regarding the verification process of our improved versions of UOV and Rainbow. The schemes were implemented in C and run on a Lenovo ThinkPad with one Intel Core 2Duo processor with 2.53 GHz and 4 GB RAM.
Tables 10.2 and 10.3 show the running time of the verification process of our improved schemes as well as for the standard UOV and Rainbow scheme for different underlying fields and different levels of security. For each of the parameter sets listed in the tables we verified 100,000 valid signatures. For the verification of the standard UOV and Rainbow schemes we use the standard approach.
As the tables show, we get quite significant speed up factors of 5.8 and 2.1 for UOV and Rainbow respectively (80 bit security). For higher security levels the speed up factors further increase up to values of 6.1 and 2.4 respectively. Again, note that the schemes UOVLRS2 and RainbbowLRS2 can not be used over the fields GF(16) and GF(31).

security (bit)			verification time (ms)	CPU cycles (10^6)	speed up factor
80	GF(16)	UOV(40,80)	2.76	6.97	-
		cyclicUOV(40,80)	0.47	1.19	5.8
	GF(31)	UOV(33,66)	2.01	5.08	-
		cyclicUOV(33,66)	0.36	0.90	5.6
	GF(256)	UOV(28,56)	1.13	2.87	-
		cyclicUOV(28,56)	0.23	0.58	4.9
		UOVLRS2(28,56)	0.20	0.50	5.7
100	GF(256)	UOV(35,70)	2.11	5.34	-
		cyclicUOV(35,70)	0.40	1.00	5.4
		UOVLRS2(35,70)	0.36	0.90	5.8
128	GF(256)	UOV(45,90)	4.42	11.24	-
		cyclicUOV(45,90)	0.80	2.01	5.5
		UOVLRS2(45,90)	0.72	1.82	6.1

Table 10.2: Running time of the verification process for UOV like schemes

security (bit)			verification time (ms)	CPU cycles (10^6)	speed up factor
80	GF(16)	Rainbow(17,23,17)	0.65	1.63	-
		cyclicRainbow(17,23,17)	0.32	0.82	2.0
	GF(31)	Rainbow(14,19,14)	0.49	1.23	-
		cyclicRainbow(14,19,14)	0.23	0.58	2.1
	GF(256)	Rainbow(17,13,13)	0.29	0.72	-
		cyclicRainbow(17,13,13)	0.14	0.35	2.1
		RainbowLRS2(17,13,13)	0.15	0.38	1.9
100	GF(256)	Rainbow(26,16,17)	0.66	1.67	-
		cyclicRainbow(26,16,17)	0.29	0.72	2.3
		RainbowLRS2(26,16,17)	0.31	0.78	2.1
128	GF(256)	Rainbow(36,28,15)	1.50	3.78	-
		cyclicRainbow(36,21,22)	0.63	1.59	2.4
		RainbowLRS2(36,21,22)	0.67	1.69	2.2

Table 10.3: Running time of the verification process for Rainbow like schemes

Chapter 11

Speeding up QUAD

In this chapter we show how the techniques presented in the previous chapter can be used to speed up the multivariate stream cipher QUAD without weakening its security. In particular, we propose two variants of QUAD called cyclicQUAD and QUADLRS. By using systems of structured polynomials, we can speed up the keystream generation process of QUAD by a factor of up to 6.8. In this chapter we define $D = \frac{(n+1)\cdot(n+2)}{2}$.

11.1 The QUAD Stream Cipher

QUAD is a provable secure multivariate based stream cipher introduced in 2006 by Berbain, Gilbert and Patarin [2]. The security of QUAD is based solely on the MQ-Problem of solving nonlinear polynomial systems over a finite field.

Like all stream ciphers, QUAD encrypts a message by producing a keystream ks which has the same length as the message. The ciphertext c of the message m is then created by simply bitwise XORing of the message and the keystream, i.e.

$$c_i = m_i \oplus ks_i \quad \forall i = 0, \ldots, \mathrm{Len}(m) - 1.$$

A ciphertext c is decrypted in the same way, namely

$$m_i = c_i \oplus ks_i \quad \forall i = 0, \ldots, \mathrm{Len}(c) - 1.$$

The keystream of QUAD is generated as follows. Let $\mathbb{F}$ be a finite field. One chooses 4 multivariate quadratic systems $\mathcal{P}$, $\mathcal{Q}$, $\mathcal{S}_0$ and $\mathcal{S}_1 : \mathbb{F}^n \to \mathbb{F}^n$ [1]. These four systems are considered as system parameters and are fixed for a large number of users. Before encrypting a message, a user chooses a key $\mathbf{k} \in \mathbb{F}^n$ and an initial vector $\mathrm{IV} \in \{0,1\}^{80}$. The keystream of QUAD is then generated by following Algorithms 11.1 and 11.2. Figure 11.1 shows a graphical illustration of the keystream generation process. In Algorithm 11.2 we set $L = \lceil \frac{\mathrm{Len}(m)}{\lg(q)\cdot n} \rceil$, where q is the cardinality of the underlying field.
If the length of the so obtained keystream is larger than the length of the message m, we discard the last bits of ks.

[1] While for the functioning of QUAD the three systems $\mathcal{P}$, $\mathcal{S}_0$ and $\mathcal{S}_1$ are required to be determined, we do not need this property for the system $\mathcal{Q}$. In particular, the system $\mathcal{Q}$ could be overdetermined ($m > n$), which would make the keystream generation process more efficient. But, since overdetermined systems are easier to solve, we choose, for reasons of security, $\mathcal{Q}$ to be a determined system, too ($m = n$).

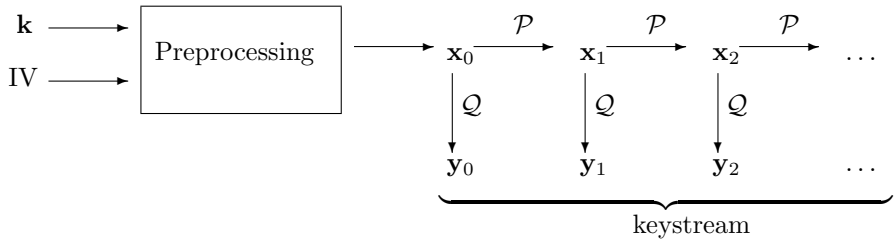


Figure 11.1: Keystream generation of QUAD

Algorithm 11.1 Preprocessing Phase of QUAD

Input: multivariate systems $\mathcal{P}$, $\mathcal{S}_0$, $\mathcal{S}_1 : \mathbb{F}^n \to \mathbb{F}^n$, key $\mathbf{k} \in \mathbb{F}^n$, initial vector IV $\in \{0,1\}^{80}$
Output: internal State IS $\in \mathbb{F}^n$

IS $\leftarrow \mathbf{k}$
for $i = 0$ to 79 **do**
    if IV[i]=1 **then**
        IS $\leftarrow \mathcal{S}_1$(IS)
    else
        IS $\leftarrow \mathcal{S}_0$(IS)
    end if
end for
for $i = 0$ to 79 **do**
    IS $\leftarrow \mathcal{P}$(IS)
end for
return IS

Algorithm 11.2 Keystream generation of QUAD

Input: multivariate systems $\mathcal{P}$, $\mathcal{Q} : \mathbb{F}^n \to \mathbb{F}^n$, internal state IS $\in \mathbb{F}^n$, length L
Output: keystream $ks \in \mathbb{F}^{n \cdot L}$

$ks \leftarrow [\,]$
for $i = 0$ to $L - 1$ **do**
    $ks \leftarrow ks \| \mathcal{Q}$(IS)
    IS $\leftarrow \mathcal{P}$(IS)
end for
return ks

11.2 Evaluation of Polynomials

The most expensive part during the keystream generation of QUAD is the evaluation of the polynomial systems $\mathcal{P}$ and $\mathcal{Q}$ (see Algorithm 11.2). As mentioned in the previous chapter, there are basically two ways to perform this step. For the following we define the standard approach and the alternative approach as in Chapter 10. We created two different implementations of the QUAD stream cipher using the two different evaluation strategies and tested their performance. As the Tables 11.2 and 11.3 show, for random systems, evaluating the polynomials using the standard approach is usually more effective. But, for structured systems, we can do much better with the alternative approach.

11.2.1 CyclicQUAD

In this subsection we describe how to speed up the QUAD stream cipher by using partially circulant polynomials. The result is a variant of QUAD called cyclicQUAD. The multivariate systems $\mathcal{P}$, $\mathcal{Q}$, $\mathcal{S}_0$ and $\mathcal{S}_1$ of cyclicQUAD are generated as shown in Algorithm 11.3. To get the system parameter of cyclicQUAD, we run Algorithm 11.3 four times (for each $\mathcal{P}$, $\mathcal{Q}$, $\mathcal{S}_0$ and $\mathcal{S}_1$).

Algorithm 11.3 Generation of a partially circulant system of polynomials

Input: parameter n
Output: partially circulant system $\mathcal{P} : \mathbb{F}^n \to \mathbb{F}^n$ (given by its Macauley matrix $M_P \in \mathbb{F}^{n \times D}$)
1: Choose randomly a vector $\mathbf{b} \in \mathbb{F}^D$.
2: $M_P[1] \leftarrow \mathbf{b}$, where $M_P[i]$ denotes the i-th row of the Macauley matrix M_P.
3: **for** $k = 2$ to n **do**
4: $\quad \mathbf{b} \leftarrow \mathcal{R}(\mathbf{b})$, where $\mathcal{R}(\mathbf{b})$ is the cyclic right shift of the vector $\mathbf{b}$ by one position.
5: $\quad M_P[k] \leftarrow \mathbf{b}$
6: **end for**
7: **return** M_P

Let $\mathcal{P}$ be a multivariate quadratic system as generated by Algorithm 11.3. Then the corresponding matrices $MP^{(k)}$ $(k = 1, \ldots, n)$ (see equation (2.10)) look as shown in Figure 11.2. We have

$$MP_{ij}^{(k)} = MP_{i,j-1}^{(k-1)} \; i \in \{1, \ldots, n\}, \; j \in \{i+1, \ldots, n+1\}, \; k \in \{2, \ldots, n\}. \tag{11.1}$$

Therefore we get

$$(x_1, \ldots, x_i) \cdot \begin{pmatrix} MP_{1j}^{(k)} \\ MP_{2j}^{(k)} \\ \vdots \\ MP_{ij}^{(k)} \end{pmatrix} = (x_1, \ldots, x_i) \cdot \begin{pmatrix} MP_{1,j-1}^{(k-1)} \\ MP_{2,j-1}^{(k-1)} \\ \vdots \\ MP_{i,j-1}^{(k-1)} \end{pmatrix} \; \forall i \in \{1, \ldots, n\}, \; j \in \{i+1, \ldots, n+1\}, \; k \in \{2, \ldots, n\}. \tag{11.2}$$

Figure 11.2 illustrates this equation. The black boxes show the vector $(MP_{1,j-1}^{(k-1)}, \ldots, MP_{i,j-1}^{(k-1)})^T$ on the right hand side of the equation, whereas the blue boxes represent the vector $(MP_{1j}^{(k)}, \ldots, MP_{ij}^{(k)})^T$ on the left hand side. Note that the blue boxes in the matrix $MP^{(k)}$ are exactly the same as the black boxes in the matrix $MP^{(k-1)}$ $(k = 2, \ldots, n)$.
Algorithm 11.4 uses this fact to speed up the evaluation process of a partially circulant system $\mathcal{P}$ by a large factor.

Algorithm 11.4 works as follows. The first polynomial $p^{(1)}$ is evaluated just as a random polynomial with the alternative approach. In the loop (line 2 to 4) we compute the vector $\text{temp} = \hat{\mathbf{x}} \cdot MP^{(1)}$.

$$MP^{(1)} = \begin{pmatrix} b_1 & b_2 & b_3 & \dots & b_{n-1} & b_n & b_{n+1} \\ 0 & b_{n+2} & b_{n+3} & \dots & b_{2n-1} & b_{2n} & b_{2n+1} \\ 0 & 0 & b_{2n+2} & \dots & b_{3n-2} & b_{3n-1} & b_{3n} \\ \vdots & & \ddots & \ddots & \vdots & & \vdots \\ 0 & & \dots & 0 & b_{D-5} & b_{D-4} & b_{D-3} \\ 0 & & \dots & & 0 & b_{D-2} & b_{D-1} \\ 0 & & \dots & & \dots & 0 & b_D \end{pmatrix}$$

$$MP^{(2)} = \begin{pmatrix} b_D & b_1 & b_2 & \dots & b_{n-2} & b_{n-1} & b_n \\ 0 & b_{n+1} & b_{n+2} & \dots & b_{2n-2} & b_{2n-1} & b_{2n} \\ 0 & 0 & b_{2n+1} & \dots & b_{3n-3} & b_{3n-2} & b_{3n-1} \\ \vdots & & \ddots & \ddots & \vdots & & \vdots \\ 0 & & \dots & 0 & b_{D-6} & b_{D-5} & b_{D-4} \\ 0 & & \dots & & 0 & b_{D-3} & b_{D-2} \\ 0 & & \dots & & \dots & 0 & b_{D-1} \end{pmatrix}$$

$$\vdots$$

$$MP^{(n-1)} = \begin{pmatrix} b_{D-n+2} & b_{D-n+3} & b_{D-n+4} & \dots & b_D & b_1 & b_2 \\ 0 & b_3 & b_4 & \dots & b_n & b_{n+1} & b_{n+2} \\ 0 & 0 & b_{n+3} & \dots & b_{2n-5} & b_{2n-4} & b_{2n-3} \\ \vdots & & \ddots & \ddots & \vdots & & \vdots \\ 0 & & \dots & 0 & b_{D-n-4} & b_{D-n-3} & b_{D-n-2} \\ 0 & & \dots & & 0 & b_{D-n-1} & b_{D-n} \\ 0 & & \dots & & \dots & 0 & b_{D-n+1} \end{pmatrix}$$

$$MP^{(n)} = \begin{pmatrix} b_{D-n+1} & b_{D-n+2} & b_{D-n+3} & \dots & b_{D-1} & b_D & b_1 \\ 0 & b_2 & b_3 & \dots & b_{n-1} & b_n & b_{n+1} \\ 0 & 0 & b_{n+2} & \dots & b_{2n-6} & b_{2n-5} & b_{2n-4} \\ \vdots & & \ddots & \ddots & \vdots & & \vdots \\ 0 & & \dots & 0 & b_{D-n-5} & b_{D-n-4} & b_{D-n-3} \\ 0 & & \dots & & 0 & b_{D-n-2} & b_{D-n-1} \\ 0 & & \dots & & \dots & 0 & b_{D-n} \end{pmatrix}$$

Figure 11.2: Matrices $MP^{(k)}$ for cyclicQUAD

Algorithm 11.4 Evaluation of partially circulant polynomials

Input: partially circulant system $\mathcal{P} : \mathbb{F}^n \to \mathbb{F}^n$ (given in matrix form), internal state $\text{IS} \in \mathbb{F}^n$
Output: $\text{res} = \mathcal{P}(\text{IS})$

1: $\hat{\mathbf{x}} \leftarrow (\text{IS}_1, \ldots, \text{IS}_n, 1)$
2: **for** $i = 1$ to $n+1$ **do** ▷ first polynomial ($p^{(1)}$)
3: $\quad \text{temp}_i \leftarrow \sum_{j=1}^{i} MP_{ji}^{(1)} \cdot \hat{\mathbf{x}}_j$
4: **end for**
5: $\text{res}_1 \leftarrow \sum_{j=1}^{n+1} \text{temp}_j \cdot \hat{\mathbf{x}}_j$
6: **for** $k = 2$ to n **do** ▷ remaining polynomials ($p^{(2)}, \ldots, p^{(n)}$)
7: $\quad$ **for** $i = n+1$ to 2 by -1 **do**
8: $\quad\quad \text{temp}_i \leftarrow \text{temp}_{i-1} + MP_{ii}^{(k)} \cdot \hat{\mathbf{x}}_i$
9: $\quad$ **end for**
10: $\quad \text{temp}_1 \leftarrow MP_{11}^{(k)} \cdot \hat{\mathbf{x}}_1$
11: $\quad \text{res}_k \leftarrow \sum_{j=1}^{n+1} \text{temp}_j \cdot \hat{\mathbf{x}}_j$
12: **end for**
13: **return** res

In line 5 the algorithm then computes the scalar product $\text{temp} \cdot \hat{\mathbf{x}}^T$. In the big loop (line 6 to 12) we evaluate the remaining polynomials $p^{(2)}, \ldots, p^{(n)}$. Again we start by computing the matrix vector product $\text{temp} = \hat{\mathbf{x}} \cdot MP^{(k)}$ (line 7 to 10). During this step we can reuse the values of temp_i $(i = 1, \ldots, n)$ computed during the evaluation of the previous polynomial. In fact, we can compute the product $\hat{\mathbf{x}} \cdot MP^{(k)}$ by using only n multiplications (instead of $\frac{(n+1)\cdot(n+2)}{2}$). In line 11 we finally compute the scalar product $\text{temp} \cdot \hat{\mathbf{x}}^T$.

To evaluate the $n \times n$ system $\mathcal{P}$, Algorithm 11.4 needs

$$\frac{(n+1)\cdot(n+4)}{2} + 2 \cdot n \cdot (n-1) \tag{11.3}$$

field multiplications. For a system over GF(256) with $n = 26$ (80 bit security) Algorithm 11.4 reduces the number of field multiplications needed during the evaluation process by a factor of 5.9 (compared to the evaluation of a random system using the standard approach). For $\mathbb{F} = \text{GF}(16)$, $n = 30$, the reduction factor is 6.5.

11.2.2 QUADLRS

In this subsection we describe how to use linear recurring sequences (LRS's) to speed up the keystream generation process of the QUAD stream cipher. By doing so, we get a variant of QUAD called QUADLRS.
The system parameter of QUADLRS is generated as shown in Algorithm 11.5. We run Algorithm 11.5 four times to get the quadratic systems $\mathcal{P}$, $\mathcal{Q}$, $\mathcal{S}_0$ and $\mathcal{S}_1 : \mathbb{F}^n \to \mathbb{F}^n$.
Let $\mathcal{P}$ be a multivariate quadratic system as generated by Algorithm 11.5. Then the corresponding matrices $MP^{(k)}$ $(k = 1, \ldots, n)$ (see equation (2.10)) look as shown in Figure 11.3. We have

$$MP_{ij}^{(k)} = \gamma_k \cdot MP_{i,j-1}^{(k)} \;\forall\; i \in \{1, \ldots, n\},\; j \in \{i+1, \ldots, n+1\},\; k \in \{1, \ldots, n\}. \tag{11.4}$$

Therefore we get

$$(x_1, \ldots, x_i) \cdot \begin{pmatrix} MP_{1j}^{(k)} \\ \vdots \\ MP_{ij}^{(k)} \end{pmatrix} = \gamma_k \cdot (x_1, \ldots, x_i) \cdot \begin{pmatrix} MP_{1,j-1}^{(k)} \\ \vdots \\ MP_{i,j-1}^{(k)} \end{pmatrix} \forall i \in \{1, \ldots, n\},\; j \in \{i+1, \ldots, n+1\},\; k \in \{1, \ldots, n\}. \tag{11.5}$$

Algorithm 11.5 Generation of a multivariate system of the LRS type

Input: parameter n

Output: multivariate system $\mathcal{P} : \mathbb{F}^n \to \mathbb{F}^n$ of the LRS type (given by its Macauley matrix M_P)

1: Choose a vector $\gamma \in \mathbb{F}^n$. The elements of γ have to be pairwise distinct.
2: **for** $k = 1$ to n **do**
3: Compute the first D elements of the linear recurring sequence $S = LRS(1, \gamma_k \cdot X + 1)$ and put them into a vector $\mathbf{b}^{(k)} \in \mathbb{F}^D$.
4: $M_P[k] \leftarrow \mathbf{b}^{(k)}$
5: **end for**
6: **return** M_P

$$MP^{(k)} = \begin{pmatrix} 1 & \gamma_i & \gamma_i^2 & \dots & \dots & \gamma_i^{n-2} & \gamma_i^{n-1} & \gamma_i^n \\ 0 & \gamma_i^{n+1} & \gamma_i^{n+2} & \dots & \dots & \gamma_i^{2n-2} & \gamma_i^{2n-1} & \gamma_i^{2n} \\ 0 & 0 & \gamma_i^{2n+1} & \dots & \dots & \gamma_i^{3n-3} & \gamma_i^{3n-2} & \gamma_i^{3n-1} \\ \vdots & & \ddots & & & \vdots & \vdots & \vdots \\ \vdots & & & & & \vdots & \vdots & \vdots \\ \vdots & & & & \ddots & \gamma_i^{D-6} & \gamma_i^{D-5} & \gamma_i^{D-4} \\ 0 & \dots & & \dots & & 0 & \gamma_i^{D-3} & \gamma_i^{D-2} \\ 0 & \dots & & \dots & & & 0 & \gamma_i^{D-1} \end{pmatrix} \qquad (k \in \{1, \dots, n\})$$

Figure 11.3: Matrices $MP^{(k)}$ for LRSQUAD

The boxes in Figure 11.3 illustrate this equation. The black boxes show the vector $(MP^{(k)}_{1,j-1}, \dots, MP^{(k)}_{i,j-1})^T$ on the right hand side of the equation, whereas the blue boxes represent the vector $(MP^{(k)}_{1j}, \dots, MP^{(k)}_{ij})^T$ on the left hand side. Note that one gets every blue box by multiplying the corresponding black box by γ_k $(k = 1, \dots, n)$.

Algorithm 11.6 uses this fact to evaluate the multivariate systems of QUADLRS much faster than random systems.

Algorithm 11.6 Evaluation of polynomials of the LRS type

Input: multivariate system $\mathcal{P} : \mathbb{F}^n \to \mathbb{F}^n$ of the LRS type (given in matrix form), internal state $\mathrm{IS} \in \mathbb{F}^n$

Output: $\mathrm{res} = \mathcal{P}(\mathrm{IS})$

1: $\hat{\mathbf{x}} \leftarrow (\mathrm{IS}_1, \dots, \mathrm{IS}_n, 1)$
2: **for** $k = 1$ to n **do**
3: $\mathrm{temp}_1 \leftarrow \hat{\mathbf{x}}_1$
4: **for** $i = 2$ to $n + 1$ **do**
5: $\mathrm{temp}_i \leftarrow \gamma_i \cdot \mathrm{temp}_{i-1} + MP^{(k)}_{ii} \cdot \hat{\mathbf{x}}_i$
6: **end for**
7: $\mathrm{res}_k \leftarrow \sum_{j=1}^{n+1} \mathrm{temp}_j \cdot \hat{\mathbf{x}}_j$
8: **end for**
9: **return** res

	n	9	10	11	12	13	14
GF(256)	random system	5.5 s	40.9 s	299.9 s	2,393 s	19,054 s	169,317 s
	part. circ. system	5.4 s	40.6 s	298.8 s	2,390 s	19,046 s	168,846 s
	LRS system	5.4 s	40.7 s	289.9 s	2,396 s	18,964 s	168,962 s

Table 11.1: Running time of the direct attack against QUAD (standard and improved versions)

Algorithm 11.6 works as follows:
Every iteration of the main loop evaluates one of the polynomials $p^{(k)}$ $(k = 1, \ldots, n)$. From line 3 to 6 we compute the matrix vector product $\hat{\mathbf{x}} \cdot MP^{(k)}$ and store the result in the vector temp. During the computation of temp_i $(i = 2, \ldots, n)$ we can reuse the value of temp_{i-1}, which enables us to compute the vector temp using only $2 \cdot n$ multiplications. Finally, in line 7, the algorithm computes the scalar product $\text{temp} \cdot \hat{\mathbf{x}}^T$.

To evaluate the whole system $\mathcal{P} : \mathbb{F}^n \to \mathbb{F}^n$, Algorithm 11.6 needs

$$n \cdot (3 \cdot n + 1) \text{ field multiplications.} \tag{11.6}$$

For QUAD over GF(256) with $n = 26$ (80 bit security), Algorithm 11.6 reduces the number of field multiplications by a factor of 6.0 (compared to the evaluation of a random system using the standard approach), for QUAD over GF(16) and $n = 30$ the reduction factor is 7.0.

11.3 Security

We analyzed the security of our schemes by performing a large number of experiments with MAGMA v.2.13-10. For this, we looked at determined systems of the random, partially circulant and LRS type and solved them using the MAGMA command `Variety`. As Table 11.1 shows, there is no significant difference between the running time for random, partially circulant and systems of the LRS type.

11.4 Parameters and Results

Based on our experiments presented in the previous section, we propose for our improved variants of QUAD the same parameters as for the standard QUAD stream cipher, namely (for 80 bit security)

- $n = 30$ for QUAD over GF(16) and
- $n = 26$ for QUAD over GF(256).

To check our theoretical results presented in Section 11.2 we created a straightforward C implementation of the QUAD stream cipher (both standard and improved variants). The program runs on a Lenovo ThinkPad with a single Intel Core 2 Duo processor with 2.53 GHz and 4 GB RAM. Tables 11.2 and 11.3 show the results of our experiments. For each of our schemes we encrypted 1,000 messages of different lengths and measured the running time of QUAD as well as the number of CPU cycles used during the encryption process.

		Data Throughput (kB/s)	CPU cycles/byte	speed up factor
GF(16) $n = 30$	random system standard approach	71.7	35,265	-
	random system alternative approach	53.4	47,325	-
	part. circ. system	458.3	5,513	6.4
	LRS system	477.0	5,298	6.8
GF(256) $n = 26$	random system standard approach	158	15,777	-
	random system alternative approach	113	21.860	-
	part. circ. system	875	2,820	5.5
	LRS system	915	2,730	5.8

Table 11.3: Data Throughput of QUAD (80 bit security)

		message size	100 kB		1 MB	
				s. f. [1]		s. f. [1]
GF(16) $n = 30$	random system standard approach	running time (ms)	1,403	-	14,014	-
		CPU cycles (10^9)	3.54	-	35.37	-
	random system alternative approach	running time (ms)	1,871	-	18,706	-
		CPU cycles (10^9)	4.73	-	47.23	-
	partially circulant system	running time (ms)	219	6.4	2,183	6.4
		CPU cycles (10^9)	0.56	6.4	5.52	6.4
	LRS system	running time (ms)	208	6.8	2,077	6.8
		CPU cycles (10^9)	0.52	6.8	5.23	6.8
GF(256) $n = 26$	random system standard approach	running time (ms)	625	-	6,229	-
		CPU cycles (10^9)	1.58	-	15.7	-
	random system alternative approach	running time (ms)	873	-	8,724	-
		CPU cycles (10^9)	2.21	-	22.0	-
	partially circulant system	running time (ms)	113	5.5	1,139	5.5
		CPU cycles (10^9)	0.30	5.5	2.88	5.5
	LRS system	running time (ms)	108	5.8	1,072	5.8
		CPU cycles (10^9)	0.27	5.7	2.68	5.8

[1] speed up factor compared to evaluating random systems with the standard approach

Table 11.2: Running time of QUAD (80 bit security)

As can be seen from Table 11.2, evaluating random polynomials using the standard approach is more efficient than doing it with the alternative approach. By using structured polynomials we can achieve a speed up of a factor of up to 6.8. The same holds for the data throughput of QUAD (see Table 11.3).

Conclusion

In this dissertation we addressed two of the open problems in the area of multivariate cryptography mentioned in the introduction of this thesis, namely

1. Parameter Choice of Multivariate Schemes and
2. Reducing Key Sizes of Multivariate Schemes.

In the first part of this thesis (Chapter 4 - 6) we considered the question of the parameter choice for multivariate cryptosystems. After defining an appropriate security model, we analyzed known attacks against multivariate schemes and Rainbow to find good parameters for these schemes for now and the near future.

We started with the model of Lenstra and Verheul (see Chapter 4), which, based on assumptions like the computational power and the budget of an attacker, yields the level of security a cryptographic scheme must reach to be thought secure in the year y.

In Chapter 5 we analyzed the complexity of direct attacks against multivariate systems. The main result here is presented by Table 5.5 which, for the years 2012 to 2050, shows the minimal number of equations in a system needed to reach the security level proposed by our model. Additionally, this chapter provides recommended parameters for the UOV signature scheme for different levels of security (see Table 5.6).

In Chapter 6 we dealt with the question of parameter choice for the Rainbow signature scheme. We developed a strategy which allows us to find, for a given level of security, parameters for Rainbow which minimize both public and private key size. We did our analysis separately for Rainbow schemes over GF(16), GF(31) and GF(256) (see Tables 6.4, 6.8 and 6.13). As a further result we found that, in terms of signature length, Rainbow schemes over GF(16) lead to the best results, whereas Rainbow schemes over GF(31) have the smallest public and private keys.

In the second part of the dissertation (Chapters 7 - 11) we proposed a way to reduce the public key size and to speed up the verification process of the UOV and Rainbow signature schemes. Using the results of the first part we showed that the security of the schemes is not weakened by our construction.

In the first chapter of this part (Chapter 7) we presented the general idea of our construction to create UOV and Rainbow key pairs with a structured public key. To do this, we showed how to insert an arbitrary matrix B into the Macauley matrix M_P of the public key of the multivariate signature schemes.

In Chapter 8 we described several improved versions of the UOV signature scheme created by the techniques presented in the previous chapter. The main result here is shown in Table 8.8 which compares the public key sizes of our improved schemes with those of the original UOV scheme. For UOVLRS2 over GF(256) for example, we can reduce the public key size from 99.9 to 13.5 kB (80 bit security), which means a reduction by a factor of 7.5. Besides these UOV versions with reduced public key size, we showed with UOVrand a variant of UOV which offers provable security under direct attacks.

Chapter 9 presented two improved versions of the Rainbow signature scheme which reduce the public key size of the original scheme by a significant factor. The main result here is given by Table 9.8 which shows that e.g. for RainbowLRS2 over GF(256), we can reduce the public key size from 25.1 to 9.8 kB (80 bit security) which means a reduction by a factor of 2.5.

In Chapter 10 we showed how the structure in the public key of our schemes can be used to speed up the verification process of the improved schemes. The main results here are given by the Tables 10.2 and 10.3. As the tables show we can, by using the structure in the public key, achieve speed ups of the verification process of UOV and Rainbow by factors of 6.1 and 2.4 respectively.

In the last chapter of this part (Chapter 11) we showed how we can use structured polynomials to speed up the multivariate stream cipher QUAD [2]. The main result is presented in Table 11.2 which shows that we can use our techniques to speed up the key stream generation process of QUAD by a factor of up to 6.8.

We hope that our work will help to overcome the disadvantages of multivariate cryptography and make multivariate cryptosystems more suitable for the use in practice.

Future Work

As Future Work we plan to address the two problems "Developing Multivariate Schemes with Provable Security" and "Developing "Advanced" Multivariate Cryptosystems" mentioned in the introduction of this thesis (see page 15).

As stated in Section 2.3, the security of most of the existing multivariate public key cryptosystems is not solely based on the MQ-Problem (see Subsection 2.3.1), but also on some version of the IP-Problem. This fact has prevented researchers to establish security proofs for their schemes and led to the fact, that a number of multivariate schemes were broken. This development has undermined the confidence of users in the security of multivariate cryptography in general and prevented multivariate schemes from being used in practice. It is therefore necessary to create multivariate schemes which offer provable security. One direction we want to follow is to base multivariate schemes on the provable secure identification scheme of Sakumoto et al. [52] (see also Subsection 2.2.2). Via the Fiat-Shamir paradigm [26] it is possible to convert this identification scheme into a provable secure multivariate signature scheme. This direction of research is closely related to the creation of multivariate signature schemes with special properties. One first result in this area is our provable secure threshold ring signature scheme [P4]. We plan to go further in this direction and to extend the identification scheme of [52] to a ring and group identification and signature scheme.

Another way to create multivariate signature schemes with special properties we want to look at is by developing new advanced schemes on the basis of existing multivariate signature schemes like UOV and Rainbow. In this area we are working on a multivariate blind signature scheme based on UOV using the techniques described in Chapter 7. Furthermore we want to explore if these techniques can be used to develop other cryptographic primitives, such as key exchange protocols, on the basis of multivariate polynomials.

Content of the CD

Besides electronic versions of this dissertation and our publications the CD contains implementations of our improved versions of the UOV and Rainbow signature schemes. These implementations can be divided into three groups

MAGMA code

folders UOV, cyclicUOV, UOVLRS, cycUOVLRS, UOVLRS2, 0/1UOV, Rainbow, cyclicRainbow, RainbowLRS2

Each of the upper folders contains the three modules `keygen`, `sign` and `verify`.
The module `keygen` generates a UOV/Rainbow key pair and stores it in the files `public_key.txt` and `private_key.txt`. The module `sign` reads the private key from the file and generates a signature for a random message. Signature and message are stored in the file `signature.txt`. The module `verify` reads the public key and the message/signature pair from the files and checks the correctness of the signature.
Note that the implementations of the improved schemes are suited only for small toy examples since the generation/inversion of the large matrices A_{UOV} and A_{Rb} is too expensive.

C programs for Windows

- folder QUAD: contains C programs for QUAD, cyclicQUAD and QUADLRS
- folder Verification: contains C programs for UOV and Rainbow (both standard and improved versions)

  The programs in these folders are mainly used to determine the running time of the verification process of our improved schemes.

C programs for Linux

- folder UOVRainbow: contains implementations of UOV/Rainbow and improved versions (using the M4RIE library)
  Each of the projects is divided into the three modules `keygen`, `signgen` and `signveri`.
- folder QUAD: implementation of QUAD over GF(256) and GF(2).

  To run the above projects, one needs the software Oracle Virtual Box including a Linux distribution containing the M4RIE library. Unfortunately, the size of this distribution is much too big for this CD.

References

[1] M. Albrecht. M4RIE library for fast computations with dense matrices over $GF(2)^e$. m4ri.sagemath.org, 2010.

[2] C. Berbain, H. Gilbert and J. Patarin. QUAD: A practical stream cipher with provable security. In *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 109–128. Springer, 2006.

[3] D. Bernstein. The Salsa20 family of stream ciphers. In *The eSTREAM Finalists*, volume 4986 of *Lecture Notes in Computer Science*, pages 84–97. Springer, 2008.

[4] D. Bernstein, J. Buchmann and E. Dahmen, editors. *Post Quantum Cryptography*. Springer, 2008.

[5] L. Bettale, J.-C. Faugère and L. Perret. Hybrid approach for solving multivariate systems over finite fields. *Journal of Mathematical Cryptology*, 3:177–197, 2009.

[6] O. Billet and H. Gilbert. Cryptanalysis of Rainbow. In *SCN*, volume 4116 of *Lecture Notes in Computer Science*, pages 336–347. Springer, 2006.

[7] A. Bogdanov, T. Eisenbarth, A. Rupp and C. Wolf. Time-area optimized public-key engines: MQ-cryptosystems as replacement for elliptic curves? In *CHES*, volume 5154 of *Lecture Notes in Computer Science*, pages 45–61. Springer, 2008.

[8] W. Bosma, J. Cannon and C. Playoust. The MAGMA algebra system. I. The user language. *Journal of Symbolic Computation*, 24(3-4):235–265, 1997.

[9] C. Bouillaguet, H.-C. Chen, C.-M. Cheng, T. Chou, R. Niederhagen, A. Shamir and B.-Y. Yang. Fast exhaustive search for polynomial systems in $\mathbb{F}_2$. In *CHES*, volume 6225 of *Lecture Notes in Computer Science*, pages 203–218. Springer, 2010.

[10] B. Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal*. PhD thesis, Innsbruck, 1965.

[11] A. I.-T. Chen, M.-S. Chen, T.-R. Chen, C.-M. Cheng, J. Ding, E. L.-H. Kuo, F. Y.-S. Lee and B.-Y. Yang. SSE implementation of multivariate PKCs on modern x86 CPUs. In *CHES*, volume 5747 of *Lecture Notes in Computer Science*, pages 33–48. Springer, 2009.

[12] C. Clough, J. Baena, J. Ding, B.-Y. Yang and M.-S. Chen. Square, a new multivariate encryption scheme. In *CT-RSA*, volume 5473 of *Lecture Notes in Computer Science*, pages 252–264. Springer, 2009.

[13] D. Coppersmith, J. Stern and S. Vaudenay. Attacks on the birational signature scheme. In *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 435 –443. Springer, 1994.

[14] N. Courtois, A. Klimov, J. Patarin and A. Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 392–407. Springer, 2000.

[15] J. Ding, J. E. Gower and D. S. Schmidt. *Multivariate Public Key Cryptosystems*. Springer, 2006.

[16] J. Ding and D. Schmidt. Rainbow, a new multivariable polynomial signature scheme. In *ACNS*, volume 3531 of *Lecture Notes in Computer Science*, pages 164–175. Springer, 2005.

[17] J. Ding, C. Wolf and B.-Y. Yang. ℓ-invertible cycles for multivariate quadratic public key cryptography. In *PKC*, volume 4450 of *Lecture Notes in Computer Science*, pages 266–281. Springer, 2007.

[18] J. Ding, B.-Y. Yang, C.-H. O. Chen, M.-S. Chen and C.-M. Cheng. New differential-algebraic attacks and reparametrization of Rainbow. In *ACNS*, volume 5037 of *Lecture Notes in Computer Science*, pages 242–257. Springer, 2008.

[19] J. Ding, B.-Y. Yang, V. Dubois, C.-M. Cheng and O. Chen. Breaking the symmetry: a way to resist the new differential attack. http://eprint.iacr.org/2007/366.

[20] V. Dubois, P. A. Fouque, A. Shamir and J. Stern. Practical cryptanalysis of Sflash. In *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, page 1–12. Springer, 2007.

[21] V. Dubois and N. T. Gama. The degree of regularity of HFE systems. In *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 557–576. Springer, 2010.

[22] N. Eén and N. Sörensen. MINISAT. minisat.se.

[23] J.-C. Faugère. A new efficient algorithm for computing Gröbner Bases (F4). *Journal of Pure and Applied Algebra*, 139:61–88, 1999.

[24] J.-C. Faugère. A new efficient algorithm for computing Gröbner Bases without reduction to zero (F5). In *ISSAC 2002*, pages 75–83. ACM Press, 2002.

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

[26] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[27] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[28] S. W. Golomb. *Shift Register Sequences*. Holden Day, San Francisco, 1967.

[29] S. W. Golomb and G. Gong. *Signal Design for Good Correlation*. Cambridge University Press, 2005.

[30] L. Goubin and N. Courtois. Cryptanalysis of the TTM cryptosystem. In *ASIACRYPT*, volume 1976 of *Lecture Notes in Computer Science*, pages 44–57. Springer, 2000.

[31] F. Gray. Pulse code communication. US Patent 2,632,058, 1953.

[32] Y.-J. Huang, F.-H. Liu and B.-Y. Yang. Public-key cryptography from new multivariate quadratic assumptions. In *PKC*, volume 7293 of *Lecture Notes in Computer Science*, pages 190–205. Springer, 2012.

[33] A. Kipnis, J. Patarin and L. Goubin. Unbalanced Oil and Vinegar schemes. In *EUROCRYPT*, volume 1592 of *Lecture Notes in Computer Science*, pages 206–222. Springer, 1999.

[34] A. Kipnis and A. Shamir. Cryptanalysis of the Oil and Vinegar signature scheme. In *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 257–266. Springer, 1998.

[35] A. K. Lenstra and H. W. Lenstra. *The Development of the Number Field Sieve*. Springer, 1993.

[36] A. K. Lenstra and E. R. Verheul. Selecting cryptographic key sizes. In *PKC*, volume 1751 of *Lecture Notes in Computer Science*, pages 446–465. Springer, 2000.

[37] R. Lidl and W. Niederreiter. *Finite Fields and their application*. Cambridge University Press, 1986.

[38] T. Matsumoto and H. Imai. Public quadratic polynomial-tuples for efficient signature verification and message-encryption. In *EUROCRYPT*, volume 330 of *Lecture Notes in Computer Science*, pages 419–553. Springer, 1988.

[39] M. S. E. Mohamed, J. Ding and J. Buchmann. Towards algebraic cryptanalysis of HFE challenge 2. In *ISA*, volume 200 of *Communications in Computer and Information Science*, pages 123–131. Springer, 2011.

[40] H. Miura, Y. Hashimoto and T. Takagi. Extended Algorithm for Solving Underdefined Multivariate Quadratic Equations. In *PQCrypto*, volume 7932 of *Lecture Notes in Computer Science*, pages 118 - 135. Springer, 2013.

[41] National Institute for Standards and Technology. Digital signature standard. FIPS PUB 186-3, 130 pages, 2009.

[42] D. Neßelmann. Gröbnerbasen und nichtlineare Gleichungssysteme. Vorlesungsskript. Rostock, 2009.

[43] H. Ong, C.-P. Schnorr and A. Shamir. Efficient signature schemes based on polynomial equations. In *CRYPTO*, volume 196 of *Lecture Notes in Computer Science*, pages 37–46. Springer, 1984.

[44] J. Patarin. Cryptanalysis of the Matsumoto and Imai public key scheme of EUROCRYPT'88. In *CRYPTO*, volume 963 of *Lecture Notes in Computer Science*, pages 248–261. Springer, 1995.

[45] J. Patarin. Assymmetric cryptography with a hidden monominal. In *CRYPTO*, volume 1109 of *Lecture Notes in Computer Science*, pages 45–60. Springer, 1996.

[46] J. Patarin. Hidden field equations (HFE) and isomorphisms of polynomials (IP): two new families of asymmetric algorithms. In *EUROCRYPT*, volume 1070 of *Lecture Notes in Computer Science*, pages 33–48. Springer, 1996.

[47] J. Patarin. The Oil and Vinegar signature scheme. presented at the Dagstuhl Workshop on Cryptography, September 1997.

[48] J. Patarin, N. Courtois and L. Goubin. Flash, a fast multivariate signature algorithm. In *CT-RSA*, volume 2020 of *Lecture Notes in Computer Science*, pages 298–307. Springer, 2001.

[49] B. Preneel. The NESSIE project: Towards new cryptographic algorithms. www.cryptonessie.org, 2000.

[50] A. Pyshkin. *Algebraic cryptanalysis of block ciphers using Gröbner Bases*. PhD thesis, TU Darmstadt, 2008.

[51] R. L. Rivest, A. Shamir and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[52] K. Sakumoto, T. Shirai and H. Hiwatari. Public-key identification schemes based on multivariate quadratic polynomials. In *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 706–723. Springer, 2011.

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

[54] M. Soos. CryptoMiniSat. www.msoos.org.

[55] C. Tao, A. Diene and J. Ding. Simple Matrix Scheme for Encryption. In *PQCrypto 2013*, volume 7932 of *Lecture Notes in Computer Science*, pages 231 - 242. Springer, 2013.

[56] E. Thomae and C. Wolf. Solving underdetermined systems of multivariate quadratic equations revisited. In *PKC*, volume 7293 of *Lecture Notes in Computer Science*, pages 156–171. Springer, 2012.

[57] P. Turán. On an extremal problem in graph theory. *Matematiko Fizicki Lapok*, 48:436 – 452, 1941.

[58] L.-C. Wang, B.-Y. Yang, Y.-H. Hu and F. Lai. A "medium-field" multivariate public-key encryption scheme. In *CT-RSA*, volume 3860 of *Lecture Notes in Computer Science*, pages 132–149. Springer, 2006.

[59] C. Wolf and B. Preneel. Equivalent keys in HFE, C^*, and variations. In *Mycrypt*, volume 3715 of *Lecture Notes in Computer Science*, pages 33–49. Springer, 2005.

[60] C. Wolf and B. Preneel. Equivalent keys in multivariate quadratic public key systems. http://eprint.iacr.org/2005/464.

[61] B.-Y. Yang and J.-M. Chen. Building secure tame-like multivariate public-key cryptosystems: The new TTS. In *ACISP*, volume 3574 of *Lecture Notes in Computer Science*, pages 518–531. Springer, 2005.