Universität Regensburg
Fakultät für Informatik und Data Science
Lehrstuhl für Datensicherheit und Kryptographie

Practical Challenges of UOV-based Signature Schemes: Physical Security and Advanced Security Notions

Dissertation
zur Erlangung des Grades Doctor rerum naturalium (Dr. rer. nat.)
eingereicht an der Fakultät für Informatik und Data Science
der Universität Regensburg
vorgelegt von

Thomas Aulbach

Erstgutachterin: Prof. Dr. Juliane Krämer
Zweitgutachter: Prof. Dr. Tim Güneysu

Tag der Disputation: 05. November 2025

Achknowledgement

The completion of such a large project provides a good opportunity to pause and reflect on the past few years. I have met many people who have had a profound influence on me as a person and as a researcher, and to whom I would like to express my heartfelt gratitude.

First of all, I would like to thank my supervisor, Prof. Juliane Krämer. Juliane introduced me to the topics of post-quantum cryptography and physical security and provided me with tremendous support throughout my doctoral studies. She has a very calm and composed, but also humorous manner, which made working with her very enjoyable. I am very grateful for her professional advice, the research projects we completed together, and her reliability. I could not have imagined a better supervisor for this period.

I would also like to thank the second reviewer of my thesis, Prof. Tim Güneysu. I am glad that he agreed to review my work and I was very pleased to get to know him better in the process. Thanks also to the members of the examination committee, Prof. Radu Curticapean and Prof. Philipp Rümmer. My sincere thanks also go to Prof. Fabio Campos, Prof. Simona Samardjiska, and Dr. Patrick Struck, who formed my thesis advisory board and supported me not only in joint research projects but throughout my entire doctoral studies. In addition, I would like to thank all my co-authors, from whom I have learned so much. My thanks also go to Prof. Steuding, the supervisor of my bachelor's and master's theses, and Prof. Reith, who introduced me to Juliane.

I consider myself fortunate to have been part of an appreciative team during my time as a doctoral student, a team that supports each other and never loses sight of the fun in their work. The colleagues I spent the most time with are Patrick, Samed, Michael, and Maxi. Patrick is incredibly helpful, patient, and kind—simply a wonderful friend. Even his move to Konstanz did not prevent him from always being there for me and many others in the group. Thank you very much! Samed is a selfless person with many interests from whom I was able to learn an enormous amount. Our conversations about math, sports, politics, and justice constantly taught me new things. Thank you for your tremendous support. Michael, my office mate, darts rival, and backgammon mentor. Thank you for all the fun moments we shared and for being a role model for me in scientific work and research. Maxi is a great friend. She is very kind and empathetic, but also humorous, and it was inspiring to work with her.

All four of you are absolute role models when it comes to staying calm under pressure and working with focus. Our many breaks, evenings after work, and business trips together have always recharged my motivation. I laughed a lot with you and will never forget the last few years. Many thanks also to Antoine for fun conversations about food and our runs together. I really liked your pragmatic approach to nutrition and sports, and I immediately internalized your trick for peeling eggs with a spoon. My thanks also go to my former and new colleagues Oscar, Harrison, Stefan, and Jan. I would have loved to work with you longer and hope that our paths will continue to cross. Furthermore, I want to thank Heidi for her continuous support at university.

Special thanks go to my parents, Andreas and Reinhilde, who sparked my enthusiasm for lifelong learning, especially when I was young, and gave me the freedom to develop in the way I wanted to. Many thanks also to my sister Juliane, from whom I learned what it means to act independently, to persevere, and that it is worth taking unusual paths. I would also like to thank all my friends in Würzburg, Gräfendorf, Munich, Stuttgart, and many other places, who have shown me time and again how multifaceted and wonderful life is.

Most importantly, a huge thank you to my wife Marlene. She has accompanied me throughout my entire academic career and witnessed all its ups and downs. Thank you for your encouragement, distractions, and incredible support during this long journey. It is a wonderful experience to raise our little Mariella, who is herself the greatest motivation to always give it our best.

Würzburg, December 12, 2025
Thomas

Abstract

This thesis presents research results on the challenges of deploying UOV-based signature schemes in practical applications.

UOV-based signature schemes constitute a popular branch of multivariate cryptography, one of the oldest families of post-quantum cryptography. They offer desirable features, like small signature sizes as well as fast signing and verification times, but further research about their security is indispensable before they can be considered for use in security protocols. In this thesis, we address these challenges and contribute to the research area of UOV-based signature schemes in three directions.

First, we develop and implement concrete physical attacks against the signature schemes Rainbow, MAYO, and UOV. This includes the detection of vulnerable code lines, a detailed description of the side-channel information or fault propagation achieved by the physical attack, and the presentation of algebraic tools that are utilized to recover the secret key. Moreover, we either simulate the attack on an emulation platform or execute it directly on a target device.

Second, we further contribute to more resistant implementations of UOV-based signature schemes. On the one hand, we create an overview of vulnerabilities based on all side-channel and fault attacks available in the literature. This overview is developed especially for UOV and transferred to all UOV-based signatures that are candidates in the ongoing standardization process initiated by NIST. On the other hand, we implement and benchmark dedicated countermeasures that are designed to mitigate these attacks. Together, these two parts enhance the knowledge about the physical security of UOV-based signatures.

Third, there are approaches to reduce the public key size of UOV by utilizing sparse polynomials instead of random polynomials in the scheme specific equations. We analyze the security of three variants of such an instantiation, called MQ-Sign, and present a polynomial-time key-recovery attack against two of them, proving their insecurity. Furthermore, we analyze the security of several UOV-based signature schemes with respect to advanced security notions (the BUFF notions), which are not covered by the standard EUF-CMA security notion. These notions are crucial to protect certain real-world protocols against misuse. Finally, we deduce a lightweight method to achieve these notions, which works equally for all analyzed UOV-based signature schemes.

Publications

Publications Contributing to this Thesis

[**A**CK25] Thomas Aulbach, Fabio Campos, and Juliane Krämer. "SoK: On the Physical Security of UOV-based Signature Schemes". In: *International Conference on Post-Quantum Cryptography*. Springer. 2025.

[**A**CK+23] Thomas Aulbach, Fabio Campos, Juliane Krämer, Simona Samardjiska, and Marc Stöttinger. "Separating Oil and Vinegar with a Single Trace: Side-Channel Assisted Kipnis-Shamir Attack on UOV". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 3 (2023).

[**A**DM+24] Thomas Aulbach, Samed Düzlü, Michael Meyer, Patrick Struck, and Maximiliane Weishäupl. "Hash Your Keys Before Signing: BUFF Security of the Additional NIST PQC Signatures". In: *International Conference on Post-Quantum Cryptography*. Springer. 2024.

[**A**KK+22] Thomas Aulbach, Tobias Kovats, Juliane Krämer, and Soundes Marzougui. "Recovering Rainbow's Secret Key with a First-Order Fault Attack". In: *International Conference on Cryptology in Africa*. Springer. 2022.

[**A**MS+24] Thomas Aulbach, Soundes Marzougui, Jean-Pierre Seifert, and Vincent Quentin Ulitzsch. "MAYo or MAY-not: Exploring implementation security of the post-quantum signature scheme MAYO against physical attacks". In: *2024 Workshop on Fault Detection and Tolerance in Cryptography (FDTC)*. IEEE. 2024.

[**A**ST24] Thomas Aulbach, Simona Samardjiska, and Monika Trimoska. "Practical Key-Recovery Attack on MQ-Sign and More". In: *International Conference on Post-Quantum Cryptography*. Springer. 2024.

[SM**A**+24] Oussama Sayari, Soundes Marzougui, Thomas Aulbach, Juliane Krämer, and Jean-Pierre Seifert. "HaMAYO: A Fault-Tolerant Reconfigurable Hardware Implementation of the MAYO Signature Scheme". In: *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer. 2024.

Further Publications

[MKK+23] Soundes Marzougui, Ievgen Kabin, Juliane Krämer, Thomas Aulbach, and Jean-Pierre Seifert. "On the Feasibility of Single-Trace Attacks on the Gaussian Sampler using a CDT". In: *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer. 2023.

Contents

List of Figures

List of Tables

List of Acronyms

1

Introduction

The desire to protect information during transmission appears natural and is indeed very old. Attempts of obscuring messages such that only specific receiver have access to them, i.e., achieving confidentiality, can be traced back to ancient Egypt, Mesopotamia, and ancient Greece and Rome. It started with very simple methods, like substitution or transposition ciphers, e.g., the scytale and the well-known Caesar cipher. The main goal at this time was to keep knowledge of manufacturing methods or planned military operations secret.

Besides confidentiality one expects more from a communication path that is called secure. Information should not be altered or tampered with during its storage and transmission, which is commonly termed as integrity. Furthermore, authentication is an important aspect to verify the identity of parties involved in the communication or data exchange.

Nowadays, designing techniques, algorithms, and protocols to securing communication is attributed to the field of cryptography. Although it is a highly diverse field with numerous procedures in improbably many applications and scenarios, there are some core techniques one should be aware of. To encrypt messages between two parties, one commonly uses symmetric-key algorithms. The most prominent algorithm is the Advanced Encryption Standard (AES), which belongs to the class of block ciphers. Symmetric-key algorithms are generally quite efficient, but require a secret key, which the two parties share. In order to agree on a common secret key over a possibly insecure channel, one needs to employ techniques from asymmetric or public-key cryptography (PKC). The key exchange is classically realized by schemes like Diffie-Hellman (DH), elliptic-curve Diffie-Hellman (ECDH), or other variations of them. These algorithms are based on the factorization problem or the discrete logarithm problem. Currently, there is no algorithm known that can solve either of these problems efficiently on classical computers. The same holds for digital signatures, like RSA and the Digital Signature Algorithm (DSA) or its elliptic-curve variant elliptic-curve Digital Signature Algorithm (ECDSA), which are implemented to authenticate the two communicating parties and ensure data integrity.

The former paragraph conveys a glimpse of the interaction of symmetric and asymmetric cryptography in frequently used protocols, as Transport Layer Security (TLS) or Secure Shell (SSH), just to name a few. The importance of these protocols render secure cryptographic algorithms absolutely indispensable

in any kind of public information system. However, the security of all aforementioned asymmetric schemes is threatened by the theoretical breakthrough constituted by Shor's quantum algorithms [Sho94] and the continuous advances in the physical development of quantum computers. Shor presented polynomial-time solutions for factoring and computing discrete logarithms, but they place massive demands on the number of qubits and error rate of the quantum computer, which go quite far beyond currently realizable capacity and reliability. A popular diagram and brief explanation of the quantum computing landscape can be found in [Jaq24]. Furthermore, [BSI24] contains a status report on quantum computer development, estimating their relevance for cryptography.

Due to these heavy requirements, the sword of Damocles to asymmetric cryptography, which Shor's algorithms present, received little attention for some years until significant advances in quantum computing made the threat more real. Driven by the gigantic opportunities large-scale quantum computers offer, many resources were put into their development. Competitors like Google Quantum AI and IBM composed ambitious roadmaps [Goo23; IBM24] towards large scale error-corrected quantum computers. The successive completion of strategic intermediate milestones visualize the relentless development progress and it seems that quantum computers capable of breaking cryptographic algorithms based on the factorization or discrete logarithm problem are within reach in the foreseeable future. The exact year of their arrival is difficult to estimate, since unpredictable setbacks or breakthroughs might appear along the way, but cybersecurity experts have agreed that the transition from classical to quantum-safe or post-quantum cryptography (PQC) is inevitable and should be prepared as soon as possible [Mos18].

The various approaches, that are currently thought to be secure, even in the presence of quantum computers, can be divided into the following families: hash-based and multi-party computation (MPC)-in-the-head signatures, as well as code-based, multivariate, lattice-based, and isogeny-based cryptography. Each of them is equipped with its own set of advantages and drawbacks, stemming mainly from the underlying mathematical hardness assumption. Furthermore, many of the suggested algorithms are quite young and their security needs to be evaluated consistently. In 2016, the National Institute for Standards and Technology (NIST) started the PQC Standardization Process with a first call for proposals ending in November 2017 [NIST17]. The goal was to evaluate and standardize one or more quantum-resistant digital signature algorithm and key-encapsulation mechanism (KEM). Indeed, after three evaluation rounds, three new standards have been published in August 2024, under the abbreviation FIPS 203-205, namely ML-KEM, ML-DSA, and SLH-DSA. The key encapsulation ML-KEM is derived from the lattice-based submission KYBER. The digital signatures ML-DSA and SLH-DSA are derived from the lattice-based submission DILITHIUM and the hash-based submission SPHINCS$^+$, respectively.[1] However, despite putting forth some new, hopefully quantum-safe standards, the standardization process is not yet completed. The community is still looking for secure algorithms, to diversify the portfolio of standards and find suitable solutions for various applications. After a fourth evaluation round just for KEMs, the code-based scheme HQC was selected for standardization in March 2025.

[1] Another winner, named FALCON, was announced among the digital signatures. FALCON, which is also based on lattices, will be included in the FIPS 206 standard and dubbed FN-DSA.

Regarding digital signatures, the lack of alternatives to lattice-based schemes led to a completely new call for proposals that ended in June 2023. Meanwhile, this additional standardization process [NIST22] entered its second round, with 14 candidates for digital signatures remaining.

This thesis deals with multivariate signature schemes that employ the oil-and-vinegar principle. They are also often termed UOV-based signatures. All of the four multivariate candidates that advanced to the second round of the additional standardization process are of this type. With their very short signature sizes, and efficient signing and verification time, these schemes constitute attractive candidates for standardization. In the next section, we will give a brief history of multivariate cryptography with an emphasis on UOV-based signature schemes, as these represent the main scope of this thesis.

1.1 Multivariate Cryptography

All multivariate cryptosystems build upon the problem of finding solutions to a system of nonlinear polynomial equations. This problem is shown to be NP-complete even in its simplest version of quadratic polynomials over the finite field with two elements [FY79; GJ79]. For the sake of efficiency, most multivariate cryptosystems employ the aforementioned case where all polynomials have degree two. Then, the problem of solving the polynomial system is called the MQ problem (for Multivariate Quadratic).

Let the system consist of m quadratic equations in n variables. If $m \sim n$, i.e., the system is not heavily under- or overdetermined, then all known algorithms to solve this system have exponential running time in n and m. There are many ways to develop an encryption or signature scheme on top of this problem. Most commonly, a trapdoor is installed into the system of equations, that allows only the person with knowledge of the trapdoor to find solutions efficiently. We defer the details of how this is technically realized for UOV-based signature schemes to Section 2.2.

In the following, we want to provide a comprehensive overview of the landscape of multivariate cryptography, starting with the Matsumoto-Imai (MI) cryptosystem [MI88]. After breaking this cryptosystem [Pat95], Patarin (and others) designed schemes based on hidden field equations (HFE) [Pat96] and the oil and vinegar (OV) principle [KPG99]. A lot—but not all—of the subsequently developed schemes can be assigned to one of the three branches emanating from these design principles, as visualized in Figure 1.1. It is recognizable that the history of multivariate encryption and signature schemes is characterized by numerous successful cryptanalytic attacks. In order to restore the security of the affected schemes, several variants or modifiers were applied, which eventually got broken again by adapted or new attacks. Two prominent examples are the attacks [Beu22; TPD21] on GeMSS [CFM+20] and Rainbow [DCP+20], which had a significant impact on the PQC standardization process, as indicated above. The repeated adaption of broken or weakened schemes, led to a vast amount of interrelated cryptographic schemes, of which currently only a small group of signature schemes are considered secure and efficient enough for practical use. This is underlined by the fact that all four multivariate schemes that advanced to the second round of NIST's call for additional signatures are

UOV itself, or variants of it.[2] In Figure 1.1 we mark these particular schemes with green background color. To visualize all UOV-based signatures, which are analyzed in at least one of the publications contained in this thesis, we give them a thick dashed frame. All of them have been relevant either in a standardization process by NIST [NIST17; NIST22] or in the Korean PQC competition (KpqC) [QRC21].

1.2 Digital Signatures in Real-World Applications

Digital signatures are an elegant tool to provide authenticity and data integrity for electronic communication between two parties. Therefore, they are ubiquitous in all technological areas. One of the most prominent examples is their deployment in the TLS protocol, to secure client-server communication over computer networks. Here, the server needs to provide a digital certificate to prove its identity and the integrity of its displayed public key. This certificate also contains the digital signature of a Certificate Authority (CA) that vouches for the validity of the certificate's content. If the CA is a trusted entity, the client can then use the public key to generate session keys for secure communication, i.e., proceed with the TLS handshake. We will return to the specifics of the TLS protocol later in this section and analyze where and how many signatures it uses, since this will pose an interesting example for the different use-cases of signature schemes.

First, we want to present more application areas of digital signatures. They are used by email providers and messaging platforms to guarantee the identity of the communicating parties and ensure that transferred messages are not tampered with. The same holds for software distribution and updates. Software vendors sign their executable files, libraries, and patches, to prove they originate from a legitimate source and have not been modified. Furthermore, digital signatures are a fundamental building block of cryptocurrencies, because they are used to authenticate transactions in blockchains. Here, they ensure that only the owner of a secret key can authorize a transfer from that wallet. Another important field is the one of embedded systems and the internet of things (IoT), especially in the industrial setting. IoT devices need to communicate with centralized servers or cloud platforms. On the one hand, it is essential to verify the authenticity of the device, to prohibit an adversary from infiltrating the system with a corrupted device. On the other hand, the commands sent from the cloud to the specific devices need to be trustworthy, in order to guarantee a safe and functional operating process. In many cases, these systems operate in constrained environments with limited computational power, memory, and energy resources, posing additional requirements to the employed signature schemes. There are countless other fields of application, where authenticity, data integrity, and security is paramount. This includes governmental services, electronic voting, healthcare systems, legal affairs, and many more.

The examples above illustrate that digital signatures are absolutely indispensable in our everyday world. Before we introduce the technical definition and properties in Section 2.1, we want to indicate the operating principle and

[2] In total, 14 signature schemes made it to Round 2 [NIST24]. The four multivariate ones are MAYO, QR-UOV, SNOVA, and UOV.

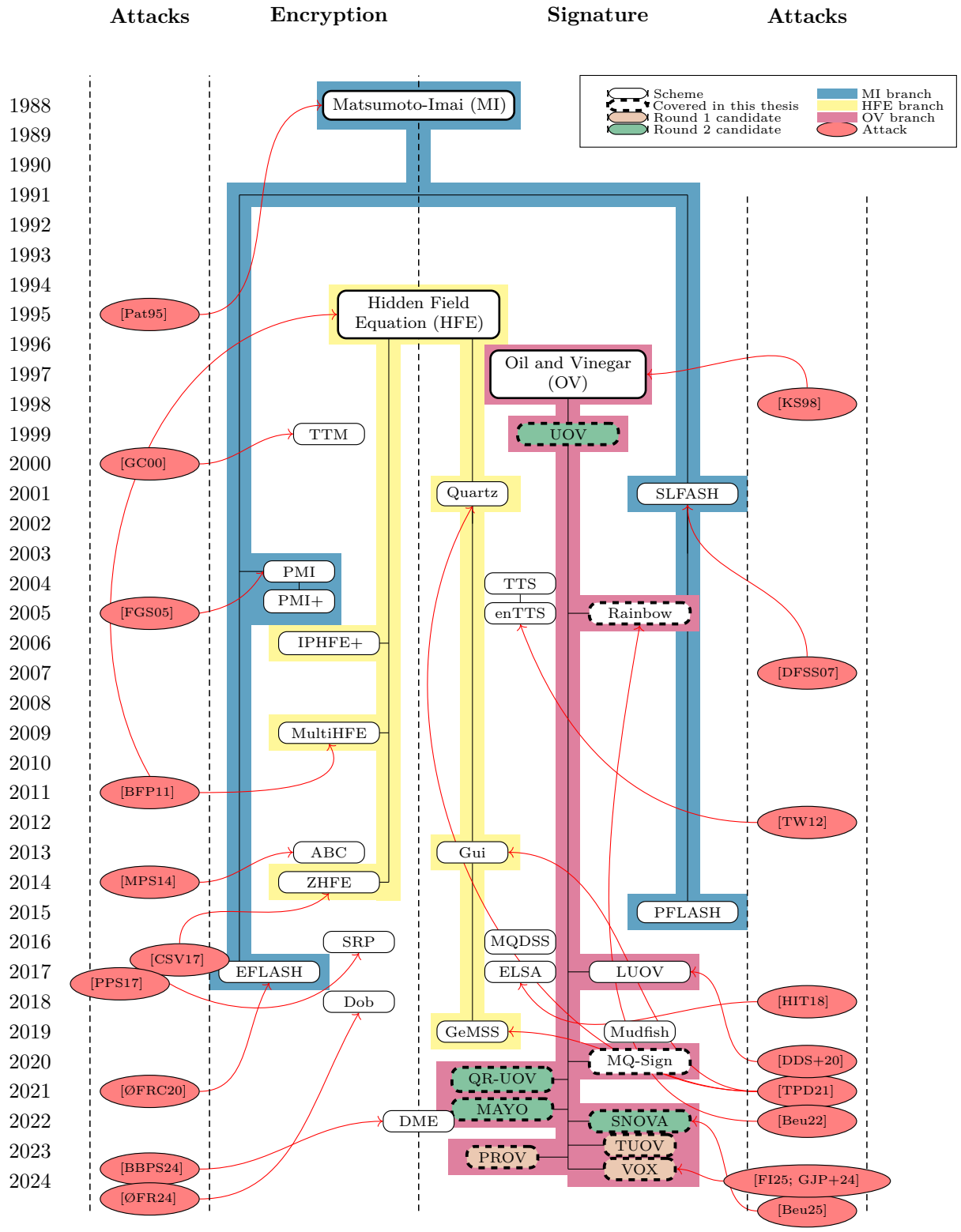


Figure 1.1: The landscape of multivariate cryptography. Schemes with strong relations to each other are connected with a line. The shown attacks had considerable impact on the schemes, by either breaking them or strongly affecting the parameter choices.

performance measures for signature schemes. Furthermore, we present concrete examples of classical and post-quantum signature schemes.

1.2.1 Working Principle

Roughly speaking, digital signatures use cryptographic algorithms with underlying mathematical problems to link an identity to a message, just as written signatures tie a person to a particular document. To this end, users generate a pair of keys, one private and one public key. The public key represents the owner's identity and is publicly available. The private key is kept secret, allowing only its owner to sign specific messages. The generated signature is then sent alongside the message. Thus, anyone can use the displayed public key to verify that first, the owner of that public key really signed the message (authenticity) and second, the message has not been modified during its transmission (integrity). Another important feature is that a signer cannot later deny having signed a message (non-repudiation).

1.2.2 Performance Metrics

There are various performance metrics that determine the suitability of signature schemes for certain applications. These include the size of the public and private key, the size of the signature, and, certainly, the computation time that is required by the algorithms to set up the key pair, sign the document, and verify the signature. Out of these performance metrics, the secret key size and the key generation time are arguably the least important for most applications, since the secret key will usually not be transferred and key generation happens only once and is most often not time critical. However, the requirements for signature schemes depend heavily on the application they are used in. In the context of software distribution for instance, the signature generation time is less important, because signing is usually performed only once per release. The signature, in contrast, is distributed to all users alongside the software, rendering the signature size and verification time more critical.

Even in a single application, various requirements can be set to the used signature schemes. We return to the already mentioned TLS protocol. Besides the handshake signature of the server, three more signatures appear in the certificate chain and another two are part of the signed certified timestamps (SCT). The server's signature during handshake is the only one that is generated online. It is created and verified only once, whereas the other signatures are verified by different clients. Furthermore, the public keys of the CA and the SCTs are not part of the handshake, but pre-distributed and included in the browsers or operating system's trusted log list [Wes24]. Thus, some signatures require a good balance of public key and signature size, as well as signing and verification efficiency, while others only demand short signatures and fast verification.

In Table 1.1 we present performance metrics of various classical and post-quantum signature schemes. We included the older, but widely deployed classical standards, the newly published post-quantum standards, and some of the current candidates in the call for additional signatures, that have a chance of being standardized in the coming years. What strikes immediately is that no post-quantum representative comes close to the all-round qualities of Ed25519.

Table 1.1: Performance metrics of standardized and on-ramp signature schemes. The style of the table is adopted from a similar table in the blog post *The state of the post-quantum Internet* [Wes24]. The numbers for the PQ signatures are taken from TII's pqsort webpage [TII25] and for the classical signatures we used benchmarks from supercop [ECR25].

Scheme	Family	in Bytes		in CPU Kcycles	
		public key	signature	sign	verify
Standardized classical schemes					
Ed25519	ECC	32	64	51	182
RSA-2048	RSA	256	256	3 317	51
Standardized PQC schemes					
ML-DSA-44	lattice-based	1 312	2 420	250	87
FN-DSA-512	lattice-based	897	752	667	159
SLH-DSA-128f	hash-based	32	17 088	29 181	2 399
Selection of on-ramp PQC signature schemes					
CROSS	code-based	54	9 120	2 737	1 862
HAWK	lattice-based	1 024	555	94	114
LESS	code-based	97 484	1 329	57 537	55 117
MAYO	multivariate	1 420	454	673	278
PERK	MPC-in-the-head	241	8 026	7 516	5 059
QR-UOV	multivariate	24 256	200	2 375	1 834
SNOVA	multivariate	1 016	248	468	254
SQIsign	isogeny-based	65	148	103 042	5 173
UOV	multivariate	66 576	96	106	119

Isogeny- and hash-based schemes have tiny public keys, but are discouragingly slow. Lattice-based schemes are quite efficient but feature medium sized public keys and signatures. Multivariate schemes offer smaller signatures compared to their lattice-based counterparts, but (mostly) suffer from large public keys. However, it can be expected that more signatures will be standardized and deployed in the coming years to diversify the list of underlying mathematical problems and to cover the wide range of applications more effectively.

1.2.3 Classical Signature Schemes

In the following we provide concrete examples of signature schemes and a brief overview of historical milestones in the development and standardization of digital signatures.

Shortly after Diffie and Hellman presented their famous key exchange and established the concept of public-key cryptography [DH76] in 1976, Rivest, Shamir, and Adleman developed the RSA cryptosystem [RSA78], which supports both, public-key encryption and digital signatures. RSA is based on the hardness of factoring large integers and despite its age, it is still widely deployed in many applications. It has been standardized in the PKCS#1 v1.5[3] by RSA Laboratories in 1993 and republished by the Internet Engineering Task Force (IETF) in RFC 2313. The document, of course, was subject to many changes over the years, with PKCS#1 v2.2 from 2012 being its latest version to date, which was republished by IETF in RFC 8017.

[3]PKCS is short for Public-Key Cryptography Standards. Earlier versions of the standard were only privately distributed, and not published officially.

Two other schemes that have shaped the landscape of digital signatures are the ElGamal signature scheme [Gam85] and Schnorr signatures [Sch89; Sch91], both relying on the discrete logarithm problem. These schemes played a major role in the development of DSA, which can be seen as a standardized variant of ElGamal signatures, that include conceptual influences of Schnorr signatures. DSA was proposed by NIST in 1991 and a few years later adopted as FIPS 186 in their Digital Signature Standard (DSS). Initially, it was the only supported algorithm in FIPS 186, before they included RSA in the first revision FIPS 186-1 of their standard in 1998. However, the popularity of DSA has declined sharply and it has been mostly superseded by its two successors ECDSA and Edwards-curve Digital Signature Algorithm (EdDSA). ECDSA is an analog of DSA based on the discrete logarithm problem for elliptic curves over finite fields. The use of elliptic curves in cryptography was analyzed and suggested independently by Koblitz [Kob87] and Miller [Mil85]. It is specified in ANSI X9.62 and approved by NIST from FIPS 186-2 onwards. The digital signature scheme EdDSA is a variant of Schnorr signatures based on twisted Edwards curves [BDL+11]. Its most popular representative is Ed25519, which uses a special elliptic curve with transparent parameter choices and fast arithmetic. It has been standardized by IETF in RFC 8032 and included in FIPS 186-5 by NIST.

In summary, the most widespread digital signature schemes are RSA, ECDSA, and EdDSA. All of them are vulnerable to Shor's algorithm and therefore considered insecure in the presence of sufficiently large quantum computers. Many leading entities have already announced that these algorithm will be declared as deprecated by 2030, and their usage will end 2035, at the latest.

1.2.4 Post-Quantum Signature Schemes

Driven by the need for quantum-resistant public-key algorithms, NIST initiated a process to evaluate and standardize one or more digital signature schemes and KEMs. In a call for proposals at the end of 2016, NIST was soliciting proposals for post-quantum cryptosystems that are based on mathematical problems, which are assumed to be hard to solve, even in the presence of large-scale quantum computers. Some of these cryptosystems have already been investigated by researchers for a long time, but their overall performance could not match the one of classical schemes. For instance, encryption schemes based on the hardness of decoding a general linear code were developed in 1978 and signature schemes based on the hardness of solving a system of multivariate quadratic equations were present since the late 1980s. Both problems are considered to be secure against quantum algorithms, but the derived cryptosystems usually suffer from large public keys.

The proclaimed standardization process was intended to deepen research in this area and ensure a transparent selection process to which the global research community in the field of cryptography contributed. At the end of 2017, 69 submissions of cryptographic algorithms were accepted as complete and proper. In a first evaluation round, the schemes were analyzed regarding three criteria: security, performance, and implementation characteristics. At the beginning of 2019, 26 of the 69 submissions advanced to the second round. The 26 submissions consisted of 17 public-key encryptions and nine signature schemes. Since this thesis focuses on signature schemes, we present these nine signature

schemes in Table 1.2 and sketch their further path in the standardization process.

Table 1.2: The nine signature schemes that advanced to round 2 of the NIST standardization process in 2019. The six schemes that advanced to round 3 are divided into finalists and alternate candidates. NIST expected to select one or more finalist for standardization at the end of the third round, while the alternate candidates were supposed to be further evaluated in a fourth round.

NIST PQC standardization process

Status	Signature scheme	Family	Standardized
Finalist in round 3	DILITHIUM	lattice-based	✓
	FALCON	lattice-based	✓
	Rainbow	multivariate	✗
Alternate candidate in round 3	GeMSS	multivariate	✗
	Picnic	MPC-in-the-head	✗
	SPHINCS$^+$	hash-based	✓
Eliminated at the end of round 2	LUOV	multivariate	✗
	MQDSS	multivariate	✗
	qTESLA	lattice-based	✗

As can be seen in Table 1.2, four of the signature schemes are based on multivariate quadratic equations: GeMSS, LUOV, MQDSS, and Rainbow. Out of these, LUOV and Rainbow are based on UOV. Lattice-based schemes were represented by DILITHIUM, FALCON, and qTESLA, while Picnic is based on the MPC-in-the-head paradigm and SPHINCS$^+$ is a hash-based signature. At the end of the second evaluation round in July 2020, LUOV was eliminated, since a cryptanalytic attack broke its security claims, and MQDSS and qTESLA were no longer considered for performance reasons. During the third evaluation round, in which DILITHIUM, FALCON, and Rainbow were highlighted as finalists, both GeMSS and Rainbow suffered from devastating attacks, excluding these schemes from further consideration. In July 2022, DILITHIUM, FALCON, and SPHINCS$^+$ were selected for standardization, and renamed ML-DSA, FN-DSA, and SLH-DSA.

As depicted in Table 1.1, the two lattice-based schemes offer good overall performance and are therefore considered as the most suitable general-purpose algorithms. The hash-based scheme is preferable in applications that prioritize conservative security design, since the only underlying security assumption is the existence of secure cryptographic hash functions, which are very well-studied. On the downside, SLH-DSA has large signature sizes and slow signing times, which makes it impractical for a wide range of applications. Thus, to avoid over-reliance on lattice-based signature schemes, a more diverse signature portfolio in terms of a broader range of underlying hardness assumptions was still—if not even more—desirable. Consequently, NIST announced to issue a new call for proposals for additional quantum-resistant digital signature schemes. In July 2023, 40 submissions were accepted as complete and proper. These include ten multivariate signature schemes, of which seven can be classified as UOV-based. Similar to the main PQC standardization project, the selection of additional digital signatures is organized in several evaluation rounds. At the end of

the first evaluation round in October 2024, 14 candidates were announced to advance to round 2. We list these schemes in Table 1.3.

Table 1.3: The 14 signature schemes that advanced to round 2 of NIST's call for additional digital signatures.

NIST PQC additional digital signatures schemes

Algorithm	Family	Algorithm	Family
CROSS	code-based	HAWK	lattice-based
LESS		Mirath	
SQIsign	isogeny-based	MQOM	
MAYO		PERK	MPC-in-the-head
QR-UOV	multivariate	RYDE	
SNOVA	(UOV-based)	SDitH	
UOV		FAEST	symmetric

Three of the seven UOV-based schemes got eliminated: PROV, TUOV, and VOX. The status report by NIST only covered schemes that advanced to the next round and lacked reasoning for the exclusion of candidates. From the discussions during the first evaluation round, we assume that the attack presented in [FI23] led to the elimination of VOX, and the advantages that PROV and TUOV pose over UOV were not considered vital enough. The four schemes that advanced to the second round are MAYO, QR-UOV, SNOVA, and UOV. Thus, together with the MPC-in-the-head signatures, UOV-based signatures form the largest group of the second round candidates.

The remaining schemes offer different advantages and disadvantages regarding performance metrics and security foundations. It will be intriguing to see in which direction the standardization process will develop. NIST plans to select finalists for a third evaluation round in 2026. In the meantime, the cryptographic research community is challenged to enhance the knowledge about candidate algorithms, e.g., to work on security claims, develop optimized implementations, analyze their suitability for constrained platforms, and investigate their physical security.

1.3 Physical Security

We have not yet defined what it means for a signature scheme to be secure and we will return to this question in Section 2.1.2. Nevertheless, it is clear that an adversary should not be able to compute the secret key efficiently from publicly available data. In this case, anyone could impersonate the actual owner of the secret key and sign messages in their name. Since all the mentioned signature schemes rely on a certain mathematical problem, this means there should not exist an efficient number theoretic or algebraic algorithm to compute a secret key from the corresponding public key or certain message-signature pairs. This property is sometimes phrased as *mathematical security*, if one wants to emphasize that only the underlying mathematical model is considered, while physical aspects are not regarded.

However, cryptographic algorithms are executed on real hardware devices, which makes it necessary to also consider physical effects. The devices running

the algorithms have a specific power consumption and execution time, which not only depends on the executed operations, but also on the processed data. Furthermore, they emit electromagnetic radiation, residual heat, and even low sound emissions. The precise characteristic of these effects depends on several parameters, but also on the exact values of the secret key. The term side-channel attack (SCA) describes the approach to passively gather these types of information from a target device, analyze them with statistical tools, and exploit the gained information about intermediate values or certain key bits to realize efficient key-recovery methods, which depend on the implemented cryptographic algorithm. On the other hand, an adversary could also attempt to manipulate the device to make it behave incorrectly. There are various strategies to disrupt the normal operation of a device or algorithm, like clock and voltage glitching, laser fault injection, temperature attacks, and electromagnetic fault injection. These types of active attacks are known as fault injection attacks (FIA).

Including the previously mentioned strategies into the attacker model might increase the amount of information an adversary has at hand to retrieve secret data. The ability of cryptographic algorithms and especially their implementation to resist attacks, which are based on the aforementioned physical aspects, is termed *physical security*.

Even though these approaches may appear somewhat futuristic at first glance, it was already proven in the late 1990s, that secrets could be extracted from devices like smart cards [BDH+97; BDL97; MDS99]. In the 2000s, interest in these methods increased sharply and various techniques for their application were developed, even with low-cost equipment [BMM00; BS03; GMO01; SA02]. At the latest since the 2010s, physical security became a serious design requirement in cryptographic implementations, since researchers demonstrated a multitude of realistic attacks in various environments. We elaborate closer about the history and background of physical attacks in Section 2.3. In the following, we shortly present a selection of attacks that attracted a lot of attention in recent years, underlining the real-world relevance of secure implementations.

Side-channel attacks. Lucky13 [AP13] is a timing side-channel attack on padding during the decryption of encrypted TLS records. The authors noticed, that for certain padding length, i.e., 13 bytes, there are tiny time differences in the decryption, which leak information about the plaintext and enable them to recover it incrementally. It allowed the attacker to restore session cookies, passwords, or sensitive data in TLS versions that worked with Cipher Block Chaining (CBC)-mode ciphers. It was prevented entirely by switching to authenticated encryption with associated data (AEAD) ciphers in TLS 1.3 and can also be mitigated with constant-time decryption and padding checks.

The Raccoon attack [MBA+21] exploits a vulnerability in the TLS key exchange when using DH or ECDH. The adversary queries the server with crafted messages and is able to reveal the most significant bits of the private key from timing variations or validation errors. The key can be recovered bit-by-bit by adapting the crafted messages in an iterative procedure. The attack compromises the server's secure communication. It was demonstrated on real-world TLS implementations (including OpenSSL) and affected the versions TLS 1.2 and earlier. TLS 1.3 is protected against this attack, since it uses ephemeral key exchanges by default.

The authors of Hertzbleed [WPH+22] showed how to exploit power side-channel information via timing variations without having physical access to the device. They noticed that modern Central Processing Unit (CPU) dynamically adjust clock frequency based on power consumption and observed that secret-dependent computations can affect CPU frequency due to subtle variations in power usage. This has an impact on execution time, even if the cryptographic operations are implemented in constant-time. Hertzbleed bridges a gap between power side-channels, which are typically local, and remote timing attacks. It affects CPUs from Intel and AMD and can be mitigated by avoiding data-dependent execution patterns.

There are other famous micro architectural attacks, like Spectre [KHF+19], Meltdown [LSG+18], and CrossTalk [RMR+21], which indicate that even certain kernel and core isolation techniques cannot guarantee complete security against cache-timing attacks. These attacks affected a variety of processors and had a large impact on secure software and hardware design.

Fault attacks. Rowhammer [KDK+14] is a fault attack that exploits a weakness in Dynamic Random-Access Memory (DRAM) modules. By repeatedly accessing (hammering) a row of memory (the aggressor row) one can cause bit flips in adjacent rows (the victim rows), even without accessing them directly. The frequent activation causes electromagnetic interference, such that capacitors of the memory cells might exceed or fall below a certain threshold, which leads to a bit flip. The Rowhammer attack demonstrates a hardware vulnerability that could be triggered completely from software. It has a large impact, since it leads to arbitrary memory corruption and breaks process and memory isolation. Possible solutions are probabilistic row refreshing or stronger memory isolation policies on a hardware level.

In Plundervolt [MOG+20], the authors managed to undervolt the CPU inside an Intel Software Guard Extensions (SGX) enclave,[4] which causes faults during sensitive operations (e.g., AES encryption). The induced faults could be exploited by Differential Fault Analysis (DFA) techniques to recover secret AES keys. Similar to Rowhammer, this attack works without physical access, thereby representing another software-based fault injection attack. To prevent the Plundervolt attack, Intel released code updates to lock voltage controls when SGX is active, which disabled undervolting via software. VoltPillager [CVM+21] builds on Plundervolt, but utilizes a hardware tool to control the CPU's voltage externally, thereby bypassing the countermeasure and update introduced by Intel. This shows that undervolting attacks are still viable with physical access and hardware-level tools. The mitigation of such attacks requires hardware redesign with fault-tolerant ot tamper-resistant future enclaves.

1.4 Research Topics in this Thesis

The research focus of this thesis lies on the security of UOV-based signatures. They belong to the family of multivariate cryptographic schemes, build an important branch of signatures, and provide promising candidates for efficient,

[4] Intel SGX is a hardware-based security technology that creates isolated memory regions called enclaves within an Intel processor.

quantum-resistant signature schemes that might even be selected for standardization in NIST's call for additional digital signatures. On their way to practical deployment, there is still a great need for research regarding their security and efficiency. This thesis contributes to a better understanding of the physical security of UOV-based signature schemes and, additionally, analyzes several of them against advanced security notions. We divide this thesis into three main chapters.

Developing physical attacks. The first part, featured in Chapter 3, presents physical attacks against several multivariate schemes, namely Rainbow, UOV, and MAYO. The development of new physical attacks is necessary to reveal vulnerabilities in implementations of cryptographic algorithms. The presented attacks comprise various stages, such as identifying vulnerable code lines that might reveal useful intermediate values, and exploiting the gained information with algebraic techniques to recover the secret key. Furthermore, the attacks are either simulated or practically executed on devices that carry the architecture of targeted microprocessors, in order to demonstrate that there is indeed a need to apply countermeasures and protect the elaborated code lines. In this chapter we present both SCA and FIA. They are completed by the suggestion of tailored countermeasures, that mitigate these specific attacks.

Enhancing implementation security. The second part is dedicated to harden various UOV-based schemes against physical attacks and is given in Chapter 4. Here, we focus on the integration of developed countermeasures to cover a wide range of possible vulnerabilities. Even though several physical attacks have been developed against UOV and its variants, the discussion for countermeasures remained merely theoretical and existing implementations are not hardened against side-channel and fault attacks. We take first steps towards more secure implementations by presenting the first masked versions of UOV and MAYO, the two signature schemes that exhibit the most advanced implementations of the UOV-based schemes. Furthermore, we provide a more comprehensive overview of vulnerabilities, by transferring existing attacks on individual schemes to UOV, MAYO, QR-UOV, and SNOVA, i.e., the schemes that advanced to the second round of the NIST PQC standardization effort for additional digital signatures.

Analyzing security features. The final part, presented in Chapter 5, deals with security features of UOV-based signature schemes. Our analysis is twofold. First, since the basic UOV signature scheme suffers from large public keys, there have been several attempts to reduce the public key size by introducing modifications. Hence, it is necessary to check if the tweaked schemes are still secure. Hereby, we present a security analysis of MQ-Sign, a variant of UOV that employs very sparse coefficient matrices to reduce the amount of coefficients that need to be stored. We introduce a key-recovery attack and a forgery attack on different variants of MQ-Sign, which feature different sparsity levels. Second, we analyze seven UOV-based signatures regarding advanced security features that go beyond unforgeability. In the rapidly growing field of applications of cryptography, there is a large variety of protocols tailored to specific use cases. Protocol developers might use signature schemes in a way they are not designed

for, such that the standard security guarantees may not be sufficient anymore. Therefore, it is desirable that signature schemes fulfill certain security notions like exclusive ownership (EO), non resignability (NR), and message bound signatures (MBS). We investigate these properties for UOV, MAYO, QR-UOV, SNOVA, PROV, TUOV, and VOX.

1.5 Contributions

In this section, I present a detailed description of my contributions to the three research areas, classified in Section 1.4, as part of my dissertation. To this end, I give a thorough summary of our related research papers and focus on my share of the ideas and work packages within them. In Chapter 3, 4, and 5, I provide more context to these research papers and elaborate on how our contributions fit into the research areas. The corresponding publications can be found in Appendix A, B, and C, respectively.

1.5.1 Developing Physical Attacks

The publications [**A**CK+23; **A**KK+22; **A**MS+24] contribute to the field of physical attacks against UOV-based signature schemes.

Together with Kovats, Krämer, and Marzougui [**A**KK+22], we develop two fault attacks against Rainbow, a multi-layered version of UOV and finalist in the NIST PQC Standardization Process. The idea for the first fault attack was already present in the literature [KL19; SK20], but we managed to improve the complexity of the algebraic post-processing significantly. We introduced a cryptanalytic method that allows us to reveal the secret key just by solving linear equations, after the fault injection attack was executed successfully. Before that, the complexity to complete the fault attack was estimated to be around 2^{38} to 2^{42} for the security level I parameters. Our approach works in polynomial-time and is efficiently applicable to the parameter sets across all security levels. The second fault attack is new and targets an operation that has previously not been the target of a fault attack. It works in complexity $\mathcal{O}(q^{v_1-o_1})$, which is as small as 2^{16} for the security level I parameter set. Both fault attacks require only a limited amount of faulted signatures in the range from a few to several dozens, depending on the concrete parameter set. Additionally, we verify both attacks on an emulated ARM M4 architecture. Thus, the compiled binary is executed as it is running on a real signing device and we skip the relevant instruction of the assembly code. Moreover, we specify countermeasures for both attacks.

I developed the ideas for the improved algebraic post-processing and the second fault attack under the supervision of Krämer. To support the theoretical observations, which formed the basis of the attacks, I implemented a Sagemath script that confirmed the complexity claims of the attack. The attack simulation was implemented by Kovats. The countermeasures were developed jointly by all co-authors.

In joint work with Marzougui, Seifert, and Ulitzsch [**A**MS+24], we assessed the practicality of a fault injection attack on MAYO, a compact UOV-based signature scheme, that is a promising candidate in NIST's call for additional digital signatures. The paper discusses two variations of a FIA, where the

critical values are either reused or set to zero, as a consequence of the fault injection attack. In both cases only a single faulted signature is sufficient to achieve key-recovery in a matter of seconds. We execute the attack on an STM32F4 target board using a ChipWhisperer Lite. We provide all attack parameters to allow a convenient reproduceability of the given attacks, as well as a Sagemath implementation, which computes the secret key from the faulted signature and the public key.

For this research work, I contributed the idea for the attacks and provided the mathematical background for an efficient key-recovery attack. Furthermore, I supported Marzougui, who was working on the practical execution of the attack, and Ulitzsch, who verified the algebraic attacks with Sagemath simulation scripts. Both were working under the supervision of Seifert in this project.

Joint with Campos, Krämer, Samardjiska, and Stöttinger [**A**CK+23], we developed an end-to-end profiled side-channel attack against UOV, which is also a candidate in NIST's call for additional digital signatures. We were the first to target the inversion of the central map during signature generation. In more detail, we exploit a correspondence of the public and secret key, which helped us to recover the (secret) vinegar vector by means of a simple power analysis. We chose a profiling approach that allowed us to mount a single-trace attack, i.e., we only need a single attack trace taken during the signature generation by the target device. The power traces for the templates are conducted by executing the attacked subroutine with self-chosen input vectors on a similar device beforehand. This is quite crucial, since the targeted vinegar variables are generated randomly and vary across different signing procedures. From the vinegar vector and the generated signature, we deduce an oil vector, which enables an efficient algebraic key-recovery attack to determine the complete oil space. Thus, we recovered the core of the secret key, which allows to create signatures for arbitrary messages, just as the legitimate signer would compute them. We performed all steps of the introduced side-channel attack. We conducted power measurements with the ChipWhisperer Lite on a STM32F3 target board, identified regions of interest, and implemented a correlation analysis, which provided us with the most likely values for the targeted vinegar vector. Furthermore, we implemented the modified Kipnis-Shamir attack, which efficiently revealed the complete oil space upon this input. The work is concluded by an elaborate discussion of general and scheme-specific countermeasures.

The idea for the side-channel attack was developed in joint work between the co-authors. I identified the vulnerable operation and noticed the public and secret key correspondence that increased the efficiency of the attack under the supervision of Krämer. Together with Campos and Stöttinger, I conducted the power measurements and analyzed the traces. Samardjiska and I worked jointly on the implementation of the algebraic attack.

1.5.2 Enhancing Implementation Security

The goal of the publications [**A**CK25; SM**A**+24] was to harden implementations of UOV-based signature schemes, such that they exhibit more resistance against physical attacks.

In [SM**A**+24], we present a hardware implementation of the MAYO signature scheme. We developed a reconfigurable hardware design compatible with various Field Programmable Gate Array (FPGA) architectures and adaptable

to different security levels. Certain key modules, like the matrix-vector multiplication and Gaussian solver, were optimized with respect to the total memory consumption. The implementation was tested on the Zynq Zedboard with a Zynq-7020 SoC. Furthermore, we investigated the physical security of MAYO against side-channel and fault-injection attacks, and implemented lightweight countermeasures. Power analysis methods like [**A**CK+23; PSKH18; PSN21] are mitigated by performing the matrix vector multiplication in a shuffled fashion, instead of treating the same secret values in blocks, before hopping to the next one. First-order fault attacks like [**A**KK+22; KL19; SK20] are prevented by adding certain double checks and check sums at specified parts of the implementation.

Sayari took care of the hardware implementation, while Marzougui and I supported him with background information about MAYO, and digital signature schemes, in general. Together with Marzougui, I performed the analysis of potential vulnerabilities and constructed dedicated countermeasures. The implementation of the countermeasures was again realized by Sayari. The project was carried out under the supervision of Krämer and Seifert, who guided us with strategic advice.

Together with Krämer and Campos, we created an extensive overview of vulnerabilities against physical attacks against UOV-based signature schemes in [**A**CK25]. First, we began by reviewing existing side-channel and fault attacks on various—also possibly broken or deprecated—UOV-based schemes, updating these assessments to align with the current UOV specifications, which have evolved over recent years. Moreover, we introduce new physical attacks to create a more complete picture of potential vulnerabilities. Our study also examines how these attacks apply to related schemes like MAYO, QR-UOV, and SNOVA. We selected the four signature schemes that advanced to the second round of NIST's call for additional signatures, as they have the chance of being standardized in the upcoming years. To enhance resistance against physical attacks, the paper highlights that certain implementation choices, such as key compression techniques and randomization methods, significantly impact physical security, particularly concerning the effectiveness of fault attacks. Furthermore, we provide implementations of UOV and MAYO for the ARM Cortex-M4 architecture, incorporating first-order masking and protections against selected fault attacks. We benchmark the performance overhead on a NUCLEO-L4R5ZI board and validate our approach through test vector leakage assessment (TVLA), observing significantly reduced t-values in the protected subroutines. The power traces for the TVLA were taken with the ChipWhisperer Lite on a STM32F4 target board.

Krämer initiated the project and had the idea of providing an extensive literature overview. I suggested to incorporate dedicated countermeasures to achieve a first assessment of the expected overhead for first-order masking in UOV-based schemes. Krämer and I prepared the literature survey of existing attacks. The transfer of the physical attacks across the schemes was done mostly by myself, with the support of Krämer. The main part of the implementation of countermeasures was done by Campos, while we worked together regarding scheme-specific and strategic aspects. The TVLA was realized in joint work with Campos.

1.5.3 Analyzing Security Features

With the publications [**A**DM+24; **A**ST24], we assess the security of various UOV-based signature schemes against different security notions.

Together with Samardjiska and Trimoska, we developed attacks against the UOV-based signature scheme MQ-Sign, breaking its existential unforgeability under chosen message attack (EUF-CMA) security. MQ-Sign, proposed by Shim, Kim, and An [SKA22] as a candidate in the first round of South Korea's post-quantum cryptography competition (KpqC), comes in four variants: MQ-Sign-SS, MQ-Sign-RS, MQ-Sign-SR, and MQ-Sign-RR, with MQ-Sign-RR being equivalent to standard UOV. First, we present a polynomial-time key-recovery attack against MQ-Sign-SS and MQ-Sign-RS. Second, we describe a forgery attack against MQ-Sign-SR with reduced exponential running time. The complexity estimates show that the variant falls indeed short of the claimed security level. We provide a verification script for the key-recovery attack, which reconstructs the secret key in less than seven seconds, even for security level V. The complexity of the forgery attack is composed of an enumeration step, and the process of solving a smaller system of quadratic equations. We implement the non-trivial system solving part to reinforce the complexity claims.

While I provided the initial idea for the key-recovery attack on two of the variants, all co-authors worked on the details in joint work, and its implementation was developed by Trimoska. The forgery attack was joint work with Samardjiska and Trimoska, while I was visiting the research group at Radboud University. The implementation was again realized mainly by Trimsoka, with the support of Samardjiska and myself.

In joint work with Düzlü, Meyer, Struck, and Weishäupl [**A**DM+24], we analyzed the Beyond UnForgeability Features (BUFF) security of seventeen signature schemes based on codes, lattices, isogenies, and multivariate equations, that were submitted to NIST's call for additional digital signatures [NIST22]. BUFF security addresses vulnerabilities related to maliciously generated keys, ensuring properties such as EO, MBS, and NR. The results vary across the whole spectrum, from schemes that achieve neither notion (e.g., WAVE) to schemes that achieve all notions (e.g., CROSS). Inspired by the results of the analysis for HAWK and PROV, which satisfy all BUFF properties, by essentially using the PS-3 transform, we investigated whether this transform suffices for the other schemes as well.

Motivated by NIST's declaration of BUFF security as a desired feature in the additional call for digital signatures, the idea to analyze the candidates was jointly elaborated. The analysis of the seventeen selected signature schemes was divided among all co-authors according to their expertise and resources. I participated mainly in the analysis of the code-based and multivariate schemes. The insights we deduced from the result and conclusions we were able to draw, were developed in joint work.

2

Background

This chapter presents the background for the main topics of this thesis and is organized as follows. First, we provide more details about signature schemes in general. This includes their definition and the properties that let them achieve the security goals we stated in Chapter 1. Furthermore, we briefly introduce classical signature schemes, which are already used in practice, and post-quantum signature schemes, which have emerged in the recent years as a result of the PQC standardization process hosted by NIST. Subsequently, we give insights into multivariate signatures, show different methods of how to construct signature schemes from the MQ-Problem, and give a detailed description of the UOV-based signature schemes UOV and MAYO. To conclude this chapter, we discuss the history and working principle of physical attacks. These can be divided into fault-injection and side-channel attacks.

2.1 Signature Schemes

In Section 1.2 we elaborated on the applications of digital signatures and their deployment in certain cryptographic protocols. In this section, we will briefly introduce the technical definition and properties of signature schemes. Furthermore, we give some examples of classical signature schemes that have been used in real-world protocols for many years and provide insights into the standardization process of quantum-resistant signature schemes.

2.1.1 Definition

A signature scheme Σ consists of three efficient algorithms:

KeyGen: The key generation algorithm $\mathsf{KeyGen}(1^\lambda)$ generates a pair of two keys. As input, it gets a security parameter λ, which determines the strength of the keys. The output is a secret key sk along with a public key pk.

Sign: The signing algorithm $\mathsf{Sign}(\mathsf{sk}, \mathsf{msg})$ is used to generate a signature sig for a given message msg. Accordingly, it gets a secret key sk and a message msg as input and outputs a signature sig, which serves as proof that msg was signed by the owner of sk.

Verify: The verification algorithm $\mathsf{Verify}(\mathsf{pk}, \mathsf{msg}, \mathsf{sig})$ is used to check the validity of a signature. It takes as input a public key pk, a message msg, and a signature sig, and it outputs a boolean value v, which is true if the signature is valid and false otherwise.

2.1.2 Properties

The two main properties of a signature scheme are correctness and security.

Correctness. Correctness follows the intuition that an honestly generated signature, obtained by using the secret key, should verify under the corresponding public key. The formal definition is straightforward. A signature scheme is correct if, for any key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, we have

$$\mathsf{Verify}(\mathsf{pk}, \mathsf{msg}, \mathsf{Sign}(\mathsf{sk}, \mathsf{msg})) = \mathsf{true}$$

for every possible message msg. This property ensures that the signature scheme works as intended under normal conditions and honest users.

Security. The definition of security requires more attention. Naturally, it should not be feasible to compute the secret key from public information. These kind of attacks are called key-recovery attacks. Moreover, it is clear that only the party in possession of the secret key should be able to generate valid signatures for a given message. This property is termed *unforgeability* and attacks against this feature are dubbed forgery attacks. There are different threat models that vary in the amount of information a potential adversary has at hand. Is the adversary given only the public key (key-only attack), or also valid signatures for a variety of messages (known message attack)? In an even stronger threat model (adaptive chosen message attack), the attacker is allowed to learn signatures on arbitrary messages of its own choice. The standard security notion EUF-CMA follows the latter approach and requires the adversary to find one valid message-signature pair $(\mathsf{msg}, \mathsf{sig})$, for a message msg that has not been signed by the legitimate signer before. To make this more concrete, consider the following experiment [KL14] for an adversary $\mathcal{A}$:

The signature experiment $\text{Sig-forge}_{\mathcal{A},\Sigma}(\lambda)$ is defined by:

1. $\mathsf{KeyGen}(1^\lambda)$ is run to obtain keys $(\mathsf{pk}, \mathsf{sk})$.
2. The adversary $\mathcal{A}$ is given pk and access to an oracle $\mathsf{Sign}(\mathsf{sk}, \cdot)$ that outputs valid signatures for messages of its choice. Let $\mathcal{Q}$ denote the set of all queries that $\mathcal{A}$ asked its oracle. The adversary then outputs $(\mathsf{msg}, \mathsf{sig})$.
3. $\mathcal{A}$ succeeds if and only if $\mathsf{Verify}(\mathsf{pk}, \mathsf{msg}, \mathsf{sig})$ and $\mathsf{msg} \notin \mathcal{Q}$.

A signature scheme $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ is EUF-CMA, or just secure, if the probability that an adversary succeeds in the signature experiment $\text{Sig-forge}_{\mathcal{A},\Sigma}(\lambda)$ is negligible.

Even though EUF-CMA security is the standard security notion for digital signatures, there exist other advanced security notions that are relevant in practice. These are especially interesting, when participating parties act maliciously, i.e., they do not stick to honestly generated key pairs, but try to

manipulate the public key, such that certain signatures seem valid. We will define and discuss these advanced notions in Section 5.2.

2.2 Multivariate Signatures

We indicated in Section 1.1 that multivariate signatures are based on the hardness of finding a solution to a system of multivariate quadratic equations. We will elaborate now, how one constructs a signature scheme upon this problem and focus especially on the oil-and-vinegar approach in this section.

Let m be the number of equations, n the number of variables, and $\mathbb{F}_q$ the finite field with q elements. Each quadratic equation over $\mathbb{F}_q$ is defined by a quadratic polynomial $\mathcal{P}^{(k)} : \mathbb{F}_q^n \to \mathbb{F}_q$ with

$$\mathcal{P}^{(k)}(x_1, \ldots, x_n) = \sum_{1 \leq i \leq j \leq n} \gamma_{ij}^{(k)} x_i x_j + \sum_{i=1}^{n} \beta_i^{(k)} x_i + \alpha^{(k)},$$

where $k \in \{1, \ldots, m\}$ and $\gamma_{i,j}^{(k)}, \beta_i^{(k)}, \alpha^{(k)} \in \mathbb{F}_q$ represent the quadratic, linear, and constant coefficients of this equation. Together, these m polynomials constitute the multivariate quadratic map $\mathcal{P} = (\mathcal{P}^{(1)}, \ldots, \mathcal{P}^{(m)}) : \mathbb{F}_q^n \to \mathbb{F}_q^m$, which represents the system of quadratic equations. Solving a system of equations can be translated to finding a solution vector $\mathbf{s} \in \mathbb{F}_q^n$ given a target vector $\mathbf{t} \in \mathbb{F}_q^m$, such that $\mathcal{P}(\mathbf{s}) = \mathbf{t}$ holds. Already for m and n at moderate size, i.e., even around 50 or 60, this becomes an infeasible task for a random map $\mathcal{P}$ without any additional information.

Constructing signature schemes. There are two main concepts on how to construct a signature scheme upon this problem.

1) One can compute a pair $(\mathbf{s}, \mathbf{t}) \in \mathbb{F}_q^n \times \mathbb{F}_q^m$ with $\mathcal{P}(\mathbf{s}) = \mathbf{t}$, by just randomly sampling $\mathbf{s}$ and evaluating it under $\mathcal{P}$ to obtain $\mathbf{t}$. It is possible to develop a zero-knowledge proof of knowledge of a solution $\mathbf{s}$ to the multivariate quadratic system $\mathcal{P}$ with target value $\mathbf{t}$. Proposals for those zero knowledge proofs are given in [SSH11] and [Beu20]. They can be used as a tool to build identification schemes, and thus, can be turned into a signature scheme with the well-known Fiat-Shamir transform, introduced in [FS86]. Two examples for such signature schemes based on the MQ-problem are MQDSS [CHR+16] and Mudfish [Beu20].

2) One can install a trapdoor into $\mathcal{P}$, which allows to efficiently compute solutions to given target vectors. To this end, the map $\mathcal{P}$ is usually derived as the composition of several maps, i.e. $\mathcal{P} = S \circ \mathcal{F} \circ T$. Here, the so-called central map $\mathcal{F} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ is also a multivariate quadratic map, but one with a certain structure that allows to invert $\mathcal{F}$ efficiently. The maps S and T, however, are affine or linear bijections, introduced to hide the structure of $\mathcal{F}$. To obtain a signature scheme, one can use a hash function $\mathsf{H} : \mathbb{F}_q^* \to \mathbb{F}_q^m$ to map a certain message into the target space $\mathsf{H}(\mathsf{msg}) = \mathbf{t} \in \mathbb{F}_q^m$, which is also the image space of the public map $\mathcal{P} = S \circ \mathcal{F} \circ T$. The signer, who has knowledge of the secret trapdoor, i.e., the individual maps $\mathcal{P}$ is composed of, can invert these maps one after the other to compute a preimage $\mathbf{s}$. To check if $\mathbf{s}$ is indeed valid, the verifier simply has to compute $\mathcal{P}(\mathbf{s})$ and check whether $\mathcal{P}(\mathbf{s}) = \mathbf{t}$ actually holds.

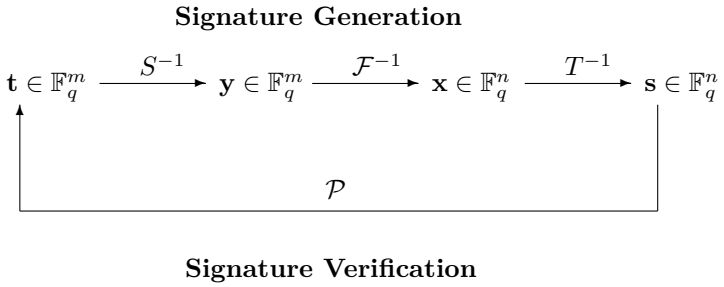


Figure 2.1: General workflow of trapdoor-based multivariate signature schemes.

Here, the map $\mathcal{P}$ is publicly available. It is commonly called the public map $\mathcal{P}$. This procedure is sketched in Figure 2.1.

The inverses of S and T can be computed efficiently, so the main task here lies in the inversion of $\mathcal{F}$. There are several options on how to define the central map $\mathcal{F}$ such that finding preimages $\mathbf{y}$ with $\mathcal{F}(\mathbf{y}) = \mathbf{x}$ for given $\mathbf{x}$ becomes feasible. One example is the HFE approach, where the central polynomial is defined as a univariate polynomial over an extension field $\mathbb{F}_{q^n}$. This allows a signer with knowledge of this hidden representation of the polynomial to efficiently invert the system. There exists a variety of schemes based on the HFE approach, but most of them got broken eventually. The most known representative is GeMSS, a highly efficient signature scheme, which was broken by the attack of Tao et al. [TPD21]. In this thesis, however, we focus on the OV approach, first introduced by Patarin [Pat97].

The Oil-and-Vinegar (OV) approach. The key idea of this approach to enable fast inversion of $\mathcal{F} : \mathbb{F}_q^n \to \mathbb{F}_q^m$, is to split the input variables $\mathbf{x} = \{x_1, \ldots, x_n\}$ into two sets, the vinegar variables $\{x_1, \ldots, x_v\}$ and the oil variables $\{x_{v+1}, \ldots, x_n\}$, and not allow $\mathcal{F}$ to have any quadratic terms in the oil variables. Thus, the polynomials of such a map $\mathcal{F} = (\mathcal{F}^{(1)}, \ldots, \mathcal{F}^{(m)})$ are defined by

$$\mathcal{F}^{(k)}(\mathbf{x}) = \sum_{1 \le i \le n} \sum_{1 \le j \le v} \gamma_{i,j}^{(k)} x_i x_j + \sum_{i=1}^{n} \beta_i^{(k)} x_i + \alpha^{(k)} \tag{2.2.1}$$

for $k \in \{1, \ldots, m\}$. Here, v denotes the number of vinegar variables. Note that every polynomial $\mathcal{F}^{(k)}$ becomes linear if one fixes these v vinegar variables to arbitrary values. In doing so, one can turn $\mathcal{F}$ into a system of m linear equations, in the remaining $n - v$ oil variables. By selecting parameter, s.t. $n - v = m$, the derived linear system has a solution with large probability. The solution vector $\mathbf{x}$ to the equation $\mathcal{F}(\mathbf{x}) = \mathbf{y}$ is then composed of the randomly generated v vinegar variables, and the m oil variables that constitute the solution of the derived linear system with m variables in m equations.

Remark 2.1. Let O' be the m-dimensional vector space that contains all the vectors whose first v entries are zero, i.e., $O' = \{\mathbf{x} \in \mathbb{F}_q^n \mid x_i = 0 \text{ for all } i \le v\}$. It is obvious that the quadratic part of $\mathcal{F}$ vanishes on this subspace O', since

each quadratic monomial contains one of these zero variables. Therefore, the quadratic part of $\mathcal{P}$ vanishes on the space $O := T^{-1}(O')$, where $T : \mathbb{F}_q^n \to \mathbb{F}_q^n$ is the secret invertible transformation from above, introduced to hide the structure of the central map $\mathcal{F}$. This subspace O is also known as the oil space and will play a major role in the upcoming description of the schemes and attacks in this thesis, since its knowledge allows anyone to sign arbitrary messages. Consequently, the oil space O constitutes the 'real' secret of signature schemes, that follow the OV approach.

Kipnis-Shamir attack on balanced OV. In the initial presentation of the OV approach [Pat97], the number of vinegar variables v was chosen equally large as the number of oil variables m, i.e., $n = v+m = 2m$. This (in hindsight) lead to the name balanced OV. Soon after, Kipnis and Shamir [KS98] found an attack against this scheme, making use of exactly this property. They noticed that certain matrices, which can be derived directly from the public key, map the oil space $O \subset \mathbb{F}_q^n$ onto itself. This set of matrices, which have O as a common invariant subspace, is sufficiently large, so that the authors were able to present two efficient algorithms for detecting the invariant subspace solely from the matrices derived from the public key. As a result, the attack presented in [KS98], also known as the *Kipnis-Shamir attack*, constitutes a polynomial-time key recovery attack against the balanced instance of OV schemes. For more details on the utilized set of matrices and the algorithms to detect invariant subspaces, we refer to [KS98].

Unbalancing the number of oil and vinegar variables. A crucial condition for the Kipnis-Shamir attack to work so efficiently is that the oil variables make up for at least half of the total variables, i.e., $2m \geq n$. Kipnis et al. [KPG99] suggest to use more vinegar than oil variables, i.e., $v > m$, leading to $2m < m + v = n$. Then the matrices mentioned above do not have O as an invariant subspace anymore, preventing the polynomial-time attack from above. Instead, they might have an invariant subspace, which is a subspace of O itself. The probability for this to happen is related to the difference $d = v - m$ of the number of vinegar and oil variables. The authors estimate the complexity of this adapted attack as proportional to $q^{d-1}n^4$, rendering it exponential in d. Thus, they choose v sufficiently larger than m, dubbing the modified scheme UOV, short for Unbalanced Oil-and-Vinegar. While the authors initially suggest to use $v \geq 2m$, current implementations employ $v \approx 1.5m$ or $n = 2.5m$, to reduce the number of total variables, since this still gives enough security margin against the Kipnis-Shamir attack [BCD+23].

UOV is known for its fast signing and verification algorithm, but suffers from large expanded public and secret keys. With compression techniques, it is possible to reduce the size of the secret key tremendously (at the cost of a less efficient signing algorithm) and the public key to some extent. We will present concrete performance numbers of the different variants in the following section. However, compared to classical or lattice-based signature schemes, the public key size is still enormous, even in its compressed variant. Not least for this reason, a considerable number of modifications was invented to aim for improvements of UOV.

UOV-based signature schemes. Two prominent examples of such signature schemes that were submitted to the NIST PQC standardization process in 2017, are LUOV and Rainbow.

LUOV [BPSV19] was a round 2 candidate, but got eliminated before round 3. The field lifting approach used in LUOV was developed by Beullens and Preneel in [BP17]. The idea is to generate the keys over the field $\mathbb{F}_2$ to achieve small keys, but to lift them to a larger extension field $\mathbb{F}_{2^7}$ or $\mathbb{F}_{2^{47}}$, where finding solutions to the MQ Problem is harder. Ding et al. [DDS+20] exploited the 'lifted' structure and developed a new method called the Subfield Differential Attack, with complexity estimates below the targeted security levels. The attacks could be countered by adapting the parameter sets, without affecting the efficiency of the scheme too heavily. Nevertheless, NIST did not select LUOV for the third round, since, in their opinion, the field lifting innovation has not been analyzed enough.

Rainbow [DCP+20] was even one of the finalists in round 3 of the NIST PQC standardization process. Ding and Schmidt [DS05a] constructed Rainbow as a multi-layered version of UOV to increase its efficiency and decrease its key sizes. The layers are solved individually following the oil-and-vinegar principle, and the solution is inserted into the next layer. While the authors initially suggested to employ five layers, it soon became clear that only two layers make for the most efficient instantiation of the scheme. In this fashion, Rainbow was submitted to the standardization process in 2017, as it was considered to be a more efficient variant of UOV. However, Beullens [Beu22] eventually managed to exploit this layer structure and mounted an efficient key-recovery attack, destroying all benefits Rainbow had over UOV. Consequently, it was not chosen for standardization and eliminated from the process.

More UOV-based signature schemes were submitted to NIST's call for additional digital signatures in 2023, namely MAYO [BCC+23], PROV [GCF+23], QR-UOV [FIH+23], SNOVA [WCD+23], TUOV [DGG+23], VOX [PCF+23], and UOV [BCD+23] itself. MAYO was developed by Beullens in [Beu21b]. It employs an oil space of considerable, smaller dimension compared to UOV. To compensate for that, the public key map gets 'whipped-up' during signature generation, to ensure that solutions to the linear system can still be found efficiently. This brings another security assumption into play, which the author calls the Multi-Target Whipped MQ Problem. At the point of writing this thesis, there are no cryptanalytic approaches that can exploit this additional attack vector. The benefit of this approach is that the smaller parameters entail drastically reduced key sizes and efficient algorithms for key generation, signing, and verification. Overall, the performance numbers of MAYO are quite convincing and are at the level of standardized lattice-based signatures, as depicted in Table 1.1. QR-UOV, SNOVA, and VOX also apply techniques to reduce the public key size compared to UOV. PROV is defined in a way that admits an EUF-CMA security proof under the assumption that the public key map is hard to invert. TUOV adds some 'triangular' polynomials to the central map, but does not provide more efficiency or compactness compared to UOV.

Since UOV and MAYO are at the core of several of the research papers contributing to this thesis, we will introduce these schemes in more details in the following sections. For the remaining schemes, we will provide additional information in the respective chapter, if necessary.

2.2.1 UOV

The specification of UOV [BCD+23] has a different look than the short introduction to the OV approach we gave above. On the one hand, this is due to the modifications and optimizations that were applied over the years. The scheme only employs homogeneous polynomials, since the constant and linear parts do not add to the security of the scheme, but increase the key sizes and hamper efficiency. The same holds for the secret transformation T, which is chosen as a linear transformation, not as an affine one [BWP05]. The transformation S, which is present in some other multivariate schemes, was omitted right from the start, since it is not contributing to the security of UOV.

On the other hand, the description of UOV now focuses more on the oil space O, instead of the secret central map $\mathcal{F}$ that was defined by the equation $\mathcal{P} = \mathcal{F} \circ T$. In fact, one gets a concise characterization of UOV and its signing strategy, from the description manifested in [Beu21a]. Recall that $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ is a homogeneous quadratic map and the oil space O has the property that $\mathcal{P}(\mathbf{o}) = \mathbf{0}$ for all $\mathbf{o} \in O$. Assume that we want to find a preimage $\mathbf{s} \in \mathbb{F}_q^n$ with $\mathcal{P}(\mathbf{s}) = \mathbf{t}$, for a certain target value $\mathbf{t} = \mathsf{H}(\mathsf{msg}, \mathsf{salt}) \in \mathbb{F}_q^m$. We pick a (vinegar) vector $\mathbf{v} \in \mathbb{F}_q^n$ at random and insert it into the equation

$$\mathcal{P}(\mathbf{v} + \mathbf{o}) = \mathcal{P}(\mathbf{v}) + \mathcal{P}(\mathbf{o}) + \mathcal{P}'(\mathbf{v}, \mathbf{o}) = \mathbf{t}. \tag{2.2.2}$$

The map $\mathcal{P}' : \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q^m$, defined by Equation (2.2.2), is called the differential of $\mathcal{P}$ and is bilinear and symmetric [Beu21a]. Since for $\mathbf{o} \in O$, it holds that $\mathcal{P}(\mathbf{o}) = 0$, the equation above is a system of linear equations in $\mathbf{o}$, which can be reformulated to

$$\mathcal{P}'(\mathbf{v}, \mathbf{o}) = \mathbf{t} - \mathcal{P}(\mathbf{v}). \tag{2.2.3}$$

The oil vector $\mathbf{o}$ can be described as a linear combination of m basis vectors given in O. Hence, Equation (2.2.3) is a system of m linear equations in m variables. This can be solved efficiently for $\mathbf{o}$, or in case no solution exists, one samples a new vinegar vector $\mathbf{v}$ and tries again. The sum of the vinegar and oil vector $\mathbf{s} = \mathbf{v} + \mathbf{o}$ yields a preimage of the target $\mathbf{t}$ and constitutes the main part of the scheme's signature.

UOV functionalities and variants. There are three variants of UOV, namely, `uov.classic`, `uov.pkc`, and `uov.pkc+skc`, where the suffix indicates whether the keys are uncompressed, only the public key is compressed, or both the public and private key are compressed. The compression technique was introduced by Petzoldt et al. in [PTBW11]. The reduced key size is at the expense of increased signing and verification time, since expansion operations are added to the corresponding algorithms. This trade-off is visualized in Table 2.1.

Basically, all three variants are composed of the same five functionalities below, but depending on the variant, the UOV.ExpandSK and UOV.ExpandPK algorithm are considered as part of the key generation or the sign and verify algorithm. We go through the five functionalities in the following and explain the scheme in more detail, alongside its pseudocode. The algorithms are displayed exactly as in the same manner as they are shown in the specification of the UOV signature scheme [BCD+23].

Table 2.1: UOV variants with different key compression levels. CPU cycles are obtained from [BCD+23, Table 6] and benchmarked on a Intel Skylake processor. Public and secret key sizes are obtained from [BCD+23, Table 4].

UOV variant	Key pair	pk size	sk size	sign	verify
		in Bytes		in CPU cycles	
`uov.classic`	(esk, epk)	278 432	237 896	105 324	90 336
`uov.pkc`	(esk, cpk)	43 576	237 896	105 324	224 006
`uov.pkc+skc`	(csk, cpk)	43 576	48	1 876 442	224 006

Key generation. The key generation algorithm (see Figure 2.2) sets up the oil space O and the public map $\mathcal{P}$, such that $\mathcal{P}$ vanishes on O.

The m-dimensional oil space O is expanded from a randomly generated secret seed $\mathtt{seed}_{\mathsf{sk}}$ and stored in $\mathbf{O} \in \mathbb{F}_q^{n-m\times m}$, where the columns of the matrix $\bar{\mathbf{O}} = (\mathbf{O}, \mathbf{I}_m)^\top$ are the basis vectors that define O. The coefficients of the polynomials $\mathcal{P}^{(k)}(\mathbf{x})$ are stored in matrices

$$\mathbf{P}_i = \begin{pmatrix} \mathbf{P}_i^{(1)} & \mathbf{P}_i^{(2)} \\ 0 & \mathbf{P}_i^{(3)} \end{pmatrix}. \tag{2.2.4}$$

Hereby, the submatrices $\mathbf{P}_i^{(1)}$ and $\mathbf{P}_i^{(2)}$ can be chosen randomly and are expanded from a public seed $\mathtt{seed}_{\mathsf{pk}}$, which has been randomly generated. In contrast, $\mathbf{P}_i^{(3)}$ has to be computed according to the equation in Line 6 of Figure 2.2, to ensure $\mathcal{P}(\mathbf{o}) = \mathbf{0}$ holds for all $\mathbf{o} \in O$.

UOV.CompactKeyGen():

1 : $\mathtt{seed}_{\mathsf{sk}} \leftarrow \{0,1\}^{\mathsf{sk\_seed\_len}}$
2 : $\mathtt{seed}_{\mathsf{pk}} \leftarrow \{0,1\}^{\mathsf{pk\_seed\_len}}$
3 : $\mathbf{O} := \mathtt{Expand}_{\mathsf{sk}}(\mathtt{seed}_{\mathsf{sk}})$
4 : $\{\mathbf{P}_i^{(1)}, \mathbf{P}_i^{(2)}\}_{i\in[m]} := \mathtt{Expand}_P(\mathtt{seed}_{\mathsf{pk}})$
5 : **for** $j = i, \ldots, m$ **do**
6 : $\quad \mathbf{P}_i^{(3)} := \mathtt{Upper}(-\mathbf{O}^\top \mathbf{P}_i^{(1)} \mathbf{O} - \mathbf{O}^\top \mathbf{P}_i^{(2)})$
7 : $\mathsf{cpk} := (\mathtt{seed}_{\mathsf{pk}}, \{\mathbf{P}_i^{(3)}\}_{i\in[m]})$
8 : $\mathsf{csk} := (\mathtt{seed}_{\mathsf{pk}}, \mathtt{seed}_{\mathsf{sk}})$
9 : **return** $(\mathsf{cpk}, \mathsf{csk})$

Figure 2.2: UOV key generation algorithm [BCD+23].

Expand sk. During secret key expansion, the compressed secret key $\mathsf{csk} = (\mathtt{seed}_{\mathsf{pk}}, \mathtt{seed}_{\mathsf{sk}})$ gets expanded to $\mathsf{esk} = (\mathtt{seed}_{\mathsf{sk}}, \mathbf{O}, \{\mathbf{P}_i^{(1)}, \mathbf{S}_i\}_{i\in[m]})$. The algorithm is depicted in Figure 2.3. Here, $\{\mathbf{P}_i^{(1)}\}_{i\in[m]}$ is needed to evaluate $\mathcal{P}(\mathbf{v})$ from Equation (2.2.3) during signing. Note that the last m elements of $\mathbf{v} \in \mathbb{F}_q^n$ are set to zero, and random values get only assigned to the first v entries of $\mathbf{v}$. Therefore, it is indeed sufficient to add $\{\mathbf{P}_i^{(1)}\}_{i\in[m]}$ to esk. The matrices $\{\mathbf{S}_i\}_{i\in[m]}$ are computed according to the equation in Line 4 of Figure 2.3. This

can be interpreted as inserting the information obtained from $\mathbf{o} \in O$ into the term $\mathcal{P}'(\mathbf{v}, \mathbf{o})$. The matrix $\mathbf{O}$, which describes the oil space O also needs to be added to esk, since it is needed to compute the resulting oil vector $\mathbf{o}$ from the solution $\mathbf{x}$ of the linear system at the end of the signing algorithm, as we will see below.

$\mathsf{UOV.ExpandSK}(\mathsf{csk})$: // $\mathsf{csk} = (\mathsf{seed}_{\mathsf{pk}}, \mathsf{seed}_{\mathsf{sk}})$

1: $\mathbf{O} := \mathtt{Expand}_{\mathsf{sk}}(\mathsf{seed}_{\mathsf{sk}})$
2: $\{\mathbf{P}_i^{(1)}, \mathbf{P}_i^{(2)}\}_{i\in[m]} := \mathtt{Expand}_P(\mathsf{seed}_{\mathsf{pk}})$
3: **for** $i = 1, \ldots, m$ **do**
4: $\quad \mathbf{S}_i := (\mathbf{P}_i^{(1)} + \mathbf{P}_i^{(1)\top})\mathbf{O} + \mathbf{P}_i^{(2)}$
5: $\mathsf{esk} := (\mathsf{seed}_{\mathsf{sk}}, \mathbf{O}, \{\mathbf{P}_i^{(1)}, \mathbf{S}_i\}_{i\in[m]})$
6: **return** esk

Figure 2.3: Algorithm that expands csk to esk in UOV [BCD+23].

Sign. As indicated above, signing in UOV boils down to setting up and solving Equation (2.2.3). The individual steps are depicted in Figure 2.4. The target vector $\mathbf{t}$ is derived from the message and a randomly generated salt. Then, the vinegar vector $\mathbf{v}$ is generated and inserted into the term $\mathcal{P}'(\mathbf{v}, \mathbf{o})$ in Line 7. If the resulting matrix is invertible, i.e., the system is solvable, we evaluate also $\mathcal{P}(\mathbf{v})$ (Line 9) and solve the derived linear system (Line 10). The solution $\mathbf{x}$ yields the coefficients of the basis representation of the oil vector, so $\mathbf{o} = \bar{\mathbf{O}}\mathbf{x}$ computes the oil vector. The sum of $\mathbf{v}$ and $\mathbf{o}$ account for the solution vector $\mathbf{s}$ that forms the signature together with the salt.

$\mathsf{UOV.Sign}(\mathsf{esk}, \mathsf{msg})$: // $\mathsf{esk} = (\mathsf{seed}_{\mathsf{sk}}, \mathbf{O}, \{\mathbf{P}_i^{(1)}, \mathbf{S}_i\}_{i\in[m]})$

1: $\mathsf{salt} \leftarrow \{0,1\}^{\mathsf{salt\_}len}$
2: $\mathbf{t} \leftarrow \mathsf{H}(\mathsf{msg}||\mathsf{salt})$ // $\mathbf{t} \in \mathbb{F}_q^m$
3: **for** $ctr = 0, \ldots, 255$ **do**
4: $\quad \mathbf{v} := \mathtt{Expand}_v(\mathsf{msg}||\mathsf{salt}||\mathsf{seed}_{\mathsf{sk}}||ctr)$ // $\mathbf{v} \in \mathbb{F}_q^{n-m}$
5: $\quad \mathbf{L} := \mathbf{0}_{m\times m}$
6: $\quad$ **for** $i = 1, \ldots, m$ **do**
7: $\quad\quad$ Set i-th row of $\mathbf{L}$ to $\mathbf{v}^\top \mathbf{S}_i$
8: $\quad$ **if** $\mathbf{L}$ is invertible **then**
9: $\quad\quad \mathbf{y} := [\mathbf{v}^\top \mathbf{P}_i^{(1)} \mathbf{v}]_{i\in[m]}$ // $\mathbf{y} \in \mathbb{F}_q^m$
10: $\quad\quad$ Solve $\mathbf{L}\mathbf{x} = \mathbf{t} - \mathbf{y}$ for $\mathbf{x}$
11: $\quad\quad \mathbf{s} := [\mathbf{v}, \mathbf{0}_m] + \bar{\mathbf{O}}\mathbf{x}$ // $\mathbf{s} \in \mathbb{F}_q^n$
12: $\quad\quad \mathsf{sig} := (\mathbf{s}, \mathsf{salt})$
13: $\quad\quad$ **return** sig
14: **return** $\perp$

Figure 2.4: Algorithm that signs a message msg in UOV [BCD+23].

Expand pk. This routine, given in Figure 2.5, simply expands $\mathtt{seed}_{\mathsf{pk}}$ to get the coefficients in $\mathbf{P}_i^{(1)}$ and $\mathbf{P}_i^{(2)}$ and puts them together with $\mathbf{P}_i^{(3)}$ according to Equation (2.2.4). Thus, the expanded public key is just a complete description of the public map $\mathcal{P}$.

UOV.ExpandPK(cpk): // $\mathsf{cpk} = (\mathtt{seed}_{\mathsf{pk}}, \{\mathbf{P}_i^{(3)}\}_{i\in[m]})$

1 : $\{\mathbf{P}_i^{(1)}, \mathbf{P}_i^{(2)}\}_{i\in[m]} := \mathtt{Expand}_P(\mathtt{seed}_{\mathsf{pk}})$
2 : **for** $i = 1, \ldots, m$ **do**
3 : $\quad \mathbf{P}_i := (\mathbf{P}_i^{(1)}, \mathbf{P}_i^{(2)}, \mathbf{P}_i^{(3)})$
4 : $\mathsf{epk} := \{\mathbf{P}_i\}_{i\in[m]}$
5 : **return** epk

Figure 2.5: Algorithm that expands cpk to epk in UOV [BCD+23].

Verify. In order to verify a signature, one just recomputes the target value $\mathbf{t}$, evaluates the public key map at the solution vector $\mathbf{s}$, which is part of the signature, and checks if the vectors are identical. This process is depicted in Figure 2.6.

UOV.Verify(epk, msg, sig): // $\mathsf{sig} = (\mathbf{s}, \mathsf{salt})$

1 : $\mathbf{t} \leftarrow \mathsf{H}(\mathsf{msg}||\mathsf{salt})$ // $\mathbf{t} \in \mathbb{F}_q^m$
2 : **return** $(\mathbf{t} == [\mathbf{s}^\top \mathbf{P}_i \mathbf{s}]_{i\in[m]})$

Figure 2.6: Verification algorithm in UOV [BCD+23].

2.2.2 MAYO

MAYO is an UOV-based signature scheme introduced by Beullens in [Beu21b]. The public and secret key exhibit exactly the same structure as in UOV. They are given by a description of a multivariate quadratic map $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ and a seed that defines the oil space $O \subset \mathbb{F}_q^n$, for which $\mathcal{P}(\mathbf{o}) = 0$ for all $\mathbf{o} \in O$ holds, respectively. The essential modification is the idea to 'whip up' the public key map $\mathcal{P}$ into a larger map $\mathcal{P}^* : \mathbb{F}_q^{kn} \to \mathbb{F}_q^m$ during signing and verification. This is realized by defining

$$\mathcal{P}^*(\mathbf{x}_1, \ldots, \mathbf{x}_k) := \sum_{i=1}^{k} \mathbf{E}_{ii}\mathcal{P}(\mathbf{x}_i) + \sum_{1\le i<j\le k} \mathbf{E}_{ij}\mathcal{P}'(\mathbf{x}_i, \mathbf{x}_j), \tag{2.2.5}$$

where the matrices $\mathbf{E}_{ij} \in \mathbb{F}_q^{m\times m}$ are fixed system parameters with the property that all their non-trivial linear combinations have rank m. Note, that the whipped map $\mathcal{P}^*$ accepts k input vectors $\mathbf{x}_i \in \mathbb{F}_q^n$.

This property allows to reduce the dimension of the oil space O, which in turn leads to less variables n and hence more compact key sizes. Recall, that in UOV, it was necessary to set the dimension of the oil space O to $\dim(O) = m$, such that we have the same number of equations and variables in Equation (2.2.3). In MAYO, however, the goal is not finding a preimage of

the target vector $\mathbf{t}$ under the public map $\mathcal{P}$, but under the whipped map $\mathcal{P}^*$, i.e., to solve the equation

$$\mathcal{P}^*(\mathbf{s}_1, \ldots, \mathbf{s}_k) = \mathcal{P}^*(\mathbf{v}_1 + \mathbf{o}_1, \ldots, \mathbf{v}_k + \mathbf{o}_k) = \mathbf{t}, \tag{2.2.6}$$

for $(\mathbf{o}_1, \ldots, \mathbf{o}_k) \in \mathbb{F}_q^{kn}$ after fixing the vectors $(\mathbf{v}_1, \ldots, \mathbf{v}_k)$ to randomly generated values. Let $\dim(O) = o$ be the dimension of the oil space O. Then Equation (2.2.6) turns into a system of m equations in ko variables. Depending on the chosen parameter k, this allows for a considerably smaller oil space O with $\dim(O) = o < m$. The solution vectors $\{\mathbf{s}_i = \mathbf{v}_i + \mathbf{o}_i\}_{i\in[k]}$ again represent the core part of the signature and the verifier just has to check if they indeed form a valid preimage of $\mathbf{t}$, i.e., if $\mathcal{P}^*(\mathbf{s}_1, \ldots, \mathbf{s}_k) = \mathbf{t}$ holds true.

MAYO functionalities. The specification of MAYO [BCC+23] defines the same five functionalities that are given in the UOV specification.[1] Since the keys are, except for parameter choices, the same as in UOV, the MAYO.CompactKeyGen, MAYO.ExpandSK, and MAYO.ExpandPK algorithms are defined analogously to their UOV counterparts, apart from slight notation deviations. Thus, we omit the presentation of those algorithms here and refer to the specification [BCC+23]. For MAYO.Verify the adaptions compared to UOV are straightforward. One simply replaces the map $\mathcal{P}$ by $\mathcal{P}^*$ and checks whether $\mathcal{P}^*(\mathbf{s}_1, \ldots, \mathbf{s}_k) = \mathbf{t}$ is satisfied. Therefore, we also omit the depiction of MAYO.Verify here.

Sign. The MAYO.Sign algorithm, in contrast, features some adaptions, mainly induced by solving a linear system derived from $\mathcal{P}^*$ instead of $\mathcal{P}$. The physical attacks, which we are going to discuss in Chapter 3 and Chapter 4, refer to specific code lines. Therefore, we depict the MAYO.Sign algorithm in Figure 2.7.

We chose to take over the depiction from the MAYO specification, such that the code lines we refer to in this thesis match with the official document. However, we skip the first two code blocks, since they are not relevant to understand the signature scheme or comprehend the mentioned attacks. The main difference, compared to UOV, is the construction of the linear system in Lines 21 to 34. The terms $\mathcal{P}(\mathbf{v}_i)$ and $\mathcal{P}'(\mathbf{v}_i, \mathbf{o}_j)$, which result from Equation (2.2.6), have to be computed for multiple vectors. This is realized in Lines 28 to 29 and Lines 25 to 27, respectively. The multiplication with the emulsifier matrices $\mathbf{E}_{ij}$ and the accumulation of the terms, which are part of the whipping procedure that transforms $\mathcal{P}$ into $\mathcal{P}^*$ (see Equation (2.2.5)), is realized in Lines 30 to 34. The remaining steps of the signing process, i.e., the system solving and the addition of the oil and vinegar terms are again similar to UOV, except that one has to treat multiple vectors $\{\mathbf{s}_i = \mathbf{v}_i + \mathbf{o}_i\}_{i\in[k]}$.

2.3 Physical Attacks

Physical attacks in cryptography exploit the fact that cryptographic implementations can leak information or be manipulated in their execution on a

[1] Theoretically, one could define the same three variants (as in `uov.classic`, `uov.pkc`, `uov.pkc+skc`) for MAYO upon these five functionalities. The submitters of MAYO decided to only have one variant in the specification, which works with compressed public and secret keys, and is therefore comparable to `uov.pkc+skc`.

MAYO.Sign(esk, msg) // $\mathsf{esk} = (\mathsf{seed_{sk}}, \mathbf{O}, \{\mathbf{P}_i^{(1)}, \mathbf{S}_i\}_{i\in[m]})$

$\vdots$

// Hash message and derive salt and target vector $\mathbf{t}$
$\mathtt{M\_digest} \leftarrow \mathtt{SHAKE256}(\mathsf{msg}, \mathtt{digest\_bytes})$
$R \leftarrow \mathbf{0}_{R_{bytes}}$ or $R \xleftarrow{\$} \mathcal{B}^{R_{bytes}}$ // Optional randomization
$\mathsf{salt} \leftarrow \mathtt{SHAKE256}(\mathtt{M\_digest} \,\|\, R \,\|\, \mathsf{seed_{sk}}, \mathtt{salt\_bytes})$
$\mathbf{t} \leftarrow \mathtt{Decode}_{vec}(m, \mathtt{SHAKE256}(\mathtt{M\_digest} \,\|\, \mathsf{salt}, \lceil (m\log(q))/8 \rceil))$

// Attempt to find a preimage for $\mathbf{t}$
for ctr $= 0, \ldots, 255$ **do**
  // Derive $\mathbf{v}_i$ and r
  $V \leftarrow \mathtt{SHAKE256}(\mathtt{M\_digest} \,\|\, \mathsf{salt} \,\|\, \mathsf{seed_{sk}} \,\|\, \mathrm{ctr}, k \cdot v_{bytes} + \lceil ko\log(q)/8 \rceil)$
  for $i = 0, \ldots, k-1$ **do**
    $\mathbf{v}_i \leftarrow \mathtt{Decode}_{vec}(n-o, V[i \cdot v_{bytes} : (i+1) \cdot v_{bytes}])$
  $\mathbf{r} \leftarrow \mathtt{Decode}_{vec}(ko, V[k \cdot v_{bytes} : k \cdot v_{bytes} + \lceil ko\log(q)/8 \rceil])$

  // Build linear system $Ax = y$.
  $\mathbf{A} \leftarrow \mathbf{0}_{m\times ko} \in \mathbb{F}_q^{m\times ko}$
  $\mathbf{y} \leftarrow \mathbf{t}, \ell \leftarrow 0$
  for $i = 0, \ldots, k-1$ **do**
    $\mathbf{M}_i \leftarrow \mathbf{0}_{m\times o} \in \mathbb{F}_q^{m\times o}$
    for $j = 0, \ldots, m-1$ **do**
      $\mathbf{M}_i[j,:] \leftarrow \mathbf{v}_i^\intercal \mathbf{L}_j$ // Set j-th row of $\mathbf{M}_i$
    for $j = k-1, \ldots, i$ **do**
      $\mathbf{u} \leftarrow \{\mathbf{v}_i^\intercal \mathbf{P}_a^{(1)} \mathbf{v}_i\}_{a\in[m]}$ **if** $i = j$
      $\mathbf{u} \leftarrow \{\mathbf{v}_i^\intercal \mathbf{P}_a^{(1)} \mathbf{v}_j + \mathbf{v}_j^\intercal \mathbf{P}_a \mathbf{v}_i\}_{a\in[m]}$ **if** $i \neq j$
      $\mathbf{y} \leftarrow \mathbf{y} - \mathbf{E}^\ell \mathbf{u}$
      $\mathbf{A}[:, i\cdot o : (i+1)\cdot o] \leftarrow \mathbf{A}[:, i\cdot o : (i+1)\cdot o] + \mathbf{E}^\ell \mathbf{M}_j$
      if $i \neq j$ **then**
        $\mathbf{A}[:, j\cdot o : (j+1)\cdot o] \leftarrow \mathbf{A}[:, j\cdot o : (j+1)\cdot o] + \mathbf{E}^\ell \mathbf{M}_i$
      $\ell \leftarrow \ell + 1$

  // Try to solve the system
  $\mathbf{x} \leftarrow \mathtt{SampleSolution}(\mathbf{A}, \mathbf{y}, \mathbf{r})$
  if $\mathbf{x} \neq \perp$ **then**
    break

// Finish and output the signature
$\mathbf{s} \leftarrow \mathbf{0}_{kn}$
for $i = 0, \ldots, k-1$ **do**
  $\mathbf{s}[i\cdot n : (i+1)\cdot n] \leftarrow (\mathbf{v}_i + \mathbf{O}\mathbf{x}[i\cdot o : (i+1)\cdot o]) \,\|\, \mathbf{x}[i\cdot o : (i+1)\cdot o]$
return $\mathsf{sig} = \mathtt{Encode}_{vec}(\mathbf{s}) \,\|\, \mathsf{salt}$

Figure 2.7: The algorithm that signs a message msg in MAYO. The pseudocode is the same as given in Algorithm 8 of the MAYO specification [BCC+23]. We omitted the first code lines that include only the secret key decoding.

hardware device. This is especially concerning for embedded devices like smart cards, microcontrollers, and hardware security modules. There are two major classes of physical attacks: side-channel and fault injection attacks. In the following, we briefly introduce the history and theory of these attacks. Notably, the development of the two fields happened almost in parallel in the late 1990s, even though they were driven by different research teams. We focus on the threat models that are crucial to understand the concepts of the attacks in Chapter 3.

2.3.1 Side-Channel Analysis

The beginning of the modern era of side-channel attacks against cryptographic algorithms was marked by the two seminal papers of Kocher in the late 1990s: He presented a timing attack [Koc96] on implementations of RSA and Diffie-Hellman, exploiting the fact that the execution time can vary based on the input values and secret key. Furthermore, he showed that the power consumption also correlates with the data and especially the secret key being processed [KJJ99]. Even though side-channel attacks were not formally studied in the cryptographic community before that, related concepts already existed. The TEMPEST program (from the 1950s to 1970s) refers to the study of unintentional electromagnetic emissions from electronic equipment by the U.S. government and covers both methods to exploit this emission and how to protect equipment against this leakage. The devices and equipment they considered were not bound to cryptographic applications, but had a larger framework, which included keyboards, displays, rotor machines, and more. However, public documentation on this was limited. In 1985, van Eck [Eck85] demonstrated how emissions from monitors can be captured and reconstructed, proving that real-world EM-eavesdropping is feasible. Afterwards, the two mentioned publications by Kocher paved the way for side-channel analysis of cryptographic algorithms as we know it today. Another important result of the early stages of side-channel analysis was brought forth by Messerges et al. [MDS99]. They analyzed the vulnerability of public key algorithms against power analysis on smartcards. The experimental validation further underlined the practical viability of Kocher's methods.

From these, a tremendous variety of different side-channel attacks emerged, which we can not possibly cover in this thesis. All side-channel attacks on UOV-based schemes we are aware of, including those we developed ourselves, are power analysis attacks. Therefore, we will provide background for these methods in the following.

Power Analysis Methods

Digital circuits have static and dynamic power consumption. Static power is needed to keep the device running, when it is idle. Dynamic power is required to switch transistors when a signal changes from 0 to 1 and vice versa. The latter is proportional to the number of bits that change (i.e., Hamming distance) or the value of newly assigned data (i.e., Hamming weight). Of course, this depends on the executed instructions, but also on the processed data and intermediate values. Power analysis exploits this correlation between power consumption and intermediate computational states. Over the years several concepts have been

developed. We briefly present the classification that is also chosen in [MOP07], and refer to this book for further reading.

simple power analysis (SPA). Here, only a single or few power measurements of an operation on a target device can be conducted by the adversary. In its most simple case an adversary inspects the collected power traces to visually identify patterns or bit-dependencies. This attack type is mainly working for vulnerable implementations with clear, structured power signatures due to conditional branches. An example are RSA exponentiation loops, where the device performs a multiplication only if the secret key bit is set to 1. The power trace directly reveals the sensitive value, e.g., by spikes that occur only in one of the conditional branches.

However, in many cases a differentiation based on completely different operations that follow branches depending on the secret key bits is not possible. If the leakage is more subtle and harder to detect, it is helpful to characterize the device under attack by determining templates for certain instruction sequences. To this end, we might use another device of the same type as the attacked one and execute these instructions with different data and keys in order to record the resulting power consumption and group them according to the key bits. This is called the template building phase. In the template matching phase, the power trace of the target device processing the actual secret key is assessed. The template that exhibits the highest correlation to this trace indicates the correct key bits. This template approach was further optimized using machine learning (ML) techniques. In this case the data collected in the template building phase is used to train a ML model and the template matching phase is given by the prediction of the ML classifier.

Differential Power Analysis (DPA). These attacks are possible when a large number of power traces of a cryptographic algorithm can be gathered. While in SPA attacks the power consumption is mainly analyzed along the time axis, DPA attacks analyze how the power consumption at a fixed moment depends on the processed data. This generally involves forming a hypothesis value and then comparing the hypothesis against measured power traces. It is usually applied by a divide and conquer approach where the attacker repeats the same process on small parts of the key until the full key is recovered. Following [MOP07], this process can be described by five steps:

1. **Choosing an intermediate result of the executed cryptographic algorithm.** This intermediate result is the function $f(d, k)$ of a known non-constant data value d and an unknown value k, which represents a small part of the secret key.

2. **Measuring the power consumption.** Measure the power consumption of the target at the position of the function localized in Step 1 while it processes several different data blocks d_i. For each run, the attacker needs to know the data values d_i and records the corresponding power traces. Here, trace alignment is important, hence the same part of the traces should be caused by the same operations.

3. **Calculating hypothetical intermediate values.** This step computes possible intermediate values $v_{i,j} = f(d_i, k_j)$ for every possible secret value

k_j and all data values d_i. This is also the reason why one cannot attack the whole key at once, but has to focus on small fractions of it.

4. **Mapping hypothetical intermediate values to hypothetical power consumption values.** Each hypothetical intermediate value is mapped to a hypothetical power consumption $h_{i,j}$ by some simulation technique. Usually, Hamming weight or Hamming distance models are applied.

5. **Statistically comparing the hypothetical power values with the measured power traces.** In this step, the attacker compares the hypothetical power consumption values $h_{i,j}$ for all key hypotheses with the recorded traces of Step 2. Statistical methods can be applied to reveal the unknown value k that is used by the device. If Pearson's correlation coefficient is used here, the attack is often referred to as Correlation Power Analysis (CPA).

The number of traces that is needed to perform such an attack depends on many parameters, such as the quality of the obtained power traces or the accuracy of the applied simulation model for the hypothetical power consumption. For certain attacks, this can even go in the range of 100 000 traces or more [MOP07].

DPA can also be turned into template-based DPA attacks, when the attacker has the chance to characterize the power consumption of a device. They are essentially an extension of what was described for template-based SPA attacks. The first template-based DPA was introduced by Agrawal et al. [ARR03].

2.3.2 Fault Injection Attacks

Two ground-breaking papers in the context of fault attacks are introduced by Boneh et al. [BDL97] and Biham and Shamir [BS97] in 1997. Boneh et al. have shown how the signing key in RSA implementations based on the Chinese Remainder Theorem is exposed from a single erroneous RSA signature, as this might leak one of the factors of the RSA modulus. Futhermore, they extend their analyis to the identification protocol by Fiat-Shamir and Schnorr and elaborate that the secret key is also exposed from a certain amount of faulty executions of the protocol. Biham and Shamir proposed a related attack on symmetric key algorithms such as the Data Encryption Standard (DES), which is the predecessor of AES. They named their approach DFA. Both publications state that the exploited faults could just appear randomly due to malfunctions of the device, but also mention methods of how an adversary could tamper with the device to actively cause such events. We briefly introduce some of the most common methods that have been established over the years. Bar-El et al. [BCN+06] and Verbauwhede et al. [VKS11] published classification papers that provide a broad overview of existing fault injection techniques.

Clock glitching. Briefly overclocking or underclocking the CPU clock signal can violate timing constraints and cause the CPU to misinterpret or skip instructions. This requires a fast and precise clock control, which can be generated using an FPGA or glitching tool.

Voltage glitching. A rapid drop or spike in voltage supply causes setup or hold time violations inside the processor. This can corrupt the control flow or logic operations, and thereby skip instructions or load incorrect data. It requires a high-speed trigger to time the glitch accurately during a specific instruction or execution round.

Laser fault injection. A precisely targeted laser pulse is fired at a specific region of a chip while it operates. This often requires the decapsulation of the chip and a high-priced laser station with micrometer precision. It can be used to disrupt gates or force bit flips and can target specific registers or memory cells.

Electromagnetic fault injection. An electromagnetic pulse is emitted near the chip to induce transient voltages in the circuit. In general, this is less invasive than a laser, but harder to precisely control.

Rowhammer. This is a software-based approach that rapidly accesses ('hammers') rows of memory to cause electrical interference in adjacent rows, which eventually can lead to bit flips. It is enabled by hardware design flaws in DRAM, where cells are packed so tightly that repeated activation can leak charge into neighboring cells. These faults can be triggered remotely without physical access to the device.

Effect and Impact of Faults

These faults can have different effects depending on where, when, and how they are introduced. When they target the memory during fetch, load, or store operations, it might cause processed data to be corrupted. Faults in a register can lead to bit flips of certain values during the execution of an operation. Faulting the instruction decoder can trigger instruction skips of whole functions and, by targeting the control logic, one can manipulate loop counters or branch conditions. In their survey on fault attacks [GT04], Giraud and Thiebeauld also differentiate between permanent and transient faults.

In order to exploit the induced fault and reconstruct sensitive values, the provoked errors often need to be tracked precisely and their effect for subsequent intermediate values and computations has to be analyzed, which requires a profound knowledge of the underlying cryptographic algorithm.

2.3.3 Experimental Setup

In the previous two sections we elaborated on several physical attack methodologies, but omitted details about how to inject glitches or conduct power measurements technically. The range of possibilities to mount such physical attacks is enormous and depends on the concrete application, the target under attack, and also the technical equipment and budget of an adversary. The post-quantum signature algorithms we analyze in this thesis are not yet standardized and do not appear in real-world protocols or security applications. Therefore, the main goal is not to recreate realistic scenarios for our attacks, but to generate understandable results that enhance the knowledge about physical attack vectors of the analyzed signature algorithms. To this end, it is advisable to

facilitate a low-cost setup with open source software, that is affordable for universities, institutions, and other researchers of the cryptographic community. In the recent years, the ChipWhisperer ecosystem has become a well accepted framework for working with side-channel and fault analysis [Tec23]. Its setup is also accessible to a wide range of researchers as it reduces the electrical engineering background that is necessary to perform glitching attacks or power measurements. Consequently, we opted for this platform to execute our developed attacks on UOV-based signature algorithms. We briefly describe the workflow for conducting physical attacks with the ChipWhisperer setup here.

The target device, e.g., a STM32 microcontroller is connected to the ChipWhisperer capture device, which itself is connected to a general work station like a laptop. The ChipWhisperer provides built-in support to flash the required cryptographic implementation, e.g., a post-quantum signature algorithm, onto the target microcontroller. The firmware must be compiled in advance and makefiles for certain platforms, like the STM32, are also provided. The flashed code usually includes a trigger signal to mark the start and end of the operation under attack. A reliable trigger is essential for a precise alignment of the executed attack, such that the same operations are captured or skipped when the attack is repeated. Depending on the type of attack, parameters need to be configured. For power measurement, one defines the sampling rate, offset and length. For glitching attacks, the parameters include the width, offset, and trigger source. The actual attack is executed by running the flashed code on certain inputs (i.e., messages in case of signature algorithms) and observing the target's behavior, by either analyzing the conducted power measurement or the (possibly) faulted signatures. Subsequently, post-processing of the collected data is necessary to reveal intermediate or sensitive values of the target device. This requires statistical tools and analysis methods as described in Section 2.3.1 and Section 2.3.2, and enhanced knowledge of the investigated signature algorithm.

3

Developing Physical Attacks

In this chapter we provide context for our research results given in [**A**KK+22], [**A**MS+24], and [**A**CK+23]. The former two are fault injection attacks on Rainbow and MAYO, while the latter one is a power analysis attack on UOV. The corresponding publications can be found in Appendix A.

The design and implementation of new physical attacks is an indispensable part of making cryptographic algorithms more resistant towards these kind of attacks, and thus, more secure. Developed attacks prove certain vulnerabilities in the analyzed implementation and materialize the need to put countermeasures into practice. These countermeasures generally produce an overhead in execution time and/or memory requirement, so that the willingness to accept these disadvantages in favor of more security only increases, when attacks are carried out properly. In multivariate cryptography, the state of the art regarding secure implementations and physical attacks is not as advanced as in other post-quantum families, in particular lattice-based cryptography. There are timing attacks [BBB+25], fault attacks [BP18; KPLG24; PP21], cache attacks [BHLY16], and other side-channel attacks [EFGT17; RRCB20; SLKG23] on a variety of lattice-based schemes, and this only represents a fraction of the existing literature on that topic. The survey by Ravi et al. [RCDB24] provides a broad overview for lattice-based schemes.

In multivariate cryptography, many schemes got broken, e.g., LUOV, GeMSS, and Rainbow, whereas others, like MAYO, SNOVA, and QR-UOV, are really young and are mainly analyzed in terms of their mathematical security. Furthermore, many research experts focused more on lattice-based signature and encryption schemes, since those were the clear favorites to be selected for standardization in the recent years. Nevertheless, there are some physical attacks on multivariate signature schemes. We present them subsequently and show how our research work fits into this scope.

3.1 Fault Injection Attacks on Multivariate Signature Schemes

In this section we discuss our research works [**A**KK+22] and [**A**MS+24], present related work, and point out the minutiae of the attacks. The former contains fault attacks via instruction skips on Rainbow, while the latter targets MAYO.

Both contain an attack that exploits the role of fixed vinegar variables in these schemes. To enable a better classification of our work, we first outline the history of fault attacks on multivariate schemes, starting from the pioneering work of Hashimoto et al. [HTS11]. Second, since there are many attacks, we present an overview of all publications that deal with the fault model of fixing the random ephemeral vinegar variables to elaborate the differences between the results.

3.1.1 Literature Overview

The groundwork for fault attacks against multivariate public-key cryptography was laid in 2011, when Hashimoto et al. [HTS11] presented two fault attack models that were applicable to various popular multivariate schemes. They classified multivariate schemes into big field (BF) type (e.g., MI, HFE, SFLASH [ACDG03]) and stepwise triangular system (STS) type[1] schemes (e.g., UOV [KPG99], Rainbow [DS05b], TTS [YC05]). The two introduced fault attacks inspired quite some follow-up work, so we present their development in detail here.

Bit flip on a coefficient in the central map. First, Hashimoto et al. [HTS11] introduce a fault model, where the attacker is able to cause a fault that changes a coefficient of the central map. It is applicable to both types of schemes, but the number of faults and required message-signature pairs for partial key recovery is different. They state that for BF type schemes they require one fault and $(n+1)(n+2)/2$ message-signature pairs and for STS type schemes they require $n-1$ faults, but only a single message-signature pair to recover *parts* of the secret affine transformations. In both cases, the authors do not state the concrete complexity of recovering the secret transformations S and T completely, but claim that the fault attack reduces the parameters in the affine transformation to recover them with reduced complexity using existing algebraic attacks. The presented fault attacks are neither physically executed on a concrete implementation nor simulated and remains completely theoretical. The authors present success probabilities, but these refer to the likelihood that a randomly faulted coefficient in the secret key is indeed one belonging to the central map, and not one of the other maps S or T, which are also part of the secret key. The probabilities do *not* report the likelihood that a certain physical event indeed causes a bit flip in the memory of a device that is running the algorithms. Several years later, Krämer and Loiero [KL19] tried to apply this fault model specifically to the signature schemes UOV and Rainbow. They were successful in the case of Rainbow, but found that UOV was not vulnerable to this attack, since it relied on recovering some information about the map T first. This map is simply not present in UOV, in contrast to Rainbow. Furue et al. [FKNT22] managed to adapt the approach by [HTS11] and [KL19], such that they gained information about the entries in S, which in turn enables a

[1] Multivariate schemes are divided into BF and STS type schemes in [HTS11]. In BF schemes, the central map is given by polynomials over a large extension of the original field. In STS type schemes, the variables are divided into different groups, for which the solution is determined subsequently. The latter form a subset of the single field schemes, characterized in [KL19].

secret key recovery with reduced complexity,[2]. Again, none of the mentioned attacks were executed in practice.

A different, but related approach is used by Mus et al. [MIS20] in their QuantumHammer attack on LUOV. Their fault model also employs bit flips in memory, but they target the secret transformation T instead of the central map $\mathcal{F}$. Moreover, they present a practical technique to induce the faults, namely the Rowhammer attack discovered by Kim et al. [KDK+14]. The attack is software-only, i.e., physical access to the target device is not a requirement, and applied to a constant-time AVX2-optimized implementation on a Haswell system equipped with Intel Core i7-4770 CPU @ 3.40GHz and 2 GBytes DDR3 Samsung DRAM. In an online phase, where the victim is running and performing signing operations, the bit-flips are introduced by rapidly activating neighboring DRAM rows. This is possible when the DRAM is shared between different processes or virtual machines. The resulting faulty signatures are processed offline on a separate machine to recover key bits. In less than four hours of online phase, the authors are able to recover a good share of the secret transformation T and finish the key-recovery attack algebraically in around 49 hours. For more details on the bit-tracing and algebraic post-processing attack, we refer to [MIS20]. The QuantumHammer fault attack represents the only end-to-end fault attack, where memory cells are manipulated, on a UOV-based signature scheme.

Instruction skip on random ephemeral values. Second, [HTS11] discussed a fault model, which assumes that an attacker is able to fix a part of the ephemeral random values generated during the signing procedure. Considering the BF type schemes, this is only applicable to schemes employing a 'minus' or 'vinegar' variation, where such random parameters are used. In the former, several polynomials of the central map are removed to provide less information to an attacker. The latter describes the technique to add more variables into the equations, which are then randomly fixed during the signing operation. For the STS type schemes, including UOV and its variants, the randomly generated vinegar variables are the target of such an attack. The authors state that it requires $n - u + 1$ message-signature pairs to recover a part of the secret transformation S, where u denotes the number of fixed vinegar variables.

In [KL19] the approach is applied concretely to UOV and Rainbow. The authors state complexity estimates for the reduced MinRank attack to recover S completely, confirming the findings of [HTS11].

Shim and Koo [SK20] developed this approach further to a full key-recovery attack. They discovered that if all vinegar variables are affected by the attack, full key recovery is possible in polynomial-time for UOV. In the case of Rainbow, a significant complexity remained, ranging from 2^{38} to 2^{42} depending on the concrete parameter set.[3] Additionally, they stated the complexity for their algebraic post-processing method when the number of affected vinegar variables u was less than v, the number of all vinegar variables, for several values of u.

[2] In [HTS11] the central map is called G with the public map being $F = T \circ G \circ S$. In [KL19] the central map is denoted by F and the public map $P = T \circ F \circ S$ for Rainbow and $P = F \circ S$ for UOV. In [FKNT22] the central map is called F and the public map $P = F \circ T$.

[3] [SK20] considered three parameter sets for the attack. All of them were designed to satisfy NIST security level I. For parameter sets of larger security levels, the remaining complexity is considerable larger.

Furthermore, the authors of [SK20] discussed three fault scenarios, the *Reuse*, *Reveal*, and *Zeroing* fault model, and computed the number of faulted signatures that was needed for the previously described key-recovery attack. The concrete number depends on the parameter of the investigated scheme, but ranges from a few dozens up to a few hundreds. The evaluation of the attack models above remains completely theoretical and it is not indicated how the faults could be realized in practice.

In [**A**KK+22] we drive forward the results of [SK20] by analyzing the fault attack targeting the vinegar variables in Rainbow. We managed to improve the algebraic post-processing step, such that the secret key recovery could be realized in polynomial time. Furthermore, the attacks are simulated on an ARM Cortex-M4 architecture environment based on Unicorn, which in turn is based on QEMU. This means the compiled binary of the unmodified source code is executed within the simulation just as a real device with this architecture would execute it. This framework then allows cycle-accurate skipping faults and memory analysis during execution. As proof of concept, the simulated faulty signatures are then used for the newly introduced key-recovery attack.

In [SM**A**+24] we transferred the attack to MAYO for the first time, but only discussed it theoretically in order to derive and implement countermeasures for a fault-tolerant hardware implementation of MAYO. We will provide more details about this work in Chapter 4. The first practical execution of the fault attack that targets the sampling process of the vinegar variables was realized in [**A**MS+24]. We used the ChipWhisperer-Lite evaluation platform to launch the fault injection attack. The algorithm is running on a standard STM32F4 target board containing an ARM Cortex-M4 STM32F415RGT6 processor mounted on an CW308 UFO board. We bypass the sampling loop by performing a clock-glitching attack. Therefore, the ChipWhisperer modifies the clock pulse and thus, causes the microcontroller under attack to exit the sampling loop early. Our key-recovery computations show that this attack works in polynomial-time in both cases, when the vinegar variables are initialized with zero values and when they are reused in memory for two consecutive signature generations.

Jendral and Dubrova [JD24a] introduced three fault attacks that reveal the vinegar variables by fixing the seed that is used to derive the vinegar variables. Two of the attacks target the absorption phase of the SHAKE256 algorithm that is part of the FIPS202 standard [NIST15] and used to sample the vinegar variables. The third attack skips the initialization of one of the arguments for the computation of the seed. The authors verified the attack experimentally on the ChipWhisperer-Husky setup, using the same target board as above, but mounted on a CW313 adapter board. Banegas and Villanueva-Polanco [BV24] transfer the fault model to SNOVA. They analyze two scenarios where they either fix field elements or just bits of the vinegar variables and support their findings with key-recovery simulations. This work finalizes the research branch focusing on this specific attack at the time of writing this thesis.

From the developments in the algebraic post-processing of the aforementioned fault attacks, one can summarize this fault model as follows. The random sampling of the vinegar variables is manipulated via fault-injection methods to reveal the vinegar part of the final signature to an attacker. Since the signature is the sum of the vinegar and oil part, this in turn reveals an oil vector, which enables efficient key recovery with certain algebraic attacks.

However, there is another approach that allows adversaries to separate the contribution of the oil and vinegar terms to the signature. One can try to tamper with the addition of the two parts at the end of the signing process. There are two works in the literature that apply this approach, namely [**A**KK+22] and [SM**A**+24]. The former aims at skipping the application of the secret transformation S to some intermediate values in Rainbow, which is equivalent to preventing the combination of the oil and vinegar shares of the signature. The latter discusses the attack as a theoretical threat to the presented hardware implementation of MAYO and implements countermeasures to mitigate it.

3.1.2 Classification of our Fault Attacks

The two papers contributing to this section both contain an attack that targets the random sampling of the vinegar variables. As the summary above indicates, there are many research results that deal with this approach, targeting various UOV-based signature schemes and featuring various levels of practicality. To provide an easier comparison and also to illustrate how the attack approach developed and improved over time, we present the key figures and evaluation methods in Table 3.1. Similar overviews are provided by [JD24a] and [BV24] in their related work sections.

Table 3.1: Comparison of fault attacks on UOV-based signature schemes that target the contribution of the vinegar variables to the signature by either zeroing, reusing, or revealing them. When a cell states *Multiple*, the concrete number depends on the chosen parameters m and n, and usually varies between several dozens and a few hundred.

	Scheme	Evaluation	#Signatures	#Faults
Hashimoto et al. [HTS11]	Several	Theoretical	Multiple	Multiple
Krämer and Loiero [KL19]	UOV/ Rainbow	Theoretical	Multiple	Multiple
Shim and Koo [SK20]	UOV/ Rainbow	Theoretical	Multiple	Multiple
Aulbach et al. [**A**KK+22]	Rainbow	Simulation	Multiple	1
Sayari et al. [SM**A**+24]	MAYO	Theoretical	2	1
Aulbach et al. [**A**MS+24]	MAYO	Practical	1	1
			2	1
Jendral and Dubrova [JD24a]	MAYO	Practical, Simulation	1	1
			1	1
			1	1
Banegas and Villanueva-Polanco [BV24]	SNOVA	Simulation	Multiple	1
			1	Multiple

3.2 Side-channel Attacks on Multivariate Signature Schemes

In this section we give an overview of side-channel attacks on multivariate signature schemes. The first research works in the early 2000s were motivated by the participation of some multivariate signatures in the New European Schemes for Signatures, Integrity, and Encryption (NESSIE) project and their potential use in smartcards. Afterwards, we are not aware of published results until more than a decade later, when interest in multivariate signatures and their practical applications rose again. Based on the provided overview, we outline how our profiled side-channel analysis of UOV can be classified compared to other known results.

3.2.1 Literature Overview

The first side-channel attacks on multivariate signatures were designed to break FLASH and SFLASH, which are both based on the MI-cryptosystem. These schemes attracted some interest of researchers since they have been proposed within the NESSIE project for the use on low-cost smartcards. The faster variant SFLASH was even one of the three finally selected digital signature algorithms in 2003, but got broken in 2007 by Dubois et al. [DFSS07] with a strong cryptanalytic attack. Some years before, Steinwandt et al. [SGB01] already theoretically discussed a DPA to reveal a seed that is part of the secret key. With said seed at hand, efficient algebraic attacks to recover the remaining parts of the secret key were possible. The authors claim that a DPA on the XOR-operations of the SHA-1 function that has the secret seed as input would be possible and require the power consumption of several hundred executions of the signing algorithm. The authors argue that the attack lacks an experimental proof, since there were no smart card implementations of SFLASH available at that time. Okeya et al. [OTV04] performed a similar DPA to recover the secret seed in SFLASH, but their target was the addition modulo 2^{32} inside the SHA-1 function, not the XOR-operation. They took $200,000$ samples of power consumption on an IC chip, while the device was running the SFLASH signing procedure. The authors declare that the large amount of observed samples was taken to make differences in the power consumption clearly visible, but less samples should be sufficient to differentiate between the secret key bits. Unfortunately, the authors of [OTV04] do not report more details about the target device and the used implementation.

More recently, in 2017, Yi and Li presented a DPA on enTTS [YL17], a signature scheme that contains some common features with Rainbow, such as the layer structure of the central map and the enclosing affine transformations. The authors perform their attack on a naive enTTS implementation on Application Specific Integrated Circuits (ASIC) with simulated power traces, generated with ModelSim. The experimental results show that it is possible to recover the secret affine transformation and central map with around $2,000$ traces. However, as part of the attacker model, the authors also assume a possible fault injection attack that fixes certain unknown items in the signature generation to enable the DPA. In total, this renders the assumed attacker model relatively strong. The same holds for the side-channel attack presented in [YN18]. Yi and Nie recover the secret coefficients of the central map by a DPA of the matrix-vector

multiplications and evaluation of the secret polynomials. However, similar to [YL17], the authors assume that it is possible to fix certain variables in the computation to known values with a fault attack. The authors do neither indicate how to achieve this behavior nor execute the fault attack on a target device to show that this assumption is plausible. The power measurements for the side-channel analysis are conducted with an oscilloscope, while the UOV implementation is running on a Sakura-G FPGA board. The authors state that they were able to recover the secret key from the analysis of the signing procedure on around $4,000$ messages within a few hours.

Park et al. [PSKH18] introduced a side-channel attack on Rainbow, targeting the secret linear transformations S and T, which are used to hide the structure of the central map F. The authors observed that some input values to these transformation are publicly available for an attacker, which allowed them to mount an effective Correlation Power Analysis (CPA) attack. Therefore, the authors ported the corresponding matrix-vector product code (of reduced size) on an Atmel XMEGA128 target board and recorded power traces with the ChipWhisperer-Lite evaluation platform. They showed that around 30-40 traces were enough to differentiate between certain secret key bits. The CPA was completed by an efficient key-recovery attack, revealing the complete secret key. The authors also discussed how their attack might apply to UOV, where only one of the secret transformations is present. In [PSN21], Pokorny et al. took a similar approach, also attacking Rainbow with CPA on the secret linear transformations. It took them a few hundred power traces to recover secret key bits with high probability, depending on the exact location they target with the CPA. However, compared to [PSKH18], their setup seems more realistic, since they used the implementation submitted to the NIST PQC competition round 2 and a 32-bit target device, namely the STM32F303 microcontroller based on an ARM Cortex-M4 core. The measurements were also conducted with a ChipWhisperer-Lite.

Our side-channel analysis of UOV [**A**CK+23] marks the first attack that does not directly target the secret key bits in UOV, but aims at recovering the vinegar variables first. This is achieved by analyzing their evaluation under (publicly known) coefficients of the central map, which ultimately boils down to matrix-vector multiplications over the employed finite field $\mathbb{F}_{2^8}$. Since these vinegar variables are multiplied with several dozens (depending on the concrete parameter set) of known values, this results in a strong leakage, which can be exploited. We opt for a profiling attack that takes a template trace for each possible value in the field $\mathbb{F}_{2^8}$. This enables us to implement the first single trace attack on UOV, and a UOV-based signature scheme, in general. Knowledge of the used vinegar variables and the corresponding oil vector, again leads to efficient secret key recovery, as it was the case for the mentioned recent fault attack scenarios. The measurements were taken using the ChipWhisperer-Lite and a 32-bit STM32F303 microcontroller with ARM Cortex-M4 architecture. The algebraic post-processing runs in polynomial-time, and presents the first efficient secret key recovery in UOV from a single oil vector. The result regarding the algebraic attack was achieved independently by Pébereau [Péb24].

Jendral and Dubrova [JD24b] perform side-channel attacks on two functions in MAYO. One targets the matrix-matrix multiplication during secret key expansion, and the other one the matrix-vector multiplication at the end of the signing process. The latter operation is comparable to the matrix-vector

product that was attacked in [PSKH18] and [PSN21]. The first attack is comparable to our attack [**A**CK+23], since it also exploits the large amount of multiplications with publicly known values, stored in these matrices. The authors also use a profiling approach. However, they are modeling information leakage with a powerful deep learning model instead of templates. They achieve very high success probabilities, especially for the first attack, where the number of exploitable modular multiplications is very high. From their deep learning-assisted power analysis, they directly recover an oil vector. Similar to previous works, this oil vector can be used for efficient secret key recovery. Their work presents the first single-trace attacks on the signature scheme MAYO.

3.2.2 Classification of our Side-Channel Attack

In Table 3.2, we summarize and compare the side-channel attacks on UOV-based signature schemes, which we introduced above. Our attack is the only one, where the power analysis part targets recovering the vinegar vector first, instead of directly going for the secret oil vector.

Table 3.2: Comparison of side-channel attacks on UOV-based signature schemes.

	Scheme	Target Subroutine	Attack Type	#Traces
Yi and Nie [YN18]	UOV	Vinegar eval. $\mathcal{P}(\mathbf{v})$ (FIA) and matrix-vector mul. (DPA)	FIA, DPA	4,000
Park et al. [PSKH18]	Rainbow	Matrix-vector mul. Ox	CPA	30
Pokorny et al. [PSN21]	Rainbow	Matrix-vector mul. Ox	CPA	300
Aulbach et al. [**A**CK+23]	UOV	Vinegar eval. $\mathcal{P}(\mathbf{v})$	Profiling Template	1
Jendral and Dubrova [JD24b]	MAYO	Secret key expansion or matrix-vector mul. Ox	Profiling Deep Learning	1

Note that the last two attacks are profiling attacks. They manage to get along with a single attack trace, but, naturally, more power traces are needed to create the template or train the model in the case of the deep-learning attack. The evaluation of the vinegar vector $\mathcal{P}(\mathbf{v})$ and the secret key expansion both belong to the most costly subroutines of the signing procedure of the respective UOV-based signature schemes, so protecting them against these side-channel attacks will probably introduce a lot of overhead to the respective schemes. In Chapter 4, we will provide more details on the cost of masking these operations in UOV and MAYO.

3.3 Future Research Directions

The continuous development of strong physical attacks in realistic scenarios is necessary to highlight the urgency of protected implementations before UOV-based signature schemes are widely deployed in practice. We can split the research paths to follow in three categories: (1) the technical part of the practical attack execution; (2) the analysis part, including the selection of

vulnerable code lines and evaluation of power traces and fault behavior; (3) the mathematical part of exploiting the gained information to achieve key recovery.

The technical part includes to select an optimized and mature implementation as target. UOV and MAYO already provide optimized implementations for x86 with AVX2, ARM Neon, and ARM Cortex-M4. Additionally, there are FPGA implementations that could be evaluated on different hardware target platforms. For the other schemes, it is expected that more advanced implementations will be created during the course of the standardization process.

Regarding the analysis part, it would be interesting to see how modern machine learning techniques can increase the accuracy of known attacks or reduce the number of required traces. So far, [JD24b] is the only side-channel attack on a UOV-based scheme, which employs a deep-learning approach. Furthermore, next to the subroutines that were shown to be vulnerable by previous attacks, there are more operations that seem exploitable. We discuss several of them in [**A**CK25] and present them in Table 4.2 (fault injection attacks) and in Table 4.3 (side-channel attacks).

For the algebraic part, the next step is to develop an approach that deals with vectors that are 'almost' an oil vector. Recent attacks only declare the physical attack to be successful, in case an oil vector is recovered, since it can be utilized for efficient key recovery. Since the proposed attacks are rather strong, they still achieve a high success probability and do not need to cope with the case that only a vector, which is 'close' to an oil vector, is recovered. However, in the presence of more noise or when attacking protected implementations, it might be reasonable to utilize these vectors, too. To this end, one needs a more precise definition of what 'close' to an oil vector means. Are there a few entries that are guessed incorrectly? Are the entries only recovered up to Hamming weight, but not determined exactly? Thus, the development of a sophisticated algebraic approach that manages to handle those approximate oil vectors to determine the complexity of key recovery is desirable.

4

Enhancing Implementation Security

With the research works [SM**A**+24] and [**A**CK25], listed in Appendix B, we contribute towards more secure implementations of UOV-based signature schemes with respect to physical attacks. As one would expect, the state of the art concerning implementation security is less advanced for UOV-based algorithms, and for multivariate algorithms in general, compared to those that are already used in practice. Classical algorithms like AES and RSA are in the field for several decades now, and their deployment started even before physical security was raised as a general concern. Thus, the physical attacks described in Section 1.3 and Section 2.3 affected the cryptographic algorithms and their implementation while they were already securing certain applications. This obviously constitutes a vulnerability, which eventually might lead to the loss of sensitive data or even safety issues. With the partially started, but predominantly still forthcoming, migration from classical to PQC algorithms, there is the chance of hardening new algorithms against common attacks before they are deployed in practice. In lattice-based cryptography, this has been and still is an active research area. There are works on constant-time implementations [BBE+19; KAA22], efforts to protect from cache attacks [BBK+17], and efficient masked implementations of multiple subroutines even at higher orders are available [BPO+20; BBE+18; BDH+21; OSPG18; PPRS23; SPOG19].

In multivariate cryptography the situation is different for the same reasons as elaborated in Chapter 3. To the best of our knowledge, there are no implementations of multivariate schemes available, which are considered secure or at least hardened against physical attacks. Some of the used subroutines, e.g., the Gaussian solver to obtain a solution of the derived linear system in UOV-based schemes, are implemented in constant-time, but the protection against power analysis and fault injection attacks is not yet sufficiently developed. Although the research works assembled in Section 3.1 and Section 3.2 discuss individual countermeasures for the presented attacks, these have not been incorporated in existing optimized implementations of the schemes. It is therefore still difficult to estimate the overhead that can be expected if UOV-based signature schemes are implemented with countermeasures against physical attacks.

We divide our contribution to address this gap into two main parts. The first part is the elaboration of an overview of all potential vulnerabilities. This topic is presented in Section 4.1. It is necessary to get a hold of every potential

attack vector, in order to specify the requirements of secure implementations. The second part, given in Section 4.2, is dedicated to the discussion of implementation guidelines, and the presentation of countermeasures that we incorporate into implementations of UOV and MAYO.

4.1 Overview of Vulnerabilities

The core part of the publication [**A**CK25] contributes to the findings presented in this section. We consider all physical attacks on UOV-based schemes available in the literature, also those, which have been proposed against outdated versions of UOV, or broken schemes, like LUOV and Rainbow. Afterwards, we evaluate if these attacks can be translated to UOV and try to complete the list of vulnerabilities by investigating the remaining parts of the implementation that have not been targeted yet. Furthermore, we analyze if the attacks can be transferred to the three multivariate signatures besides UOV that advanced to the second round of NIST's call for additional signatures, namely MAYO, QR-UOV, and SNOVA.

Since QR-UOV and SNOVA have not been introduced as part of Chapter 2, we will give a short introduction in the next paragraphs. Based on the provided signing algorithms, we will then extract similarities that hold for the signing procedure of UOV-based schemes. This will help us to assess the transferability of side-channel and fault attacks to the relevant schemes, in general. Lastly, we will provide our results on the applicability of existing physical attacks on concrete implementations of these schemes.

QR-UOV. QR-UOV is a variant of UOV, which is using a quotient structure in its secret and public matrices. It was initially proposed by Furue et al. in [FIKT21] and is another attempt to reduce the public key size of UOV. QR-UOV exhibits major similarities with UOV and uses the traditional description of UOV, focusing on the central map $\mathcal{F} = (\mathcal{F}^{(1)}, \ldots, \mathcal{F}^{(m)})$ in its specification. The authors specify the linear transformation S, which is employed to hide the structure of $\mathcal{F}$ in the public map $\mathcal{P}$, via $\mathcal{P} = \mathcal{F} \circ S$. Recall, that the coefficients $\gamma_{i,j}^{(k)} \in \mathbb{F}_q$ define the quadratic parts of $\mathcal{F}^{(k)}$ in Equation (2.2.1) for UOV. Thus, the private maps can be represented by m matrices $F^{(k)}$, containing the field elements $\gamma_{i,j}^{(k)}$. In QR-UOV, the matrices $F^{(k)}$ exhibit more structure and are composed of elements of the matrix ring $X_{i,j}^{(k)} \in \mathbb{F}_q^{l \times l}$ instead. These matrices correspond to elements of a quotient ring $\mathbb{F}_q[x]/(f)$, where $f \in \mathbb{F}_q[x]$ is an irreducible polynomial of degree l. Thus, not every element of the matrices $X_{i,j}^{(k)}$, which compose $\mathcal{F}^{(k)}$, needs to be stored, since the l^2 entries of $X_{i,j}^{(k)}$ can be represented by just l coefficients of the corresponding quotient ring element. The same holds for the elements of the public matrices $P^{(k)}$, which are computed by $P^{(k)} = S^\top F^{(k)} S$.

The described modifications mainly impact the key generation and allow a more compact representation of the keys. The signing and verification processes, in contrast, are the same as those for plain UOV, from a logical point of view. Nevertheless, the signing and verification algorithms look differently to those of UOV, since the submitters of QR-UOV chose the traditional description in their specification. We depict the signing algorithm in Figure 4.1, where we

allocate the found vulnerabilities to the respective code lines in the pseudocode. We allocate the logical steps of the signing algorithm to the corresponding code lines in Table 4.1.

QR-UOV.Sign(msg, pk, sk)

Input: message msg, public key pk, secret key sk
Output: signature sig

1 : $(\mathtt{seed}_{\mathsf{pk}}, \{P_{i,3}\}_{i\in[m]}) \leftarrow \mathsf{pk}$
2 : $\mathtt{seed}_{\mathsf{sk}} \leftarrow \mathsf{sk}$
3 : $\{P_{i,1}\}_{i\in[m]}, \{P_{i,2}\}_{i\in[m]} \leftarrow \mathtt{Expand}_{\mathsf{pk}}(\mathtt{seed}_{\mathsf{pk}})$
4 : $S' \leftarrow \mathtt{Expand}_{\mathsf{sk}}(\mathtt{seed}_{\mathsf{sk}})$
5 : **for** i from 1 to m **do**
6 : $\quad F_{i,1} \leftarrow P_{i,1}$
7 : $\quad F_{i,2} \leftarrow -P_{i,1}S' + P_{i,2}$
8 : **endfor**
9 : $S \leftarrow \begin{pmatrix} I_v & S' \\ 0_{m\times v} & I_m \end{pmatrix}$
10 : $\mathbf{y} = (y_1, \ldots, y_v)^\top \xleftarrow{\$} \mathbb{F}_q^v$
11 : $L \leftarrow \begin{pmatrix} 2\mathbf{y}^\top F_{1,2} \\ \vdots \\ 2\mathbf{y}^\top F_{m,2} \end{pmatrix}$ // $\mathrm{L} \in \mathbb{F}_q^{m\times m}$
12 : $\mathbf{u} \leftarrow (\mathbf{y}^\top F_{1,1}\mathbf{y}, \ldots, \mathbf{y}^\top F_{m,1}\mathbf{y})^\top$ // $\mathbf{u} \in \mathbb{F}_q^m$
13 : **repeat**
14 : $\quad r \xleftarrow{\$} \{0,1\}^\lambda$
15 : $\quad \mathbf{t} \leftarrow \mathtt{Hash}(\mathsf{msg}||r)$ // $\mathbf{t} \in \mathbb{F}_q^m$
16 : **until** $L\mathbf{x} = \mathbf{t} - \mathbf{u}$ has solutions for $\mathbf{x}$.
17 : Choose one solution $(y_{v+1}, \ldots, y_n) \in \mathbb{F}_q^m$ of $L\mathbf{x} = \mathbf{t} - \mathbf{u}$ randomly.
18 : $\mathbf{s} \leftarrow S^{-1}(y_1, \ldots, y_v, y_{v+1}, \ldots, y_n)^\top$
19 : **return** sig $= (r, \mathbf{s})$

Figure 4.1: Algorithm that signs a message msg in QR-UOV. The algorithm is depicted as in the QR-UOV specification document [FIH+23] submitted to the call for additional digital signatures.

SNOVA. SNOVA is a variant of UOV, where the quadratic polynomial equations are defined over the noncommutative ring $\mathcal{R} = \mathrm{Mat}_{l\times l}(\mathbb{F}_q)$. The authors apply even more structure to the central and public maps by including multiplication with other random matrices before and after the typical UOV structure. This renders the representation of the maps quite complicated. Each component of the central map $F = [F_1, \ldots, F_m] : \mathcal{R}^n \to \mathcal{R}^m$ is defined by

$$F_i = \sum_{\alpha=1}^{l^2} A_\alpha \cdot \left(\sum_{(j,k)\in\Omega} X_j (Q_{\alpha 1} F_{i,jk} Q_{\alpha 1}) X_k \right) \cdot B_\alpha, \tag{4.1.1}$$

where the $F_{i,jk}$ are randomly chosen from $\mathcal{R}$, A_α and B_α are invertible elements randomly chosen from $\mathcal{R}$, and $Q_{\alpha 1}, Q_{\alpha 2}$ are invertible matrices randomly chosen from $\mathbb{F}_q[S]$. Here, $\mathbb{F}_q[S]$ is a commutative symmetric subring of $\mathcal{R}$. Like in UOV, the public matrices are then set to fulfill the equation $P_i = F_i \circ T$ and therefore, are of similar type as the central matrices.

Despite these differences in the definition of the public and central maps, the signing procedure again follows a similar approach as other UOV-based signatures. We display the sign algorithm in Figure 4.2. Many of the operations are outsourced to other algorithms and we refer to [WCD+23] for the details. Nevertheless, we can divide the signing algorithm into several steps and allocate them to the corresponding code lines in Table 4.1. Furthermore, Beullens noticed that the structure in SNOVA exhibits similarities to the whipping technique in MAYO [Beu25].

SNOVA.Sign($\mathsf{seed}_{\mathsf{pk}}, \mathsf{seed}_{\mathsf{sk}}, \mathbf{digest}$)

Input: public and private seeds $(\mathsf{seed}_{\mathsf{pk}}, \mathsf{seed}_{\mathsf{sk}})$
digest of the document $\mathbf{digest} = Hash(D)$
Output: the signature sig and salt

1 : Generate $(A_\alpha, B_\alpha, Q_{\alpha 1}, Q_{\alpha 1}$ for $0 \le \alpha < l^2)$ using Algorithm 4
2 : $m \leftarrow o$
3 : Generate $(T^{12}, F_i^{11}, F_i^{12}, F_i^{21}$ for $0 \le i < m)$ using Algorithm 6
4 : $T \leftarrow \begin{pmatrix} I^{11} & T^{12} \\ 0 & I^{22} \end{pmatrix}$
5 : $\mathbf{num}_{\mathsf{sig}} \leftarrow 0$
6 : **repeat**
7 : $\quad \mathbf{num}_{\mathsf{sig}} \leftarrow \mathbf{num}_{\mathsf{sig}} + 1$
8 : $\quad$ Assign vinegar values $(X_0, \ldots, X_{v-1})$ using Algorithm 7
9 : $\quad$ Compute $(F_{i,VV}$ for $0 \le i < m)$ using Algorithm 8
10 : $\quad$ Compute $(M_{ik}$ for $0 \le i < m, 0 \le k < o)$ using Algorithm 9
11 : $\quad$ Build the augmented matrix G using Algorithm 10
12 : $\quad$ **flag_redo,**$(\tilde{X}_0, \ldots, \tilde{X}_{o-1}) \leftarrow$ **Gauss**(G)
13 : $\quad$ **if flag_redo == FALSE then**
14 : $\quad\quad \mathsf{sig} \leftarrow T(X_0, \ldots, X_{v-1}, \tilde{X}_0, \ldots, \tilde{X}_{o-1})^\top$
15 : $\quad$ **end**
16 : **until flag_redo == FALSE**;
17 : **return** (sig, salt)

Figure 4.2: Algorithm that signs the digest of a message $\mathbf{digest} = Hash(D)$ in SNOVA. The algorithm is depicted as in the SNOVA specification document [WCD+23] submitted to the call for additional digital signatures.

Note that the specifiction defines two variants of SNOVA, i.e., `SNOVA-ssk` and `SNOVA-esk`, where the former stands for 'seed-type secret key' and the latter means 'expanded secret key'. When `SNOVA-esk` is applied, the first four lines of Figure 4.2 can be disregarded, as the keys are already inserted in their expanded form.

4.1.1 Signing Strategy of UOV-based Schemes

Even though the signing algorithms of the four considered UOV-based signature schemes, shown above and in Chapter 2, appear very different at first glance, they have great similarities and the differences can be attributed in large parts to notation and style choices. However, just by observing the pseudocodes of these four signature schemes, it is hard to grasp the logic behind the algorithms. Therefore, we deduce six logical steps, which summarize the signing strategy of these signature schemes more tangible. In Table 4.1, we present the signing steps and match them with the corresponding code lines of the signing algorithms of the considered signature schemes.

Table 4.1: The first column shows the strategic steps that compose the signing algorithm of a UOV-based signature scheme. Here, the schemes UOV, MAYO, QR-UOV, and SNOVA are considered. The remaining columns localize these steps in the pseudocode of the respective signature scheme.

Signing step	UOV Figure 2.4	MAYO Figure 2.7	QR-UOV Figure 4.1	SNOVA Figure 4.2
1. Secret Key Expansion	Shifted to UOV.ExpandSK	Shifted to MAYO.ExpandSK	`Line 3-8`	`Line 1-4`
2. Generate random vinegar vector	`Line 4`	`Line 16`	`Line 10`	`Line 8`
3. Compute constant part	`Line 9`	`Line 28-30`	`Line 12`	`Line 9`
4. Compute linear part	`Line 5-7`	`Line 24-27, & 31-34`	`Line 11`	`Line 10`
5. Solve linear system	`Line 10`	`Line 37`	`Line 17`	`Line 11,12`
6. Add oil and vinegar part	`Line 11`	`Line 44`	`Line 18`	`Line 14`

The first step, i.e., the secret key expansion, can be seen as a preparation of the signing algorithm, when compressed keys are used. In this step, the seeds get expanded and the matrices needed for the subsequent steps are derived. In UOV and MAYO, the required steps are shifted to a separate algorithm. In QR-UOV and SNOVA, they are performed at the beginning of the signing algorithm. This step belongs to the more time consuming parts of the algorithm and it is prone to side-channel attacks, due to the massive amount of matrix multiplications that are necessary, as we will examine later in this section. The next step, which is the random sampling of the vinegar variable, is a common feature of all UOV-based schemes and has been subject to many fault injection attacks in the past. Step three and four serve the purpose of setting up the linear system that has to be solved during signature generation. To this end, the sampled vinegar vector is multiplied with the matrices prepared during key

expansion, to evaluate the constant and linear part of the system. Similar to the first step, the required operations are time consuming and can be attacked with power analysis. Step five entails finding a solution of the derived system via Gaussian elimination. In case this system has no solution, one has to repeat steps two to five until a solution can be found. In the last step, the found solution is mapped to an oil vector, and added to the vinegar vector. Together, they constitute the preimage, that builds the core part of the signature. This step is susceptible to both fault injection and side-channel attacks.

General transferability. From the findings above, we can conclude that the general idea behind the physical attacks can be transferred well from one UOV-based signature scheme to the other. Consider, e.g., a fault attack on the random sampling process of the vinegar vector (Step 2 in Table 4.1), or a side-channel attack on the secret values during the linear system setup (Step 3 and Step 4 in Table 4.1), or the idea to hinder the addition of the oil and vinegar values (Step 6 in Table 4.1). These approaches apply to all schemes likewise, in theory. Whether or not the practical execution of such attacks is feasible also depends on the chosen implementation. We will assess the question regarding the applicability of these attacks on current implementations in the next section.

Of course, there are also some operations that appear exclusively in one of the mentioned schemes. For instance, the multiplication with so-called emulsifier maps $\mathbf{E}_{ij}$ is a peculiarity of MAYO. Consequently, it would not be possible to transfer an attack, which targets operations on these emulsifier maps, from MAYO to other UOV-based signature schemes. So far, however, there have been no attacks that aim at such special features of the signature schemes.

4.1.2 Attack Overview

In [**A**CK25], we conduct a literature survey covering all physical attacks against UOV-based signature schemes and evaluate their applicability on the first round submission package [NIST22] of the schemes UOV, MAYO, QR-UOV, and SNOVA. The paper was submitted for publication in November 2024. At this time, NIST already announced the fourteen proposals, which advance to round 2. Hence, it was clear that the four mentioned schemes were the only multivariate schemes remaining in the process. We decided to analyze the implementations submitted to round 1, since the updated submission packages for round 2 were not published yet and due in January 2025.

In the following we present a short overview of our results regarding fault attacks and side-channel attacks. For more details about how and why certain attacks apply to the individual schemes we refer to our publication [**A**CK25], which is also shown in Appendix B.1.

Fault attack overview. In Table 4.2, we provide an overview of fault injection attacks against UOV-based signature schemes. We list the publications that introduce these attacks, indicate if they are feasible in the respective implementations, and localize the vulnerable code lines in the signing algorithm.

One fault attack, i.e., the instruction skip to prevent the addition of the oil and vinegar parts, can depend heavily on the variable handling in the

Table 4.2: Overview of existing and new fault attacks on UOV, MAYO, QR-UOV, and SNOVA. This table is adopted from [**A**CK25]. Regarding the feasibility of the attacks, we refer to the specifications submitted to the NIST call for additional signatures in mid-2023. When there are differences between the three UOV variants `uov-classic`, `uov-pkc` and `uov-pkc+skc`, deterministic and randomized MAYO or the variants `SNOVA-esk` and `SNOVA-ssk`, we list them individually in the given order. With ✓ we state that an attack is possible, while ✗ means the opposite. By ⋆ we denote that an attack is generally possible, but the technical execution is more difficult than in the initially presented attack.

Attack description	Source	Initially for	Feasible in current version	Target
Fix vinegar vector	[**A**KK+22] [**A**MS+24] [HTS11; JD24a] [KL19; SK20]	Rainbow UOV MAYO	UOV: ✓ MAYO: ✓ QR-UOV: ✓ SNOVA: ✓	`UOV.Sign Line 4` [BCC+23] `Alg.8 Line 16,18` [FIH+23] `Alg.2 Line 10` [WCD+23] `Alg.11 Line 8`
Rowhammer on oil space O	[MIS20]	LUOV	UOV: ✓ | ✓ | ⋆ MAYO: ⋆ QR-UOV: ⋆ SNOVA: ✓ | ⋆	uncompressed secret key in memory
Bit flip on stored secret matrices	[HTS11] [KL19] [FKNT22]	Rainbow UOV	UOV: ✓ | ✓ | ⋆ MAYO: ✓ | ✗ QR-UOV: ⋆ SNOVA: ✗	uncompressed secret key in memory
Prevent addition of oil and vinegar	[SM**A**+24]	MAYO	UOV: ✓ MAYO: ✓ QR-UOV: ✗ SNOVA: ✗	`UOV.Sign Line 11` [BCC+23] `Alg.8 Line 44` [FIH+23] `Alg.2 Line 18` [WCD+23] `Alg.11 Line 14`
Disturb linear system setup	[**A**CK25]	UOV	UOV: ✗ MAYO: ✓ | ✗ QR-UOV: ✗ SNOVA: ✗	`UOV.Sign Line 5-10` [BCC+23] `Alg.8 Line 22-33` [FIH+23] `Alg.2 Line 11,12` [WCD+23] `Alg.11 Line 9,10`

implementations. For instance, if the sensitive value is stored in a certain variable and another arbitrary value is added to it, then an instruction skip that avoids the addition can be critical, since it eventually reveals the sensitive value. In case the order is reversed, the attack is harmless, since it only reveals the non-sensitive value. On the other hand, we discovered that the instruction skip that targets the random sampling of the vinegar vector, affects all the listed implementations equivalently.

The other fault attacks are influenced by implementation choices that are initially not related to physical security. E.g., we showed that the question of compressed vs. uncompressed keys and randomized vs. deterministic signatures has an impact on the feasibility of these fault attacks. We will elaborate more on that in Section 4.2.

Side-channel attack overview. In Table 4.3, we provide an overview of side-channel attacks against UOV-based signature schemes. Again, we list the publications that introduce these attacks, indicate if they are feasible in the respective implementations, and localize the vulnerable code lines in the affected algorithm.

The case for side-channel attacks is less complicated. Since all considered implementations offer no designated protection against power analysis, the attacks apply to all schemes. Certainly, there are differences in the amount of effort that is required to make the power analysis work and reveal the

Table 4.3: Overview of existing and new side-channel attacks on UOV, MAYO, QR-UOV, and SNOVA. This table is adopted from [**A**CK25]. Regarding the feasibility of the attacks, we refer to the specifications submitted to the NIST call for additional signatures in mid-2023. When there is a difference between deterministic and randomized MAYO, we list them individually in the given order. With ✓ we state that an attack is possible. By ⋆ we denote that an attack is generally possible, but the technical execution is more difficult than in the initially presented attack.

Description: Power analysis ...	Source	Initially for	Feasible in	Target
of vinegar evaluation	[**A**CK+23]	UOV	UOV: ✓ MAYO: ✓ QR-UOV: ✓ SNOVA: ✓	`UOV.Sign Line 9` [BCC+23] `Alg.8 Line 29` [FIH+23] `Alg.2 Line 12` [WCD+23] `Alg.8 Line 3,4`
of secret matrix multiplication	[PSKH18] [JD24b]	Rainbow UOV MAYO	UOV: ⋆ MAYO: ✓ | ⋆ QR-UOV: ⋆ SNOVA: ⋆	`UOV.Sign Line 11` [BCC+23] `Alg.8 Line 44` [FIH+23] `Alg.2 Line 18` [WCD+23] `Alg.11 Line 14`
of linear system setup	[**A**CK25]	UOV	UOV: ✓ MAYO: ✓ QR-UOV: ✓ SNOVA: ✓	`UOV.Sign Line 7` [BCC+23] `Alg.8 Line 27,29` [FIH+23] `Alg.2 Line 11` [WCD+23] `Alg.9 Line 3,4,14,26`
of secret key expansion	[**A**CK25] [JD24b]	UOV MAYO	UOV: ✓ MAYO: ✓ QR-UOV: ✓ SNOVA: ✓	`UOV.ExpandSK Line 4` [BCC+23] `Alg.6 Line 17` [FIH+23] `Alg.2 Line 7` [WCD+23] `Alg.6 Line 6,7`
during key generation	[**A**CK25]	UOV	UOV: ✓ MAYO: ✓ QR-UOV: ✓ SNOVA: ✓	`UOV.CompactKeyGen Line 6` [BCC+23] `Alg.5 Line 16` [FIH+23] `Alg.1 Line 5` [WCD+23] `Alg.5 Line 5`

secret entries, due to parameter choices. The number of total variables n, the number of equations m, and the employed finite field $\mathbb{F}_q$, vary across the different signature schemes. Even within one scheme, often multiple parameter sets are offered. Depending on these parameters, an adversary conducting the power measurements can extract more or less information of the vulnerable subroutines, depending on the amount of multiplications that are available for analysis. Accordingly, if the underlying field is smaller, it becomes easier to select the right field element from the obtained power traces.

4.2 Implementation Guidelines and Implemented Countermeasures

In the following, we present mitigation techniques that were suggested and implemented as part of our publications [**A**CK25; SM**A**+24] to protect against the set of vulnerabilities derived above. Again, we split the attack vectors into fault injection and side-channel attacks to discuss them separately.

4.2.1 Fault-Tolerant Implementation

In the case of fault injection attacks, the setting is twofold. Besides the design of dedicated countermeasures, there are also implementation choices, which are originally made without reference to physical security, but turn out to have a large impact on the resistance of UOV-based signature schemes against

fault attacks. These implementation choices are represented by the options: compressed vs. uncompressed keys and randomized vs. deterministic signing. We explain these options briefly and argue how compressed keys and randomized signing protect against certain fault attacks from Table 4.2 and are therefore preferable in the context of fault-tolerant implementations. Subsequently, we address how the remaining fault attack vectors can be mitigated by methods based on the idea of infective computation.

Compressed keys. UOV-based signature schemes offer the possibility of working with compressed keys to tackle their main disadvantage—large key sizes. This method was already introduced by Petzoldt et al. [PTBW11]. The uncompressed UOV variant `uov-classic` stores all coefficients of the public map $\mathcal{P}$ in the expanded public key $\mathsf{epk} = \{\mathbf{P}_i^{(1)}, \mathbf{P}_i^{(2)}, \mathbf{P}_i^{(3)}\}_{i\in[m]}$. The secret key is composed of a private seed $\mathtt{seed}_{\mathsf{sk}}$, the representation matrix $\mathbf{O}$ of the oil space O, a part of the public matrices $\{\mathbf{P}_i^{(1)}\}_{i\in[m]}$, and some helper matrices $\{\mathbf{S}_i\}_{i\in[m]}$, which are needed during the signing procedure. Thus, we have $\mathsf{esk} = (\mathtt{seed}_{\mathsf{sk}}, \mathbf{O}, \{\mathbf{P}_i^{(1)}, \mathbf{S}_i\}_{i\in[m]})$.

However, $\mathbf{O}$ is generated randomly and can also be expanded from the private seed $\mathtt{seed}_{\mathsf{sk}}$. The public map $\mathcal{P}$ needs to fulfill the property that $\mathcal{P}(\mathbf{o}) = \mathbf{0}$, for all $\mathbf{o} \in O$. To this end, one can expand the submatrices $\{\mathbf{P}_i^{(1)}, \mathbf{P}_i^{(2)}\}_{i\in[m]}$ from a public seed $\mathtt{seed}_{\mathsf{pk}}$ and only has to store the values of $\{\mathbf{P}_i^{(3)}\}_{i\in[m]}$ in the public key, additionally. The remaining components from esk can also be derived from $\mathsf{csk} = \mathtt{seed}_{\mathsf{sk}}$ and $\mathsf{cpk} = (\mathtt{seed}_{\mathsf{pk}}, \{\mathbf{P}_i^{(3)}\}_{i\in[m]})$.

The UOV specification defines two more variants `uov-pkc` and `uov-pkc+skc`. As indicated by their names, the former variant uses only compressed public keys, while the latter uses both, compressed public and secret keys. Obviously, this reduces the key sizes massively at the cost of increased signing and verification time. This trade-off is depicted in Table 2.1.

In theory, the functionalities of the other mentioned UOV-based signature schemes could be composed in a similar way, such that they also constitute these three different compression variants. Still, the designers of the schemes decided to define them otherwise. According to their specification, MAYO and QR-UOV always work with compressed keys, i.e., they specify the secret key expansion subroutine as part of signing. The SNOVA specification defines two variants `SNOVA-esk` and `SNOVA-ssk`, where the former variant works with uncompressed and the latter with compressed keys.

With respect to physical security, the variants that employ compressed keys seem to be preferable across all UOV-based schemes. As can be observed from Table 4.2, there are two fault attacks that aim at altering certain secret values in the memory, namely the 'Rowhammer attack on $\mathbf{O}$' and the 'Bit flip on the secret matrices'. Both attacks rely on the possibility of flipping individual bits of the respective secret matrices. Flipping individual bits of the seed is useless, from the perspective of an adversary, since it would lead to untraceable errors due to the avalanche effect of the cryptographic hash function or extendable output function that is used to expand the seed. If the secret matrices are not stored in memory, but are expanded from the seed as part of every signature generation, this significantly reduces the time slot for an attacker to introduce the exploitable bit flips. Furthermore, it limits the insertion of persistent faults,

as each newly triggered signing procedure would overwrite introduced faults from before.

Randomized signing. Another implementation choice is the question of randomized or deterministic signing. In randomized signing external randomness is used to ensure that even if one signs the same message with the same secret key twice, one gets two different signatures. In the deterministic setting, signing the same message with the same secret key again leads to identical signatures. Currently, the four UOV-based signature schemes UOV, MAYO, QR-UOV, and SNOVA use randomized signing in their specification, while MAYO is the only one that offers both options, randomized and deterministic, according to its specification. This was not always the case for UOV-based signatures and Rainbow, for instance, opted for deterministic signing as the default setting.

There might be scenarios, where the deterministic setting is recommended, e.g., if one wants to minimize the reliance on random number generators or when reproducible signatures are desired. When it comes to resistance against physical attacks, the randomized setting is clearly preferable. In UOV-based signatures, the natural point to introduce randomness to the signing procedure is the sampling of the vinegar vector. The vinegar vector is often derived as the output of an extendable output function (XOF), like SHAKE-256. If the input to this function only depends on the message msg and the seed $\mathsf{seed}_{\mathsf{sk}}$, one generates the same vinegar vector if the message and secret key are unchanged and eventually ends up with the same signature. On the other hand, if one adds a random salt to the input of the XOF, one receives different vinegar vectors during signing, which lead to different linear systems with different solutions. The latter offers far better protection against physical attacks. This applies first and foremost to the fault attack that aims at disturbing the linear system setup. Assume deterministic signing is chosen, i.e., the same vinegar vector $\mathbf{v}$ is generated when the same message is signed twice. Then, every fault injection that alters a value during Step 3, 4, or 5 of Table 4.1 leads to a different oil vector $\mathbf{o}'$ as a result of the linear system solving. Consequently, the difference of a valid and a faulted signature $\mathbf{s} - \mathbf{s}' = (\mathbf{v} + \mathbf{o}) - (\mathbf{v} + \mathbf{o}') = \mathbf{o} - \mathbf{o}'$ reveals an oil vector. Since these steps also belong to the most time consuming steps, the time slot to inject such a fault is overwhelmingly large. Furthermore, it is not necessary that the attacker keeps track of the fault location or the error propagation, since the signing algorithm automatically generates a new oil vector, by deriving a different solution to the altered linear system. This seems extremely hard to protect, and we therefore do not recommend to use deterministic signing in applications where physical security is relevant.

Protect vinegar and oil vectors. For the remaining fault attacks of Table 4.2, i.e., reusing or zeroing the vinegar vector and preventing the addition of the oil and vinegar vector at the end of signing, we have to implement dedicated countermeasures.

Certain methods have been suggested to protect the random sampling of the vinegar vector. To avoid reusing certain variables that remain in the buffer from earlier signature generations, it was recommended to free buffers from sensitive information at the end of signing. Zeroing these buffers, however, is not desirable, since this could be exploited by argument initialization at-

tacks [SK20]. In contrast, it is more desirable to overwrite these buffers with random data [JD24a; SM**A**+24]. This countermeasure would be vulnerable to a second-order fault attack. In [**A**CK25], we suggest to instead add random data **r** to the vinegar vector **v** directly after it has been used to derive the linear system, and before it is added to the signature vector **s**, via `s += v`. Of course, the contribution of this random vector has to be removed via `s -= r`, before outputting the signature. If the addition to randomize the vinegar vector `v += r` is skipped by another fault injection, the signature output would be erroneous and useless to an attacker, since the component **r** would still be subtracted from **s**. This follows the idea of infective computation [GST12].

At the end of the signing process, the implementer has to ensure that both **v** and **o** contribute to the signature **s**. Here, the variable assignment can be crucial. In the analyzed implementation of UOV and MAYO, the vinegar part is first copied to the signature vector, and afterwards the oil part is added. Thus, a single instruction skip could remove the vinegar part completely from the signature output and reveal the sensitive oil part. In contrast, the analyzed QR-UOV and SNOVA implementations process the vinegar and oil vector together in a single function and assign the added values to the signature vector one entry after the other. This offers more resistance against instruction skips. Additionally, it is advisable to implement double checks to ensure that the finally released signature vector is not identical to the previously derived oil vector [SM**A**+24].

4.2.2 Protection against Power Analysis

The operations that are vulnerable to side-channel attacks presented in Table 4.3 are all given by matrix-vector or matrix-matrix multiplications, and eventually boil down to field multiplications in $\mathbb{F}_q$. If not protected against power analysis, these multiplications leak information about the involved factors, especially if one of the factors is already known to the attacker, since it is publicly available. The publications [**A**CK+23; JD24b; PSKH18] show how this leakage can be exploited to recover secret values, and eventually the complete secret key.

For UOV-based signature schemes no implementations were available that offer protection against power analysis. Therefore, our goal was to employ some of the well-established countermeasures, that can be applied to protect the involved secret values in these computations. In [SM**A**+24] we implemented a shuffling countermeasure to the matrix-vector multiplications in MAYO. In [**A**CK25] we present a masked implementation of UOV and MAYO.

Shuffling. The idea behind shuffling is to randomize the operation order during computation. Attackers often rely on a fixed operation order to match points of interest in a power trace to the corresponding secret values that are processed by the device at this point in time. Randomizing the execution order makes such an alignment much harder. Shuffling is especially effective against SPA and DPA, since the attacker can not rely on the same execution patterns and the misalignment causes noise in statistical attacks. However, it does not eliminate the leakage, but only adds noise, which increases the number of traces needed for a successful attack. With more effort put in trace analysis, the application of advanced statistical methods or powerful approaches like template attacks, side-channel attacks are still possible. A

benefit of shuffling is that it comes with a rather small overhead, compared to more powerful countermeasures like masking. In [SM**A**+24, Section 4.2], we explain the execution patterns we want to break by randomizing the order of the matrix-vector multiplications in MAYO.

Masking. The idea behind masking is to split sensitive intermediate variables randomly into multiple shares such that each share individually leaks no useful information to an attacker. One of the simplest and most common forms of masking is to split a secret value randomly into two additive shares $\mathbf{x} = \mathbf{x}_1 \oplus \mathbf{x}_2$. This is also called boolean first-order masking. The employed fields in UOV and MAYO are $\mathbb{F}_q = \mathbb{F}_{2^4}$ and $\mathbb{F}_q = \mathbb{F}_{2^8}$. Both of them have characteristic 2, i.e., the $\oplus$ operation is compatible with the field addition $\mathbf{x} = \mathbf{x}_1 \oplus \mathbf{x}_2 = \mathbf{x}_1 + \mathbf{x}_2$. Thus, we can mask linear operation efficiently by boolean masking, which is equivalent to field masking in this scenario. Instead of handling the secret value $\mathbf{x}$ directly, one performs the operations on both shares individually. Even if an attacker is able to observe power traces of operations on one of the shares, those values are statistically independent of the secret data. The shares are only combined at the end of the signing algorithm. In [**A**CK25, Section 5.5], we describe our masking strategy for UOV and MAYO. This includes a catalog of functions that we estimated to be vulnerable and is in line with the derived vulnerabilities we listed in Table 4.3. In [**A**CK25, Section 6.1], we present a performance comparison of our developed masked implementations for UOV and MAYO.

4.3 Future Research Directions

There are several tasks to be accomplished when it comes to the implementation security of UOV-based signature schemes.

First, protected implementations should be optimized with respect to their efficiency and stack memory consumption. As the countermeasures for power analysis attacks seem critical in this regard, the focus should lie on optimized masked implementations. In terms of security, it is desirable to develop higher-order masked implementations of the signature schemes. In related research fields, e.g., symmetric or lattice-based cryptography, attacks have been proposed even against protected implementations. We expect this to be possible in the case of multivariate signature schemes as well. Therefore, it is recommended to demonstrate the effectiveness of implemented countermeasures by leakage assessment strategies or reevaluation of the considered attack vectors.

Second, it might be worthwhile to analyze if there are any adaptations to the signature procedure that might increase the implementation security on an algorithmic level. Our results in [**A**CK25] have shown that a substantial share of the overhead in masked implementations results from protecting Step 3 and Step 4 of Table 4.1, which entail the multiplication of the vinegar vector to matrices of the expanded secret key. The vinegar vector has to be protected, since the signature is composed as $\mathbf{s} = \mathbf{v} + \mathbf{o}$. Thus, knowledge of the vinegar vector $\mathbf{v}$ in combination with the (public) signature vector $\mathbf{s}$ leads to the actual secret, an oil vector $\mathbf{o}$. Alternatively, one could add a third vector $\mathbf{e}$ to the composition, and output a signature of the form $\mathbf{s} = \mathbf{v} + \mathbf{e} + \mathbf{o}$. This would render the vinegar vector less sensitive, since its leakage would not lead to an oil

vector, if $\mathbf{e}$ is kept secret. Of course, this would imply rather big modifications to the signing algorithm and a careful security analysis of the modification is necessary. In the end, this strategy is only one possible approach of developing a more masking-friendly signature algorithm and we are curious about new concepts that will be brought forth in the future.

5

Analyzing Security Features

The research paper [**A**ST24] and [**A**DM+24], provided in Appendix C, both deal with the analysis of certain UOV-based signature schemes regarding specific security notions. The former presents mathematical attacks on MQ-Sign, a UOV-based signature scheme participating in the KpqC competition. These attacks break its EUF-CMA security. In the latter, we analyze, among others, seven UOV-based signatures regarding the security notions EO, MBS, and NR. Accordingly, we split this chapter into two parts.

In Section 5.1 we show how modifications to the EUF-CMA secure signature scheme UOV can open new attack vectors, such that this security notion is not fulfilled anymore. Therefore we analyze MQ-Sign, a variant of UOV that uses sparse coefficient matrices, i.e., sparse polynomials, in the secret map. This modification is mainly applied to reduce the size of the secret key, but it also has the potential to increase the efficiency of the scheme, since sparse polynomials can be evaluated faster at certain points. The authors presented four variants of MQ-Sign, featuring different sparsity levels, where one variant is equivalent to UOV. In [**A**ST24], we give polynomial-time key-recovery attacks for two variants, which in turn also allow polynomial-time forgery for any given message. For the third variant, we manage to present an exponential-time forgery attack with reduced complexity, showing that this variant falls short of the claimed security level.

In the second part we deal with advanced security notions beyond unforgeability. We introduce and motivate these so-called BUFF notions individually in Section 5.2. Signatures might be used in cryptographic protocols in a fashion that allows adversaries to attack them in different ways than by means of a forgery. Unforgeability is defined upon a challenge public key. The objective of the adversary is to find a valid message-signature pair under the given public key. Here, the implicit assumption is that the adversarial party has no control over that public key. In practice, however, keys themselves might be chosen maliciously. This opens the door for potential misuse, depending on the concrete use-case. We will present several real-world examples to highlight the importance of the BUFF properties, which can counter such security risks.

5.1 EUF-CMA Security of MQ-Sign

In this section, we will provide details about the MQ-Sign signature scheme, its participation in the KpqC competition [QRC21], and the impact of our attacks [**A**ST24] on certain variants of MQ-Sign.

The KpqC competition. South Korea, like many other nations, is striving to transition its cryptography systems to PQC in order to protect the nation from quantum computing threats and strengthen its cybersecurity in the long term. Therefore, South Korea's National Intelligence Service (NIS) and the National Security Research Institute launched the KpqC competition in 2021, with the goal to identify efficient and robust PQC algorithms and standardize them for national use. Furthermore, the initiators had the objective of fostering local and international technology exchange and enhancing the development of human resources in this field. The initiative is aligning with South Korea's PQC master plan, published by NIS in 2023. It provides a roadmap for the transition to PQC until 2035, which coincides with the timescales estimated by Europe and the US. Besides the already mentioned NIST PQC standardization project and the KpqC competition, we are only aware of one more PQC competition, initiated by the Chinese Association for Cryptologic Research [CAC18]. This is due to the fact that the results obtained by NIST are generally accepted in large parts of the world. As an example, the European Telecommunications Standards Institute is not running its own competition, but adheres to the one initiated by NIST.

The KpqC competition, however, is set to consist of two evaluation rounds. Nine digital signature schemes and seven public-key encryption (PKE) schemes respectively KEMs were submitted to the first evaluation round, running from November 2022 to November 2023. We list them in Table 5.1. The second evaluation round lasted from April to November 2024 and the final algorithms were selected in January 2025. We will portray the course of the competition in this section, with a focus on the UOV-based signature scheme MQ-Sign.

Table 5.1: The nine digital signature and seven PKE/KEM algorithms submitted to the KpqC competition. With * we indicate that the scheme was also submitted to NIST's call for additional digital signatures.

KpqC round 1 candidates

Digital Signatures		PKE/KEM	
Algorithm	Family	Algorithm	Family
AIMer*	MPC-in-the-head	IPCC	graph-based
Enhanced psqigRM*	code-based	Layered ROLLO-I	code-based
FIBS	isogeny-based	NTRU+	lattice-based
GCKSign	lattice-based	PALOMA	code-based
HAETAE*	lattice-based	REDOG	code-based
MQ-Sign	multivariate	SMAUG	lattice-based
NCC-Sign	lattice-based	TiGER	lattice-based
Peregrine	lattice-based		
SOLMAE	lattice-based		

Sparse polynomials in UOV. In order to understand MQ-Sign, we reconsider the central map $\mathcal{F} = (\mathcal{F}^{(1)}, \ldots, \mathcal{F}^{(m)})\colon \mathbb{F}_q^n \to \mathbb{F}_q^m$ introduced in Section 2.2 and divide the coefficients of its quadratic monomials into two sets

$$\mathcal{F}_V^{(k)}(x_1, \ldots, x_n) = \sum_{i \in V, j \in V} \alpha_{ij}^{(k)} x_i x_j$$

and

$$\mathcal{F}_{OV}^{(k)}(x_1, \ldots, x_n) = \sum_{i \in V, j \in O} \beta_{ij}^{(k)} x_i x_j,$$

where $V = \{1, \ldots, v\}$ and $O = \{v+1, \ldots, n\}$. We refer to these as the quadratic vinegar-vinegar part and the quadratic vinegar-oil part. In contrast to UOV, MQ-Sign employs inhomogeneous polynomials. The coefficients of the linear and constant terms are denoted by

$$\mathcal{F}_{L,C}^{(k)}(x_1, \ldots, x_n) = \sum_{1 \le i \le n} \gamma_i^{(k)} x_i + \delta^{(k)}.$$

They play only a minor role in our security analysis, but contribute to the secret key size, so we mention them for the sake of completeness. In total, the coefficients of the central polynomials are split into the three terms

$$\mathcal{F}^{(k)} = \mathcal{F}_V^{(k)} + \mathcal{F}_{OV}^{(k)} + \mathcal{F}_{L,C}^{(k)}.$$

The main idea behind the MQ-Sign signature scheme is to choose the central polynomial sparse, i.e., to set a certain amount of the coefficients $\alpha_{ij}^{(k)}$ and $\beta_{ij}^{(k)}$ of the quadratic parts $\mathcal{F}_V^{(k)}$ and $\mathcal{F}_{OV}^{(k)}$ to zero. As these coefficients account for the largest share of the secret key, this obviously can reduce the secret key size significantly. In the following we first present the four variants submitted by Shim et al. [SKA22] to the first round of the KpqC competition. The mathematical cryptanalysis given in [**A**ST24] concerns three of these four variants, where two variants were affected by a polynomial-time key-recovery attack. As a result, these two MQ-Sign variants were withdrawn after the first round. The third variant was subject to changes before the beginning of the second round. Subsequently, we will present the two variants, which the authors submitted to the second round of the KpqC competition.

MQ-Sign first round submission. There are many possibilities to introduce sparseness into the central map. The principal submitters of MQ-Sign [SKA22], decide to choose one with high sparseness and suggest to allow only v quadratic terms in each $\mathcal{F}_V^{(k)}$ and $\mathcal{F}_{VO}^{(k)}$, instead of $v \cdot (v+1)/2$ and $v \cdot o$, respectively. These new sparse versions of the two quadratic parts of the central map are denoted $\mathcal{F}_{V_S}^{(k)}$ and $\mathcal{F}_{VO_S}^{(k)}$, and are defined in [SKA22] as

$$\mathcal{F}_{V_S}^{(k)}(x_1, \ldots, x_v) = \sum_{i=1}^{v} \alpha_i^{(k)} x_i x_{(i+k-1(\bmod v))+1} \tag{5.1.1}$$

and

$$\mathcal{F}_{OV_S}^{(k)}(x_1, \ldots, x_n) = \sum_{i=1}^{v} \beta_i^{(k)} x_i x_{(i+k-2(\bmod m))+v+1}. \tag{5.1.2}$$

As already presented in Section 2.2, we can represent the quadratic parts of the polynomials by matrices $\mathbf{F}_V^{(k)}$ and $\mathbf{F}_{OV}^{(k)}$, storing the coefficients of the quadratic terms $\mathcal{F}_V^{(k)}$ and $\mathcal{F}_{VO}^{(k)}$. This makes also for a neat visualization of the chosen sparsity rule in $\mathcal{F}_{V_S}^{(k)}$ and $\mathcal{F}_{VO_S}^{(k)}$. As can be seen from the indices in Equations (5.1.1) and (5.1.2), the non-zero coefficients also depend on the number k of the represented equation. Exemplary, for the first quadratic vinegar-vinegar part $\mathcal{F}_{V_S}^{(1)}$ and its corresponding coefficient matrix $\mathbf{F}_{V_S}^{(1)}$, we have

$$\mathbf{F}_{V_S}^{(1)} = \begin{pmatrix} 0 & \alpha_1^{(1)} & 0 & \cdots & 0 \\ 0 & 0 & \alpha_2^{(1)} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \alpha_{v-1}^{(1)} \\ \alpha_v^{(1)} & 0 & 0 & \cdots & 0 \end{pmatrix}.$$

The non-zero entries of each equation are placed on a diagonal across the matrix and once the last column is reached, it continues on the first column of the next row. For the next equation, this diagonal is shifted one entry to the right:

$$\mathbf{F}_{V_S}^{(2)} = \begin{pmatrix} 0 & 0 & \alpha_1^{(2)} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \alpha_{v-2}^{(2)} \\ \alpha_{v-1}^{(2)} & 0 & 0 & \cdots & 0 \\ 0 & \alpha_v^{(2)} & 0 & \cdots & 0 \end{pmatrix}.$$

This pattern continues for all of the m matrices up to $\mathbf{F}_{V_S}^{(m)}$. The $\mathbf{F}_V^{(k)}$ matrices can also be represented as upper-triangular matrices, since the coefficients at the entries (i, j) and (j, i) contribute to the same monomial $x_i x_j$.

For the quadratic vinegar-oil part the modifications are very similar. The coefficient matrices have less columns than rows here (because $m < v$), but the authors again avoid any zero rows with their definition of the sparse variant. The coefficient matrix $\mathbf{F}_{OV_S}^{(1)}$ of $\mathcal{F}_{OV_S}^{(1)}$ is given by

$$\mathbf{F}_{OV_S}^{(1)} = \begin{pmatrix} \beta_1^{(1)} & 0 & \cdots & \cdots & 0 \\ 0 & \beta_2^{(1)} & \cdots & \cdots & 0 \\ \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \cdots & \beta_m^{(1)} \\ \beta_{m+1}^{(1)} & 0 & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & & 0 \\ 0 & \cdots & \beta_v^{(1)} & 0 & 0 \end{pmatrix}.$$

Again, this diagonal is shifted one entry to the right for each of the coefficient matrices, when counting through the equations.

The MQ-Sign proposal provides a parameter selection for four variants of the scheme: MQ-Sign-SS, MQ-Sign-RS, MQ-Sign-SR, and MQ-Sign-RR. The first letter S/R in the suffix specifies whether the quadratic vinegar-vinegar part is defined with sparse ($\mathcal{F}_{V_S}$) or random ($\mathcal{F}_V$) polynomials. The second letter S/R refers to the same property, but for the quadratic vinegar-oil part, also with a sparse ($\mathcal{F}_{OV_S}$) and a random ($\mathcal{F}_{OV}$) option. Note that the variant MQ-Sign-RR, which comprises two random quadratic parts, corresponds to the traditional UOV scheme, defined with inhomogenous polynomials. We present the parameters and signature size of MQ-Sign in Table 5.2, alongside the secret key size of these individual variants. If both $\mathcal{F}_{V_S}$ and $\mathcal{F}_{OV_S}$ are used, the key size reduction of the secret key is larger than one order of magnitude. If only one of the two sparse versions is applied, the secret key size is approximately halved.

Table 5.2: Secret key and signature sizes of the four MQ-Sign variants submitted to round 1 [SKA22] of the KpqC competition. The secret key size depends highly on the sparsity level of the coefficient matrices. We prove that all variants except MQ-Sign-RR fall short of the claimed security level.

Security Level	Parameter (q, v, m)	Signature size in Byte	Secret key size in Kilobyte			
			Variant MQ-Sign-xx			
			SS	RS	SR	RR
I	$(2^8, 72, 46)$	134	16	133	165	282
III	$(2^8, 112, 72)$	200	38	485	610	1 058
V	$(2^8, 148, 96)$	260	66	1 111	1 416	2 460

Algebraic attacks during round 1. In [**A**ST24], we present a polynomial-time key-recovery attack that exploits the sparseness of $\mathcal{F}_{OV_S}$. Therefore, it affects the two variants MQ-Sign-RS and MQ-Sign-SS. The attack was reported to run in 0.6 seconds for the proposed parameters for security level I, 2.3 seconds for security level III, and 6.9 seconds for security level V on a consumer grade laptop. The attack also relied on the fact that the secret transformation S was chosen to be in block matrix structure. Shortly after, Ikematsu et al. [IJY23] generalized our approach to arbitrary affine maps S. The proposed key-recovery attack is still running in polynomial-time and takes less than 30 minutes to complete, for any security level. Together, these two attacks completely break the security of the concerned variants MQ-Sign-RS and MQ-Sign-SS.

Moreover, in [**A**ST24], we introduce another algebraic attack, which exploits the sparseness and structure of $\mathcal{F}_{V_S}$ and therefore targets the variant MQ-Sign-SR. In contrast to the aforementioned attacks, this one is not a key-recovery attack, but a forgery attack, i.e., we do not compute the secret key $(\mathcal{F}, S)$, but instead show how to compute a solution vector $s \in \mathbb{F}_q^m$, with $\mathcal{P}(s) = t$ for any given $t \in \mathbb{F}_q^m$. It has exponential running time and is not practical, but it shows that also the last variant that offers a key size reduction compared to UOV falls slightly short of the claimed security level. However, this attack could be prevented quite easily by modifying the structure of $\mathcal{F}_{V_S}$.

Selected algorithms to advance to round 2. In December 2023 the KpqC team announced four digital signature schemes and four PKE/KEMs that advanced to the second and final round. We list these schemes in Table 5.3. MQ-Sign was one of the selected signature schemes that advanced to the second round. Researchers affiliated to the Eindhoven University of Technology published a detailed evaluation report [CHH+23] about all first round candidates, which provides some reasoning for the selection process. Authors were encouraged to update their submission package for the second evaluation round, which lasted from April to November 2024.

MQ-Sign second round submission. As a result of the polynomial-time key-recovery attacks mentioned above, the authors decided to remove the two variants MQ-Sign-RS and MQ-Sign-SS from the specification. The forgery attack on the variant MQ-Sign-SR was less critical, so the authors kept an approach that utilizes a sparse vinegar-vinegar part, but applied some modifications to it, in order to diminish the security concerns. They dubbed this new variant MQ-Sign-LR. We denote the modified quadratic vinegar-vinegar part $\mathcal{F}_{V_L}$. It is defined as

$$\mathcal{F}_{V_L}^{(k)}(x_1, \ldots, x_v) = \sum_{i=1}^{v} x_i L_{(i+k-1(\bmod v))+1}, \tag{5.1.3}$$

where $L_i = \sum_{j=1}^{v} \delta_{ij} x_j$ is a linear combination of the variables $\{x_1, \ldots, x_v\}$, for $i \in \{1, \ldots, v\}$. We note that this does not lead to zero coefficients in the quadratic monomials $x_i x_j$ as it was the case for $\mathcal{F}_{V_S}$. Thus, the derived coefficient matrices $\mathbf{F}_{V_L}^{(k)}$ are not sparse and we do not display them here. However, it remains quite obvious, that the v^2 elements δ_{ij} are enough to define $\mathcal{F}_{V_L}$, providing the desired key size reduction. According to Equation (5.1.3), the individual polynomials $\mathcal{F}_{V_L}^{(k)}$ of the modified quadratic vinegar-vinegar part $\mathcal{F}_{V_L}$ can be represented as

$$\begin{pmatrix} \mathcal{F}_{V_L}^{(1)} \\ \mathcal{F}_{V_L}^{(2)} \\ \vdots \\ \mathcal{F}_{V_L}^{(m)} \end{pmatrix} = \begin{pmatrix} x_1 & x_2 & \ldots & x_v \\ x_v & x_1 & \ldots & x_{v-1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{v-m+2} & x_{v-m+3} & \ldots & x_{v-m+1} \end{pmatrix} \cdot \begin{pmatrix} L_1 \\ L_2 \\ \vdots \\ L_v \end{pmatrix}.$$

Algebraic attacks during round 2. At the end of the second evaluation round, Ran and Trimoska [RT25] found a way to exploit the structure introduced in MQ-Sign-LR. They discovered a set of weak targets, for which they were able to compute preimages efficiently. The set of weak targets is large enough, to turn this into a forgery attack on any given message `msg`, by enumerating salts r until a weak target $\mathbf{t} = \mathcal{H}(\mathtt{msg}, r)$ is found. The average number of salts to try is 2^{80} for security level I, 2^{128} for security level III, and 2^{176} for security level V. Thus, the variant MQ-Sign-LR in its submitted form is not meeting the claimed security level. Again, the variant MQ-Sign-RR was not affected by the attack.

Table 5.3: The four digital signature and four PKE/KEM algorithms advancing to the second evaluation round in the KpqC competition. The schemes in green were selected for standardization. Initially, SMAUG and TiGER were two individual submissions, but got merged after the first evaluation round, because of their similarities.

KpqC round 2 candidates

Digital Signatures		PKE/KEMs	
Algorithm	Family	Algorithm	Family
AIMer	MPC-in-the-head	NTRU+	lattice-based
HAETAE	lattice-based	PALOMA	code-based
MQ-Sign	multivariate	REDOG	code-based
NCC-Sign	lattice-based	SMAUG-T	lattice-based

Selected final algorithms. In January 2025 the KpqC team announced the final algorithms and winner of the competition. Regarding digital signature schemes, AIMer and HAETAE were selected. AIMer is following the MPC-in-the-head approach, while HAETAE is a lattice-based scheme with similarities to DILITHIUM. In the case of PKE/KEMs, the lattice-based schemes NTRU+ and SMAUG-T were chosen.

5.2 Security Beyond Standard Notions and the Case of UOV-based Signatures

Signature schemes are employed in a vast number of cryptographic protocols, which might exhibit a complex structure, involve a multitude of parties, and engage various other cryptographic primitives. This increases the difficulty of assessing the protocol's security and protocol designers can often not solely rely on the basic security of the used cryptographic primitives. Thus, an in-depth security analysis of the developed protocol requires a considerable amount of time and resources to detect subtle vulnerabilities. Even though this would still be the desirable and recommendable approach, it is very optimistic to assume that such a procedure will be applied in general, and it cannot be guaranteed. Therefore, it would be beneficial to design the basic building blocks such that they achieve more advanced security features to prevent them from potential misuse. In the case of signature scheme, three different classes of advanced security notions beyond unforgeability have emerged in the recent years. In Section 5.2.1, we provide an intuition for these individual features and highlight their practical relevance. In Section 5.2.2, we present transforms that can be applied to arbitrary signature schemes to make them achieve some or all of these security features. Finally, in Section 5.2.3, we elaborate on important results discovered in [**A**DM+24] and present the case of UOV-based signature schemes in detail.

5.2.1 The Beyond Unforgeability Features

The different advanced security notions have in common that they allow maliciously chosen public keys, a concept which is not covered by EUF-CMA. In general, these crafted public keys do not need to originate from the key

generation algorithm of the signature scheme, and therefore could have no corresponding secret key. They might be constructed by an adversary to satisfy certain properties to eventually pass the verification algorithm for specific message-signature pairs. The following security notions ask to find such tupels of public keys, messages, and signatures under certain conditions depending on the concrete notion. The lack of these notions can implicate severe consequences as we will illustrate for real-world protocols. Not only for this reason, the interest in these notions increased in recent years, and when NIST announced the call for additional digital signatures [NIST22] in 2022, they declared BUFF security as a desirable feature.

Exclusive ownership (EO). Briefly summarized, EO is concerned with the question whether a signature is tied to a single public key or whether it is possible to find another (malicious) public key under which this signature also verifies. In more detail, an adversary having access to a signing oracle for a certain public key pk, is challenged to create another public key $\overline{\mathsf{pk}} \neq \mathsf{pk}$ such that one of the queried signatures sig is still valid, i.e., $\mathsf{Verify}(\overline{\mathsf{pk}}, \mathsf{msg}, \mathsf{sig}) = \mathsf{true}$. For our analysis in [**A**DM+24], we consider two versions of EO, namely strong-conservative exclusive ownership (S-CEO) and strong-destructive exclusive ownership (S-DEO). The former requires the message to be the same as in the respective query yielding sig, while the latter requires the message to be distinct from it. These notions were introduced by Cremers et al. [CDF+21]. There exists also a joint formalization of S-CEO and S-DEO, dubbed strong-universal exclusive ownership (S-UEO), where the message can be either the same or distinct from the one in the oracle query. The notions S-UEO and the even stronger variant malicious-strong-universal exclusive ownership (M-S-UEO) were put forth by Brendel et al. [BCJZ21]. We will come back to the latter in Section 5.2.3. The name *exclusive ownership* was first used by Pornin and Stern [PS05], but the underlying idea goes even back to the Duplicate-Signature Key Selection attacks by Blake-Wilson and Menezes [BM99].

Not achieving one of the EO notions clearly implies an undesired behavior of a signature scheme. If adversaries are able to find a new public key $\overline{\mathsf{pk}}$ with the described properties, they could claim ownership of certain signatures they encounter. The claimed ownership would be approved by the fact that $\mathsf{Verify}(\overline{\mathsf{pk}}, \mathsf{msg}, \mathsf{sig}) = \mathsf{true}$ indeed holds, and sig appears as a valid signature of the owner of $\overline{\mathsf{pk}}$. An early draft of a certification protocol for Let's Encrypt illustrates how this could lead to misuse in a real-world protocol. The protocol was designed to act as an automated CA. Users who wanted to obtain a certificate for their website, were asked to sign a certain challenge message and put the signature in a privileged position on the website to prove admin access. Then, the authority checks if the provided signature verifies under the user's public key with the issued challenge message, and provides the certificate, in case it does. Malicious users capable of breaking S-DEO for the underlying signature scheme are now able to request certificates for websites, which they do not own. Upon receiving the challenge message of the authority, they construct a new public key for which the existing signature verifies. Finally, the CA assumes that the malicious user provided a valid signature on the website and issues a valid certificate, even though the adversary was never able to access the website itself. The vulnerability was found by Ayer and documented in [Aye15].

An approach to fix this issue was to adapt the protocol, such that it requires the user to put the hash of its public key into the privileged position on the website, instead of the signature.

Message bound signatures (MBS). Next, we want to investigate whether a signature can be valid for more than one message. This question is dealt with in MBS, where an adversary is challenged to find a public key pk, a signature sig, and at least two distinct messages $\mathsf{msg} \neq \overline{\mathsf{msg}}$, such that the signature is valid for both messages under the given (adversarially chosen) public key, i.e., $\mathsf{Verify}(\mathsf{pk}, \mathsf{msg}, \mathsf{sig}) = \mathsf{true}$ and $\mathsf{Verify}(\mathsf{pk}, \overline{\mathsf{msg}}, \mathsf{sig}) = \mathsf{true}$. It seems natural to demand from a signature scheme that it fulfills MBS. The absence of this property would allow adversaries to switch messages after signing. This contradicts non-repudiation, one of the main features of a signature, which is ensuring that a party cannot deny their actions, such as sending a certain message or signing a specific document. The first appearance of MBS can be traced back to Stern et al. [SPMS02], who discussed it under the name *duplicate signatures*. While Jackson et al. [JCCS19] specified it formally as *non-colliding signatures*, the first game-based formalization was given by Brendel et al. [BCJZ21], who introduced the now established name *message-bound signatures*.

Non resignability (NR). The last notion NR is concerned with the question whether it is possible to provide a valid signature under another public key, given a signature for an unknown message. The exact formalization of NR is probably the most difficult one among the three properties described in this section. The reason for that is that the initial definition in [CDF+21] includes auxiliary information about the message. This models a scenario where the adversary might have access to some structural information about the message, e.g., when it is known that the message stems from a key exchange between two parties. However, if the auxiliary information contains the hash of the unknown message and the signature scheme in question uses the hash-and-sign paradigm, an adversary can just generate a new key pair and compute a new signature using this hash value. Don et al. presented this attack in [DFHS24] and also suggested a new version of NR with adaptions to the auxiliary information. In [**A**DM+24] we deal with a version of NR that avoids the auxiliary information completely, called weak non resignability (wNR). This does not depict all use cases, but for many schemes we were able to develop attacks against this notion, which also imply insecurity for any stronger versions.

Concretely, wNR assumes that an adversary is given a public key pk a signature sig, but *not* the corresponding message msg, for which sig is valid, i.e., $\mathsf{Verify}(\mathsf{pk}, \mathsf{msg}, \mathsf{sig}) = \mathsf{true}$. Then it is challenged to find a distinct public key $\overline{\mathsf{pk}}$ and a signature $\overline{\mathsf{sig}}$, such that $\mathsf{Verify}(\overline{\mathsf{pk}}, \mathsf{msg}, \overline{\mathsf{sig}}) = \mathsf{true}$. Here $\overline{\mathsf{sig}}$ is allowed to be the same signature as sig.

A signature scheme not achieving wNR is contrary to the intuition that it is necessary to know the message in order to generate a signature for this message. Moreover, Jackson et al. [JCCS19] broke authentication and collision-resistance in the Dynamically Recreatable Key (DRKey) protocol used for secure routing by using the possibility to sign unknown messages. The protocol aims at providing path integrity by requiring intermediate nodes to sign a symmetric

key which they will share with the endpoint. However, a malicious node could resign a received packet and claim that the path went through another malicious node, instead of the actual honest node. We refer to [JCCS19] for more details about the DRKey protocol and the detected attack.

Now that we defined these additional desirable security features of signature schemes, we turn to the question whether and how these notions can be satisfied. If an analyzed signature scheme does not fulfill some or all of the BUFF notions by design, it is possible to apply general transformations to it. In the next section we will discuss some of these transformations in detail.

5.2.2 Transforms Towards BUFF Security

The main results regarding transforms to achieve more advanced security notions are provided in [PS05] and [CDF+21]. The PS-1, PS-2, and PS-3 transform from [PS05] have been designed to achieve security regarding some individual features, but neither achieves full BUFF security. With full BUFF security, we imply that the security features S-CEO, S-DEO, MBS, and NR are satisfied. The BUFF-lite and the BUFF transform were developed in [CDF+21]. The BUFF-lite transform manages to achieve all features except NR, and the BUFF transform guarantees full BUFF security. In Table 5.4 we give an overview of the existing transforms. All transforms lead to some overhead. Adding the public key to the inputs of the hash function will have an impact on the efficiency of the scheme, while appending the hash output of the message and/or the public key increases the signature size. Arguably, these drawbacks are not critical, but for schemes that feature rather short signature sizes, the relative increase due to appending the hash output (64 bytes) is considerable. This might diminish the main asset of such a signature scheme and could be an impediment in use cases where short signatures are crucial. In the following, we present the PS-3 and the BUFF transform in more detail, as they will take important roles in our analysis.

Table 5.4: Comparing existing transformations regarding security of several advanced security properties. This table is based on Table 2 in [CDF+21].

Transform	Signature	S-CEO	S-DEO	MBS	NR
PS-1	$\mathsf{Sign}(\mathsf{sk}, \mathsf{msg}), \mathsf{H}(\mathsf{msg})$	✗	✓	✓	✗
PS-2	$\mathsf{Sign}(\mathsf{sk}, \mathsf{msg}), \mathsf{H}(\mathsf{pk})$	✓	✓	✗	✗
PS-3	$\mathsf{Sign}(\mathsf{sk}, \mathsf{H}(\mathsf{msg}, \mathsf{pk}))$	✗	✗	✗	✗
BUFF-lite	$\mathsf{Sign}(\mathsf{sk}, \mathsf{msg}), \mathsf{H}(\mathsf{msg}, \mathsf{pk})$	✓	✓	✓	✗
BUFF	$\mathsf{Sign}(\mathsf{sk}, \mathsf{H}(\mathsf{msg}, \mathsf{pk})), \mathsf{H}(\mathsf{msg}, \mathsf{pk})$	✓	✓	✓	✓

The PS-3 transform [PS05]. Formally defined, given a signature scheme $\Sigma = (\Sigma.\mathsf{KeyGen}, \Sigma.\mathsf{Sign}, \Sigma.\mathsf{Verify})$ and a hash function H, the PS-3 transform of Σ with respect to H is $\Sigma' = \mathsf{PS\text{-}3}[\Sigma, \mathsf{H}]$ with $\Sigma' = (\Sigma'.\mathsf{KeyGen}, \Sigma'.\mathsf{Sign}, \Sigma'.\mathsf{Verify})$ as depicted in Figure 5.1. This means, that instead of signing a message msg, one signs $h = \mathsf{H}(\mathsf{msg}, \mathsf{pk})$, the hash output of the message and the public key, to somewhat bind the used public key to the signature.

As shown in Table 5.4, the PS-3 transform is the only transform that is not appending a hash value to the signature and has therefore no impact on the

$\Sigma'.\mathsf{KeyGen}()$:

$(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \Sigma.\mathsf{KeyGen}()$

return $(\mathsf{sk}, \mathsf{pk})$

$\Sigma'.\mathsf{Sign}(\mathsf{sk}, \mathsf{msg})$:

$h \leftarrow \mathsf{H}(\mathsf{msg}, \mathsf{pk})$

$\mathsf{sig} \leftarrow\!\!\$\ \Sigma.\mathsf{Sign}(\mathsf{sk}, h)$

return sig

$\Sigma'.\mathsf{Verify}(\mathsf{pk}, \mathsf{msg}, \mathsf{sig})$:

$h \leftarrow \mathsf{H}(\mathsf{msg}, \mathsf{pk})$

return $\Sigma.\mathsf{Verify}(\mathsf{pk}, h, \mathsf{sig}) = \mathsf{true}$

Figure 5.1: The PS-3 transform $\Sigma' := \mathsf{PS\text{-}3}[\Sigma, \mathsf{H}]$ of a signature scheme $\Sigma = (\Sigma.\mathsf{KeyGen}, \Sigma.\mathsf{Sign}, \Sigma.\mathsf{Verify})$ and hash function H. The modifications applied to the plain signature scheme Σ are depicted in green boxes.

signature size. Unfortunately, it is also the only transform that does not come with any additional security guarantees for arbitrary signature schemes per se. However, it might still be the case that the transform leads to BUFF security, but this depends on properties of the respective signature scheme and needs to be analyzed carefully. We will present examples for this case in Section 5.2.3.

The BUFF transform [CDF+21]. The BUFF transform achieves all beyond unforgeability features without posing any conditions on the underlying signature scheme. Therefore, the BUFF transform can be applied in a black-box manner without further analysis or detailed knowledge of the subtle properties of the signature scheme. The description is similar to the PS-3 transform, with the difference that the hash output h is additionally appended to the signature, thereby increasing the signature size. The BUFF transform is depicted in Figure 5.2.

$\Sigma'.\mathsf{KeyGen}()$:

$(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \Sigma.\mathsf{KeyGen}()$

return $(\mathsf{sk}, \mathsf{pk})$

$\Sigma'.\mathsf{Sign}(\mathsf{sk}, \mathsf{msg})$:

$h \leftarrow \mathsf{H}(\mathsf{msg}, \mathsf{pk})$

$\mathsf{sig} \leftarrow\!\!\$\ \Sigma.\mathsf{Sign}(\mathsf{sk}, h)$

$\overline{\mathsf{sig}} \leftarrow (\mathsf{sig}, h)$

return $\overline{\mathsf{sig}}$

$\Sigma'.\mathsf{Verify}(\mathsf{pk}, \mathsf{msg}, \overline{\mathsf{sig}})$:

$(\mathsf{sig}, \hat{h}) \leftarrow \overline{\mathsf{sig}}$

$h \leftarrow \mathsf{H}(\mathsf{msg}, \mathsf{pk})$

$\mathsf{v} \leftarrow \Sigma.\mathsf{Verify}(\mathsf{pk}, h, \mathsf{sig}) = \mathsf{true}$

return $\left(\mathsf{v} = \mathsf{true} \wedge \hat{h} = h\right)$

Figure 5.2: The BUFF transform $\Sigma' := \mathsf{BUFF}[\Sigma, \mathsf{H}]$ of a signature scheme $\Sigma = (\Sigma.\mathsf{KeyGen}, \Sigma.\mathsf{Sign}, \Sigma.\mathsf{Verify})$ and hash function H. The modifications applied to the plain signature scheme Σ are depicted in orange boxes.

5.2.3 Improving the Overall Understanding of BUFF Security

In [ADM+24], we analyze 17 signature schemes of NIST's call for additional signatures regarding their BUFF security. We opt for all submitted schemes based on isogenies, lattices, codes, and multivariate polynomials, which were still considered to be EUF-CMA secure at the time of writing the paper. The complete results of our analysis can be found in Appendix C.2. Besides our concrete analysis of the individual schemes' security regarding the different advanced security notions S-CEO, S-DEO, MBS, and NR, we came across other intriguing results.

First, we noticed that some schemes, e.g., the code-based candidate CROSS, fulfill all BUFF security notions without applying any of the mentioned transforms. In this case, the achieved BUFF security is solely provided by intrinsic properties of the constructed signature scheme. This is, of course, the most preferable scenario, since the overheads mentioned in Section 5.2.2, that are caused by the transforms, can be avoided completely.

Second, most of the submitted candidates fulfill MBS security by design. Furthermore, we observed, that the investigated signature schemes fulfill all BUFF notions, when MBS security is already achieved and the PS-3 transform is applied to them. Indeed, motivated by this evidence, Düzlü et al. [DFF24] were able to prove that the lattice-based signature scheme FALCON, which initially only achieved MBS security, gains full BUFF security by using the PS-3 transform. Düzlü and Struck [DS24] showed that this implication holds in general, i.e., signature schemes satisfying (only) MBS are going to achieve full BUFF security, when the PS-3 transform is applied.[1] In particular, this is the case for all analyzed UOV-based signatures, so we will elaborate on this result in more detail later in this section.

Third, we disproved claims made by the authors of some schemes regarding BUFF security. In more detail, the submitters of the lattice-based signature scheme SQUIRRELS asserted that SQUIRRELS satisfies MBS, which we were able to disprove. Moreover, the submitters of the multivariate signature scheme VOX claimed that they fulfill all BUFF notions, since they append the public key to the message before hashing it. This would be quite similar to the changes required for the PS-3 transform, but we noticed that not the complete public key was included, which opens a door for further attacks on S-CEO, S-DEO, and NR. This illustrates that one really has to take details into account when analyzing signature schemes regarding these advanced security notions.

BUFF Security of UOV-based signature schemes. In the following we will focus on the BUFF security of UOV-based signature schemes. We present the ascertained security results of the seven analyzed multivariate signature schemes in Table 5.5. Notably, all these seven multivariate signature schemes are based on the OV approach. Additionally, we also include the UOV-based signatures Rainbow and MQ-Sign-RR in this table. Rainbow, the third round finalist of NIST's initial standardization process, was analyzed in [CDF+21]. For MQ-Sign-RR, a candidate in the KpqC as described in Section 5.1, there is no formal analysis yet, but we can determine its security regarding the BUFF notions from the findings within this section.

What immediately catches the eye is that all UOV-based signature schemes achieve MBS security. Furthermore, most of the analyzed schemes fulfill neither the EO notions nor wNR (i.e., any NR notions), whereas PROV and MQ-Sign-RR fulfill all BUFF notions. For the detailed analysis we refer to [**A**DM+24] in Appendix C.2, but we give some reasoning for these results here.

To this end, we recall that in UOV-based signatures the equation $\mathcal{P}(\mathbf{s}) = \mathbf{t}$ plays a major role, as described in Section 2.2. In this equation the solution vector $\mathbf{s}$ represents the signature of the scheme, the coefficients of the public map $\mathcal{P}$ are encoded in the public key, and the target vector $\mathbf{t}$ is derived from

[1] This holds for S-CEO and S-DEO, but not for the stronger version M-S-UEO, as we will point out later in this section.

Table 5.5: BUFF Security of several UOV-based signature schemes.

Signature scheme	Analysis	S-CEO	S-DEO	MBS	NR
Rainbow	[CDF+21]	✗	✗	✓	✗
MAYO	[ADM+24]	✗	✗	✓	✗
PROV	[ADM+24]	✓	✓	✓	✓
QR-UOV	[ADM+24]	✗	✗	✓	✗
SNOVA	[ADM+24]	✗	✗	✓	✗
TUOV	[ADM+24]	✗	✗	✓	✗
UOV	[ADM+24]	✗	✗	✓	✗
VOX	[ADM+24]	✗	✗	✓	✗
MQ-Sign-RR	This thesis	✓	✓	✓	✓

the message. During verification, $\mathcal{P}(\mathbf{s}) = \mathbf{t}'$ is evaluated and subsequently it is checked if this value is equal to the target vector $\mathbf{t}$. Thus, if a signature is supposed to be valid for two different messages msg and $\overline{\mathsf{msg}}$ as it is the case in MBS, this implies that $\mathcal{P}(\mathbf{s}) = \mathbf{t} = \bar{\mathbf{t}}$ needs to hold for the two derived target values $\mathbf{t}$ and $\bar{\mathbf{t}}$. Consequently, it is necessary to break the collision resistance of the underlying hash function, in order to attack MBS security of these schemes.

In contrast, it was relatively simple to find attacks against the EO and wNR notions for the majority of the schemes. While finding a solution $\mathbf{s}$ to a multivariate quadratic system $\mathcal{P}$, as required by forgery attacks, might be hard, it becomes less complicated to construct this system $\mathcal{P}$, such that certain properties are fulfilled, as requested by the EO and NR notions. The task of crafting such a public key, i.e., such a quadratic system with certain conditions on the solution and target vectors $\mathbf{s}$ and $\mathbf{t}$, is in general more manageable, since one has many variables to play with, namely all coefficients that define the polynomial system. PROV and MQ-Sign-RR handle these security concerns by including the public key into the derivation of the target vector $\mathbf{t}$ from the message. To be precise, they define, e.g., $\mathbf{t} = \mathsf{H}(\mathsf{msg}, \mathsf{pk})$ instead of $\mathbf{t} = \mathsf{H}(\mathsf{msg})$.[2] Then it becomes infeasible to construct a public key such that $\mathcal{P}(\mathbf{s}) = \mathbf{t}$ holds for specific vectors, since modifying the coefficients of $\mathcal{P}$ will also alter $\mathbf{t}$, therefore affecting both sides of the equation simultaneously. This strategy of including the public key as a hash input is essentially the same as applying the PS-3 transform, as can be obtained from Figure 5.1. In the next paragraph, we will see that these two schemes are no isolated cases and there is a general rule for when the PS-3 transform is enough for achieving BUFF security.

Achieving BUFF security with a lightweight transform. Motivated from the findings above, we were able to show that it is sufficient to apply the more lightweight PS-3 transform to UOV-based signature schemes to achieve full BUFF security. This is in particular interesting, since these schemes are characterized by their small signature sizes, which are depicted in Table 1.1. Using any other transform from Table 5.4 would restrain one of their main features and limit their attractiveness for real-world deployment. The additional

[2] In fact, the hash input often also includes a salt, so that the target vector is derived via $\mathbf{t} = \mathsf{H}(\mathsf{msg}, \mathsf{pk}, \mathsf{salt})$, but this has no impact on the analysis.

64 bytes due to appending the hash value to the signature would lead to a considerable relative increase in signature size, and should therefore be avoided.

Düzlü and Struck [DS24] were able to generalize the pattern described above: If a signature scheme already achieves MBS security, then it is enough to apply the PS-3 transform to additionally fulfill the notions S-CEO, S-DEO, and wNR. Here, we want to emphasize that the PS-3-transformed schemes satisfy S-CEO and S-DEO, and therefore the combined notion S-UEO, but in general not the stronger version M-S-UEO. The authors prove this by developing an attack against the M-S-UEO security of PS-3 transformed UOV. Since this suits the theme of this section very well, we briefly sketch the brilliant idea behind the M-S-UEO attack from [DS24] in the following. In M-S-UEO, the adversary is allowed to pick both public keys pk and $\overline{\mathsf{pk}}$ maliciously and find a signature sig that is valid for two messages msg and $\overline{\mathsf{msg}}$, i.e., $\mathsf{Verify}(\mathsf{pk}, \mathsf{msg}, \mathsf{sig}) = \mathsf{true}$ and $\mathsf{Verify}(\overline{\mathsf{pk}}, \overline{\mathsf{msg}}, \mathsf{sig}) = \mathsf{true}$. Let O be an (arbitrary) oil space of dimension $2m$. The public keys pk and $\overline{\mathsf{pk}}$ can be chosen in a way that the corresponding public maps $\mathcal{P}$ and $\overline{\mathcal{P}}$ both vanish on O.[3] Now, we can derive two target vectors $\mathbf{t} = \mathsf{H}(\mathsf{msg}, \mathsf{pk})$ and $\overline{\mathbf{t}} = \mathsf{H}(\overline{\mathsf{msg}}, \overline{\mathsf{pk}})$ from two arbitrary messages msg and $\overline{\mathsf{msg}}$. The overdimensioned oil space O now allows to find a solution vector $\mathbf{s}$, such that $\mathcal{P}(\mathbf{s}) = \mathbf{t}$ and $\overline{\mathcal{P}}(\mathbf{s}) = \overline{\mathbf{t}}$ hold simultaneously: After inserting a reduced vinegar vector $v \in \mathbb{F}_q^{n-2m}$, similar to the usual signing procedure of UOV, the combined linear system has $2m$ variables in $2m$ equations and is therefore efficiently solvable. For more details, we refer to [DS24, Section 3.2].

5.3 Future Research Directions

We divide this paragraph in two parts according to the two topics presented in this chapter: the use of sparse polynomials in UOV-based signatures and the achievement of the BUFF notions.

Despite the presented attacks in [**A**ST24], the employment of sparse polynomials in multivariate signature schemes remains an interesting research topic. The key-recovery attack was enabled, since some secret submatrices of coefficients were chosen sparse, which provided the attacker with additional linear equations. However, choosing the public submatrices sparse would not allow for an attack like this, but would lead to reduced public key sizes in classic, uncompressed variants, or less memory consumption when compressed keys are used. Another crucial question is: how sparse are the employed polynomials, i.e., their coefficient matrices? The forgery attack was only possible since the number of zero entries was chosen extremely large. If scheme designers opt for a more conservative approach and set only a moderate amount of coefficients to zero, a good trade-off between key size reduction and security might be achievable. There has already been a follow-up work by Ran and Trimoska [RT25], that analyzes the security of the new variant MQ-Sign-LR. The authors also found a way to exploit the structure in this variant, reducing the security claims. However, the research on sparse polynomials in UOV-based signature schemes is far from being complete. New sparse variants can be introduced and combined with schemes like MAYO and SNOVA, to further reduce public key sizes.

[3] A UOV instance with a oil space of such dimension would be insecure, but this is not of importance here, since the challenge for the adversary is only to craft two public keys with the required properties.

Providing a security reduction of a newly designed sparse variant to the case of random polynomials would be desirable, and important to increase the trust in these optimized variants.

Currently, we do not see any open question when it comes to the BUFF notions of UOV-based signature schemes. As elaborated above, the plain schemes do only satisfy MBS security. The PS-3 transform, as intrinsically applied by PROV and MQ-Sign-RR, offers a lightweight solution to achieve all remaining BUFF notions, except M-S-UEO, as proven by Düzlü and Struck in [DS24]. If one wants to fulfill this additional feature as well, it is necessary to apply the BUFF transform, which increases the signature size by the length of a hash output. To this date, we are not aware of any use case, where a protocol relies on M-S-UEO security. The BUFF security of newly proposed UOV-based schemes needs to be reassessed in case major modifications are applied to the scheme, which might impact the security analysis. This would be the case if the method of deriving the target value $\mathbf{t}$ or the strategy of verifying a signature would change fundamentally. Until, there are no indications for this to happen.

6

Conclusion

The strive for secure and efficient PQC algorithms is not yet complete. The third round of NIST's PQC standardization process ended in July 2022 with the selection of one lattice-based KEM, two lattice-based signature schemes and one hash-based signature scheme. From this point, the standardization effort followed two different paths. For KEMs, the process continued with a fourth round, including three code-based and one isogeny-based KEM. During the fourth round, the isogeny-based candidate got broken completely and in the end, the code-based candidate HQC got selected for standardization in March 2025. For signature schemes, a whole new selection process started in 2023, with 14 signature schemes remaining in the process at the time of writing this thesis in May 2025. Among them are four UOV-based signature schemes. These schemes are of great interest for the cryptographic community, since they offer the best combined performance numbers with respect to signature size, signing time, and verification time. Thus, they would not only present a valuable alternative, in case the security of lattice-based schemes is harmed by cryptanalytic breakthroughs, but even represent the preferable choice in certain applications, where short signatures and efficient signing times are crucial.

There are several research branches that need to be addressed in order to raise the trust in UOV-based signature schemes and get them ready for standardization and practical deployment. First and foremost, further mathematical in-depth security analysis for the currently considered UOV-based schemes is needed. The trust in multivariate cryptography is weakened due to the breaks of Rainbow, LUOV, GeMSS, and many more variations of multivariate schemes. Only profound cryptanalysis of the schemes and their underlying assumptions can either strengthen the confidence or reveal further vulnerabilities. This also holds for the numerous attempts to reduce the public key size of UOV-based schemes, either by introducing modifications to the signature schemes itself or by utilizing sparse instead of random equations. Many of them proved to be insecure, since they were vulnerable to individualized attacks or more general approaches such as rank attacks.

Another hurdle that needs to be tackled is the resistance of UOV-based schemes against physical attacks. This thesis has shown that the available implementations are all vulnerable to a substantial amount of fault injection and side-channel attacks. While we believe that fault tolerance can be improved

by dedicated countermeasures without dramatically impacting efficiency, the case for side-channel attacks and especially power analysis seems more severe. Established countermeasures like shuffling and masking can be implemented in a quite straightforward way, due to the many linear operations present in the cryptographic algorithms. However, in particular masking introduces quite a severe overhead and impairs the efficiency of the scheme. Even more, it is not guaranteed that these countermeasures will prevent more sophisticated and elaborate attacks with advanced analysis methods and deep-learning approaches. Further research on both, optimized protected implementations as well as newly designed attacks to break or analyze these protections, is mandatory to evaluate how the physical attack resistance of UOV-based signature schemes compares to other families, e.g., code-based and lattice-based schemes.

Bibliography

[ARR03] Dakshi Agrawal, Josyula R. Rao, and Pankaj Rohatgi. "Multi-channel Attacks". In: *Cryptographic Hardware and Embedded Systems - CHES 2003, 5th International Workshop, Cologne, Germany, September 8-10, 2003, Proceedings*. 2003. DOI: 10.1007/978-3-540-45238-6_2.

[ACDG03] Mehdi-Laurent Akkar, Nicolas T. Courtois, Romain Duteuil, and Louis Goubin. "A Fast and Secure Implementation of Sflash". In: *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings*. 2003. DOI: 10.1007/3-540-36288-6_20.

[AP13] Nadhem J. AlFardan and Kenneth G. Paterson. "Lucky Thirteen: Breaking the TLS and DTLS Record Protocols". In: *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*. 2013. DOI: 10.1109/SP.2013.42.

[**A**CK25] Thomas Aulbach, Fabio Campos, and Juliane Krämer. "SoK: On the Physical Security of UOV-based Signature Schemes". In: *International Conference on Post-Quantum Cryptography*. Springer. 2025.

[**A**CK+23] Thomas Aulbach, Fabio Campos, Juliane Krämer, Simona Samardjiska, and Marc Stöttinger. "Separating Oil and Vinegar with a Single Trace: Side-Channel Assisted Kipnis-Shamir Attack on UOV". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 3 (2023).

[**A**DM+24] Thomas Aulbach, Samed Düzlü, Michael Meyer, Patrick Struck, and Maximiliane Weishäupl. "Hash Your Keys Before Signing: BUFF Security of the Additional NIST PQC Signatures". In: *International Conference on Post-Quantum Cryptography*. Springer. 2024.

[**A**KK+22] Thomas Aulbach, Tobias Kovats, Juliane Krämer, and Soundes Marzougui. "Recovering Rainbow's Secret Key with a First-Order Fault Attack". In: *International Conference on Cryptology in Africa*. Springer. 2022.

[**A**MS+24] Thomas Aulbach, Soundes Marzougui, Jean-Pierre Seifert, and Vincent Quentin Ulitzsch. "MAYo or MAY-not: Exploring implementation security of the post-quantum signature scheme MAYO against physical attacks". In: *2024 Workshop on Fault Detection and Tolerance in Cryptography (FDTC)*. IEEE. 2024.

[**A**ST24] Thomas Aulbach, Simona Samardjiska, and Monika Trimoska. "Practical Key-Recovery Attack on MQ-Sign and More". In: *International Conference on Post-Quantum Cryptography*. Springer. 2024.

[Aye15] Ayer, Andrew. *Duplicate Signature Key Selection Attack in Let's Encrypt*. `https://www.agwa.name/blog/post/duplicate_signature_key_selection_attack_in_lets_encrypt`. 2015.

[BPO+20] Florian Bache, Clara Paglialonga, Tobias Oder, Tobias Schneider, and Tim Güneysu. "High-Speed Masking for Polynomial Comparison in Lattice-based KEMs". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 3 (2020). DOI: `10.13154/TCHES.V2020.I3.483-507`.

[BV24] Gustavo Banegas and Ricardo Villanueva-Polanco. "A Fault Analysis on SNOVA". In: *Cryptology ePrint Archive, Paper 2024/1883* (2024). `https://eprint.iacr.org/2024/1883`.

[BDH+97] Feng Bao, Robert H. Deng, Yongfei Han, Albert B. Jeng, Arcot Desai Narasimhalu, and Teow-Hin Ngair. "Breaking Public Key Cryptosystems on Tamper Resistant Devices in the Presence of Transient Faults". In: *Security Protocols, 5th International Workshop, Paris, France, April 7-9, 1997, Proceedings*. 1997. DOI: `10.1007/BFB0028164`.

[BCN+06] Hagai Bar-El, Hamid Choukri, David Naccache, Michael Tunstall, and Claire Whelan. "The Sorcerer's Apprentice Guide to Fault Attacks". In: *Proc. IEEE* 2 (2006). DOI: `10.1109/JPROC.2005.862424`.

[BBE+18] Gilles Barthe, Sonia Belaid, Thomas Espitau, Pierre-Alain Fouque, Benjamin Grégoire, Mélissa Rossi, and Mehdi Tibouchi. "Masking the GLP Lattice-Based Signature Scheme at Any Order". In: *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*. 2018. DOI: `10.1007/978-3-319-78375-8\_12`.

[BBE+19] Gilles Barthe, Sonia Belaid, Thomas Espitau, Pierre-Alain Fouque, Mélissa Rossi, and Mehdi Tibouchi. "GALACTICS: Gaussian Sampling for Lattice-Based Constant- Time Implementation of Cryptographic Signatures, Revisited". In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*. 2019. DOI: `10.1145/3319535.3363223`.

[BBB+25] Daniel J. Bernstein, Karthikeyan Bhargavan, Shivam Bhasin, Anupam Chattopadhyay, Tee Kiah Chia, Matthias J. Kannwischer, Franziskus Kiefer, Thales B. Paiva, Prasanna Ravi, and Goutam Tamvada. "KyberSlash: Exploiting secret-dependent division timings in Kyber implementations". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2 (2025). DOI: 10.46586/TCHES.V2025.I2.209-234.

[BDL+11] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. "High-Speed High-Security Signatures". In: *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*. 2011. DOI: 10.1007/978-3-642-23951-9_9.

[BFP11] Luk Bettale, Jean-Charles Faugère, and Ludovic Perret. "Cryptanalysis of Multivariate and Odd-Characteristic HFE Variants". In: *Public Key Cryptography - PKC 2011 - 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy, March 6-9, 2011. Proceedings*. 2011. DOI: 10.1007/978-3-642-19379-8_27.

[Beu20] Ward Beullens. "Sigma Protocols for MQ, PKP and SIS, and Fishy Signature Schemes". In: *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part III*. 2020. DOI: 10.1007/978-3-030-45727-3_7.

[Beu21a] Ward Beullens. "Improved Cryptanalysis of UOV and Rainbow". In: *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part I*. 2021. DOI: 10.1007/978-3-030-77870-5_13.

[Beu21b] Ward Beullens. "MAYO: Practical Post-quantum Signatures from Oil-and-Vinegar Maps". In: *Selected Areas in Cryptography - 28th International Conference, SAC 2021, Virtual Event, September 29 - October 1, 2021, Revised Selected Papers*. 2021. DOI: 10.1007/978-3-030-99277-4_17.

[Beu22] Ward Beullens. "Breaking Rainbow Takes a Weekend on a Laptop". In: *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part II*. 2022. DOI: 10.1007/978-3-031-15979-4_16.

[Beu25] Ward Beullens. "Improved Cryptanalysis of SNOVA". In: *Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4-8, 2025, Proceedings, Part VI*. 2025. DOI: 10.1007/978-3-031-91095-1_10.

[BCC+23] Ward Beullens, Fabio Campos, Sophía Celi, Basil Hess, and Matthias J. Kannwischer. *MAYO*. Tech. rep. National Institute of Standards and Technology, 2023.

[BCD+23] Ward Beullens, Ming-Shing Chen, Jintai Ding, Boru Gong, Matthias J. Kannwischer, Jacques Patarin, Bo-Yuan Peng, Dieter Schmidt, Cheng-Jhih Shih, Chengdong Tao, and Bo-Yin Yang. *UOV*. Tech. rep. National Institute of Standards and Technology, 2023.

[BP17] Ward Beullens and Bart Preneel. "Field Lifting for Smaller UOV Public Keys". In: *Progress in Cryptology - INDOCRYPT 2017 - 18th International Conference on Cryptology in India, Chennai, India, December 10-13, 2017, Proceedings*. 2017. DOI: 10.1007/978-3-319-71667-1_12.

[BPSV19] Ward Beullens, Bart Preneel, Alan Szepieniec, and Frederik Vercauteren. *LUOV*. Tech. rep. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-2-submissions. National Institute of Standards and Technology, 2019.

[BDH+21] Shivam Bhasin, Jan-Pieter D'Anvers, Daniel Heinz, Thomas Pöppelmann, and Michiel Van Beirendonck. "Attacking and Defending Masked Polynomial Comparison for Lattice-Based Cryptography". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 3 (2021). DOI: 10.46586/TCHES.V2021.I3.334-359.

[BMM00] Ingrid Biehl, Bernd Meyer, and Volker Müller. "Differential Fault Attacks on Elliptic Curve Cryptosystems". In: *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*. 2000. DOI: 10.1007/3-540-44598-6_8.

[BS97] Eli Biham and Adi Shamir. "Differential Fault Analysis of Secret Key Cryptosystems". In: *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*. 1997. DOI: 10.1007/BFB0052259.

[BBK+17] Nina Bindel, Johannes Buchmann, Juliane Krämer, Heiko Mantel, Johannes Schickel, and Alexandra Weber. "Bounding the Cache-Side-Channel Leakage of Lattice-Based Signature Schemes Using Program Semantics". In: *Foundations and Practice of Security - 10th International Symposium, FPS 2017, Nancy, France, October 23-25, 2017, Revised Selected Papers*. 2017. DOI: 10.1007/978-3-319-75650-9_15.

[BM99] Simon Blake-Wilson and Alfred Menezes. "Unknown Key-Share Attacks on the Station-to-Station (STS) Protocol". In: *Public Key Cryptography, Second International Workshop on Practice and Theory in Public Key Cryptography, PKC '99, Kamakura, Japan, March 1-3, 1999, Proceedings*. 1999. DOI: 10.1007/3-540-49162-7_12.

[BS03] Johannes Blömer and Jean-Pierre Seifert. "Fault Based Cryptanalysis of the Advanced Encryption Standard (AES)". In: *Financial Cryptography, 7th International Conference, FC 2003, Guadeloupe, French West Indies, January 27-30, 2003, Revised Papers*. 2003. DOI: 10.1007/978-3-540-45126-6_12.

[BDL97] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. "On the Importance of Checking Cryptographic Protocols for Faults (Extended Abstract)". In: *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*. 1997. DOI: 10.1007/3-540-69053-0_4.

[BWP05] An Braeken, Christopher Wolf, and Bart Preneel. "A Study of the Security of Unbalanced Oil and Vinegar Signature Schemes". In: *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*. 2005. DOI: 10.1007/978-3-540-30574-3_4.

[BCJZ21] Jacqueline Brendel, Cas Cremers, Dennis Jackson, and Mang Zhao. "The Provable Security of Ed25519: Theory and Practice". In: *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*. 2021. DOI: 10.1109/SP40001.2021.00042.

[BBPS24] Pierre Briaud, Maxime Bros, Ray A. Perlner, and Daniel Smith-Tone. "Practical Attack on All Parameters of the DME Signature Scheme". In: *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part VI*. 2024. DOI: 10.1007/978-3-031-58754-2_1.

[BHLY16] Leon Groot Bruinderink, Andreas Hülsing, Tanja Lange, and Yuval Yarom. "Flush, Gauss, and Reload - A Cache Attack on the BLISS Lattice-Based Signature Scheme". In: *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*. 2016. DOI: 10.1007/978-3-662-53140-2_16.

[BP18] Leon Groot Bruinderink and Peter Pessl. "Differential Fault Attacks on Deterministic Lattice Signatures". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 3 (2018). DOI: 10.13154/TCHES.V2018.I3.21-43.

[BSI24] Bundesamt für Sicherheit in der Informationstechnik. *Status of quantum computer development*. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Studien/Quantencomputer/Entwicklungstand_QC_V_2_1.pdf. 2024.

[CSV17] Daniel Cabarcas, Daniel Smith-Tone, and Javier A. Verbel. "Key Recovery Attack for ZHFE". In: *Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings*. 2017. DOI: 10.1007/978-3-319-59879-6_17.

[CFM+20] Antoine Casanova, Jean-Charles Faugère, Gilles Macario-Rat, Jacques Patarin, Ludovic Perret, and Jocelyn Ryckeghem. *GeMSS*. Tech. rep. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-

standardization/round-3-submissions. National Institute of Standards and Technology, 2020.

[CHR+16] Ming-Shing Chen, Andreas Hülsing, Joost Rijneveld, Simona Samardjiska, and Peter Schwabe. "From 5-Pass *MQ* -Based Identification to *MQ* -Based Signatures". In: *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II*. 2016. DOI: 10.1007/978-3-662-53890-6_5.

[CVM+21] Zitai Chen, Georgios Vasilakis, Kit Murdock, Edward Dean, David F. Oswald, and Flavio D. Garcia. "VoltPillager: Hardware-based fault injection attacks against Intel SGX Enclaves using the SVID voltage scaling interface". In: *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*. 2021.

[CAC18] Chinese Association for Cryptologic Research (CACR). *CACR post-quantum competition*. https://www.cacrnet.org.cn/site/content/854.html. 2018.

[CHH+23] Jolijn Cottaar, Kathrin Hövelmanns, Andreas Hülsing, Tanja Lange, Mohammad Mahzoun, Alex Pellegrini, Alberto Ravagnani, Sven Schäge, Monika Trimoska, and Benne de Weger. "Report on evaluation of KpqC candidates". In: *Cryptology ePrint Archive, Paper 2023/1853* (2023). https://eprint.iacr.org/2023/1853.

[CDF+21] Cas Cremers, Samed Düzlü, Rune Fiedler, Marc Fischlin, and Christian Janson. "BUFFing signature schemes beyond unforgeability and the case of post-quantum signatures". In: *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*. 2021. DOI: 10.1109/SP40001.2021.00093.

[DH76] Whitfield Diffie and Martin E. Hellman. "New directions in cryptography". In: *IEEE Trans. Inf. Theory* 6 (1976). DOI: 10.1109/TIT.1976.1055638.

[DCP+20] Jintai Ding, Ming-Shing Chen, Albrecht Petzoldt, Dieter Schmidt, Bo-Yin Yang, Matthias J. Kannwischer, and Jacques Patarin. *Rainbow*. Tech. rep. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions. National Institute of Standards and Technology, 2020.

[DDS+20] Jintai Ding, Joshua Deaton, Kurt Schmidt, Vishakha, and Zheng Zhang. "Cryptanalysis of the Lifted Unbalanced Oil Vinegar Signature Scheme". In: *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III*. 2020. DOI: 10.1007/978-3-030-56877-1_10.

[DGG+23] Jintai Ding, Boru Gong, Hao Guo, Xiaoou He, Yi Jin, Yuansheng Pan, Dieter Schmidt, Chengdong Tao, Danli Xie, and Ziyu Yang Bo-Yin Zhao. *TUOV*. Tech. rep. National Institute of Standards and Technology, 2023.

[DS05a] Jintai Ding and Dieter Schmidt. "Rainbow, a New Multivariable Polynomial Signature Scheme". In: *Applied Cryptography and Network Security, Third International Conference, ACNS 2005, New York, NY, USA, June 7-10, 2005, Proceedings*. 2005. DOI: 10.1007/11496137_12.

[DS05b] Jintai Ding and Dieter Schmidt. "Rainbow, a New Multivariable Polynomial Signature Scheme". In: *Applied Cryptography and Network Security, Third International Conference, ACNS 2005, New York, NY, USA, June 7-10, 2005, Proceedings*. 2005. DOI: 10.1007/11496137_12.

[DFHS24] Jelle Don, Serge Fehr, Yu-Hsuan Huang, and Patrick Struck. "On the (In)Security of the BUFF Transform". In: *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part I*. 2024. DOI: 10.1007/978-3-031-68376-3_8.

[DFSS07] Vivien Dubois, Pierre-Alain Fouque, Adi Shamir, and Jacques Stern. "Practical Cryptanalysis of SFLASH". In: *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*. 2007. DOI: 10.1007/978-3-540-74143-5_1.

[DFF24] Samed Düzlü, Rune Fiedler, and Marc Fischlin. "BUFFing FALCON Without Increasing the Signature Size". In: *Selected Areas in Cryptography - SAC 2024 - 31st International Conference, Montreal, QC, Canada, August 28-30, 2024, Revised Selected Papers, Part I*. 2024. DOI: 10.1007/978-3-031-82852-2_6.

[DS24] Samed Düzlü and Patrick Struck. "The Role of Message-Bound Signatures for the Beyond UnForgeability Features and Weak Keys". In: *Information Security - 27th International Conference, ISC 2024, Arlington, VA, USA, October 23-25, 2024, Proceedings, Part II*. 2024. DOI: 10.1007/978-3-031-75764-8_4.

[Eck85] Wim van Eck. "Electromagnetic radiation from video display units: An eavesdropping risk?" In: *Comput. Secur.* 4 (1985). DOI: 10.1016/0167-4048(85)90046-X.

[ECR25] ECRYPT. *eBACS: ECRYPT Benchmarking of Cryptographic Systems*. https://bench.cr.yp.to/results-sign.html. 2025. (Visited on 05/24/2025).

[EFGT17] Thomas Espitau, Pierre-Alain Fouque, Benoit Gérard, and Mehdi Tibouchi. "Side-Channel Attacks on BLISS Lattice-Based Signatures: Exploiting Branch Tracing against strongSwan and Electromagnetic Emanations in Microcontrollers". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*. 2017. DOI: 10.1145/3133956.3134028.

[FS86] Amos Fiat and Adi Shamir. "How to prove yourself: Practical solutions to identification and signature problems". In: *Conference on the theory and application of cryptographic techniques*. Springer. 1986.

[FGS05] Pierre-Alain Fouque, Louis Granboulan, and Jacques Stern. "Differential Cryptanalysis for Multivariate Schemes". In: *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*. 2005. DOI: 10.1007/11426639_20.

[FY79] Aviezri S. Fraenkel and Yaacov Yesha. "Complexity of problems in games, graphs and algebraic equations". In: *Discrete Applied Mathematics* 1-2 (1979).

[FI23] Hiroki Furue and Yasuhiko Ikematsu. "A New Security Analysis Against MAYO and QR-UOV Using Rectangular MinRank Attack". In: *Advances in Information and Computer Security - 18th International Workshop on Security, IWSEC 2023, Yokohama, Japan, August 29-31, 2023, Proceedings*. 2023. DOI: 10.1007/978-3-031-41326-1_6.

[FI25] Hiroki Furue and Yasuhiko Ikematsu. "A New Cryptanalysis Against UOV-Based Variants MAYO, QR-UOV and VOX". In: *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 3 (2025).

[FIH+23] Hiroki Furue, Yasuhiko Ikematsu, Fumitaka Hoshino, Tsuyoshi Takagi, Kan Yasuda, Toshiyuki Miyazawa, Tsunekazu Saito, and Akira Nagai. *QR-UOV*. Tech. rep. National Institute of Standards and Technology, 2023.

[FIKT21] Hiroki Furue, Yasuhiko Ikematsu, Yutaro Kiyomura, and Tsuyoshi Takagi. "A New Variant of Unbalanced Oil and Vinegar Using Quotient Ring: QR-UOV". In: *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part IV*. 2021. DOI: 10.1007/978-3-030-92068-5_7.

[FKNT22] Hiroki Furue, Yutaro Kiyomura, Tatsuya Nagasawa, and Tsuyoshi Takagi. "A New Fault Attack on UOV Multivariate Signature Scheme". In: *Post-Quantum Cryptography - 13th International Workshop, PQCrypto 2022, Virtual Event, September 28-30, 2022, Proceedings*. 2022. DOI: 10.1007/978-3-031-17234-2_7.

[Gam85] Taher El Gamal. "A public key cryptosystem and a signature scheme based on discrete logarithms". In: *IEEE Trans. Inf. Theory* 4 (1985). DOI: 10.1109/TIT.1985.1057074.

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. "Electromagnetic Analysis: Concrete Results". In: *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*. 2001. DOI: 10.1007/3-540-44709-1_21.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and intractability: A Guide to the Theory of NP-Completeness*. 1979.

[GST12] Benedikt Gierlichs, Jörn-Marc Schmidt, and Michael Tunstall. "Infective Computation and Dummy Rounds: Fault Protection for Block Ciphers without Check-before-Output". In: *Progress in Cryptology - LATINCRYPT 2012 - 2nd International Conference on Cryptology and Information Security in Latin America, Santiago, Chile, October 7-10, 2012. Proceedings*. 2012. DOI: 10.1007/978-3-642-33481-8_17.

[GT04] Christophe Giraud and Hugues Thiebeauld. "A Survey on Fault Attacks". In: *Smart Card Research and Advanced Applications VI, IFIP 18th World Computer Congress, TC8/WG8.8 & TC11/WG11.2 Sixth International Conference on Smart Card Research and Advanced Applications (CARDIS), 22-27 August 2004, Toulouse, France*. 2004. DOI: 10.1007/1-4020-8147-2_11.

[Goo23] Google Quantum AI. *Our quantum computing roadmap*. https://quantumai.google/roadmap. 2023. (Visited on 01/13/2025).

[GCF+23] Louis Goubin, Benoît Cogliati, Jean-Charles Faugére, Pierre-Alain Fouque, Robin Larrieu, Gilles Macario-Rat, Brice Minaud, and Jacques Patarin. *PROV*. Tech. rep. National Institute of Standards and Technology, 2023.

[GC00] Louis Goubin and Nicolas T. Courtois. "Cryptanalysis of the TTM Cryptosystem". In: *Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings*. 2000. DOI: 10.1007/3-540-44448-3_4.

[GJP+24] Hao Guo, Yi Jin, Yuansheng Pan, Xiaoou He, Boru Gong, and Jintai Ding. "Practical and Theoretical Cryptanalysis of VOX". In: *International Conference on Post-Quantum Cryptography*. Springer. 2024.

[HIT18] Yasufumi Hashimoto, Yasuhiko Ikematsu, and Tsuyoshi Takagi. "Chosen message attack on multivariate signature ELSA at Asiacrypt 2017". In: *Advances in Information and Computer Security: 13th International Workshop on Security, IWSEC 2018, Sendai, Japan, September 3-5, 2018, Proceedings 13*. Springer. 2018.

[HTS11] Yasufumi Hashimoto, Tsuyoshi Takagi, and Kouichi Sakurai. "General Fault Attacks on Multivariate Public Key Cryptosystems". In: *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29 - December 2, 2011. Proceedings*. 2011. DOI: 10.1007/978-3-642-25405-5_1.

[IBM24] IBM Quantum. *Technology for the quantum future*. https://www.ibm.com/quantum/technology. 2024. (Visited on 01/13/2025).

[IJY23] Yasuhiko Ikematsu, Hyungrok Jo, and Takanori Yasuda. "A Security Analysis on MQ-Sign". In: *Information Security Applications - 24th International Conference, WISA 2023, Jeju Island, South Korea, August 23-25, 2023, Revised Selected Papers*. 2023. DOI: 10.1007/978-981-99-8024-6_4.

[JCCS19] Dennis Jackson, Cas Cremers, Katriel Cohn-Gordon, and Ralf Sasse. "Seems Legit: Automated Analysis of Subtle Attacks on Protocols that Use Signatures". In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*. 2019. DOI: 10.1145/3319535.3339813.

[Jaq24] Samuel Jaques. *Landscape of quantum computing in 2024*. https://sam-jaques.appspot.com/quantum_landscape_2024. 2024. (Visited on 01/13/2025).

[JD24a] Sönke Jendral and Elena Dubrova. "MAYO Key Recovery by Fixing Vinegar Seeds". In: *IACR Commun. Cryptol.* 4 (2024). DOI: 10.62056/AB0LJBKRZ.

[JD24b] Sönke Jendral and Elena Dubrova. "Single-trace side-channel attacks on MAYO exploiting leaky modular multiplication". In: *Cryptology ePrint Archive, Paper 2024/1850* (2024). https://eprint.iacr.org/2024/1850.

[KAA22] Emre Karabulut, Erdem Alkim, and Aydin Aysu. "Efficient, Flexible, and Constant-Time Gaussian Sampling Hardware for Lattice Cryptography". In: *IEEE Trans. Computers* 8 (2022). DOI: 10.1109/TC.2021.3107729.

[KL14] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. 2014.

[KDK+14] Yoongu Kim, Ross Daly, Jeremie S. Kim, Chris Fallin, Ji-Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. "Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors". In: *ACM/IEEE 41st International Symposium on Computer Architecture, ISCA 2014, Minneapolis, MN, USA, June 14-18, 2014*. 2014. DOI: 10.1109/ISCA.2014.6853210.

[KPG99] Aviad Kipnis, Jacques Patarin, and Louis Goubin. "Unbalanced Oil and Vinegar Signature Schemes". In: *Advances in Cryptology - EUROCRYPT*. 1999. DOI: 10.1007/3-540-48910-X_15.

[KS98] Aviad Kipnis and Adi Shamir. "Cryptanalysis of the Oil & Vinegar Signature Scheme". In: *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*. 1998. DOI: 10.1007/BFB0055733.

[Kob87] Neal Koblitz. "Elliptic curve cryptosystems". In: *Mathematics of computation* 177 (1987).

[KHF+19] Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. "Spectre Attacks: Exploiting Speculative Execution". In: *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*. 2019. DOI: 10.1109/SP.2019.00002.

[Koc96] Paul C. Kocher. "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems". In: *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*. 1996. DOI: 10.1007/3-540-68697-5_9.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. "Differential Power Analysis". In: *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*. 1999. DOI: 10.1007/3-540-48405-1_25.

[KPLG24] Elisabeth Krahmer, Peter Pessl, Georg Land, and Tim Güneysu. "Correction Fault Attacks on Randomized CRYSTALS-Dilithium". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 3 (2024). DOI: 10.46586/TCHES.V2024.I3.174-199.

[KL19] Juliane Krämer and Mirjam Loiero. "Fault Attacks on UOV and Rainbow". In: *Constructive Side-Channel Analysis and Secure Design - 10th International Workshop, COSADE 2019, Darmstadt, Germany, April 3-5, 2019, Proceedings*. 2019. DOI: 10.1007/978-3-030-16350-1_11.

[LSG+18] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Anders Fogh, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. "Meltdown: Reading Kernel Memory from User Space". In: *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*. 2018.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks - revealing the secrets of smart cards*. 2007.

[MI88] Tsutomu Matsumoto and Hideki Imai. "Public Quadratic Polynominal-Tuples for Efficient Signature-Verification and Message-Encryption". In: *Advances in Cryptology - EUROCRYPT*. 1988. DOI: 10.1007/3-540-45961-8_39.

[MBA+21] Robert Merget, Marcus Brinkmann, Nimrod Aviram, Juraj Somorovsky, Johannes Mittmann, and Jörg Schwenk. "Raccoon Attack: Finding and Exploiting Most-Significant-Bit-Oracles in TLS-DH(E)". In: *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*. 2021.

[MDS99] Thomas S. Messerges, Ezzy A. Dabbish, and Robert H. Sloan. "Power Analysis Attacks of Modular Exponentiation in Smartcards". In: *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, Proceedings*. 1999. DOI: 10.1007/3-540-48059-5_14.

[Mil85] Victor S. Miller. "Use of Elliptic Curves in Cryptography". In: *Advances in Cryptology - CRYPTO '85, Santa Barbara, California, USA, August 18-22, 1985, Proceedings*. 1985. DOI: 10.1007/3-540-39799-X_31.

[MPS14] Dustin Moody, Ray A. Perlner, and Daniel Smith-Tone. "An Asymptotically Optimal Structural Attack on the ABC Multivariate Encryption Scheme". In: *Post-Quantum Cryptography - 6th International Workshop, PQCrypto 2014, Waterloo, ON, Canada, October 1-3, 2014. Proceedings*. 2014. DOI: 10.1007/978-3-319-11659-4_11.

[Mos18] Michele Mosca. "Cybersecurity in an Era with Quantum Computers: Will We Be Ready?" In: *IEEE Secur. Priv.* 5 (2018). DOI: 10.1109/MSP.2018.3761723.

[MOG+20] Kit Murdock, David F. Oswald, Flavio D. Garcia, Jo Van Bulck, Daniel Gruss, and Frank Piessens. "Plundervolt: Software-based Fault Injection Attacks against Intel SGX". In: *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*. 2020. DOI: 10.1109/SP40000.2020.00057.

[MIS20] Koksal Mus, Saad Islam, and Berk Sunar. "QuantumHammer: A Practical Hybrid Attack on the LUOV Signature Scheme". In: *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*. 2020. DOI: 10.1145/3372297.3417272.

[NIST15] National Institute of Standards and Technology. *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. https://doi.org/10.6028/NIST.FIPS.202. 2015. (Visited on 02/12/2025).

[NIST17] National Institute of Standards and Technology. *Call for Proposals for the Post-Quantum Cryptography Standardization Process*. https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf. 2017.

[NIST22] National Institute of Standards and Technology. *Call for Additional Digital Signature Schemes for the Post-Quantum Cryptography Standardization Process*. https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/call-for-proposals-dig-sig-sept-2022.pdf. 2022.

[NIST24] National Institute of Standards and Technology. *Round 2 Additional Signatures*. https://csrc.nist.gov/projects/pqc-dig-sig/round-2-additional-signatures. 2024. (Visited on 01/23/2025).

[OSPG18] Tobias Oder, Tobias Schneider, Thomas Pöppelmann, and Tim Güneysu. "Practical CCA2-Secure and Masked Ring-LWE Implementation". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 1 (2018). DOI: 10.13154/TCHES.V2018.I1.142-174.

[OTV04] Katsuyuki Okeya, Tsuyoshi Takagi, and Camille Vuillaume. "On the Importance of Protecting Delta in SFLASH against Side Channel Attacks". In: *International Conference on Information Technology: Coding and Computing (ITCC'04), Volume 2, April 5-7, 2004, Las Vegas, Nevada, USA*. 2004. DOI: 10.1109/ITCC.2004.1286713.

[ØFR24] Morten Øygarden, Patrick Felke, and Håvard Raddum. "Analysis of Multivariate Encryption Schemes: Application to Dob and C^*". In: *J. Cryptol.* 3 (2024). DOI: 10.1007/S00145-024-09501-W.

[ØFRC20] Morten Øygarden, Patrick Felke, Håvard Raddum, and Carlos Cid. "Cryptanalysis of the Multivariate Encryption Scheme EFLASH". In: *Topics in Cryptology - CT-RSA 2020 - The Cryptographers' Track at the RSA Conference 2020, San Francisco, CA, USA, February 24-28, 2020, Proceedings*. 2020. DOI: 10.1007/978-3-030-40186-3_5.

[PSKH18] Aesun Park, Kyung-Ah Shim, Namhun Koo, and Dong-Guk Han. "Side-Channel Attacks on Post-Quantum Signature Schemes based on Multivariate Quadratic Equations - Rainbow and UOV -". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 3 (2018). DOI: 10.13154/TCHES.V2018.I3.500-523.

[Pat95] Jacques Patarin. "Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt'88". In: *Advances in Cryptology - CRYPTO*. 1995. DOI: 10.1007/3-540-44750-4_20.

[Pat96] Jacques Patarin. "Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms". In: *Advances in Cryptology - EUROCRYPT*. 1996. DOI: 10.1007/3-540-68339-9_4.

[Pat97] Jacques Patarin. "The oil and vinegar algorithm for signatures". In: *Dagstuhl Workshop on Cryptography, 1997*. 1997.

[PCF+23] Jacques Patarin, Benoît Cogliati, Jean-Charles Faugére, Pierre-Alain Fouque, Louis Goubin, Robin Larrieu, Gilles Macario-Rat, and Brice Minaud. *VOX*. Tech. rep. National Institute of Standards and Technology, 2023.

[Péb24] Pierre Pébereau. "One Vector to Rule Them All: Key Recovery from One Vector in UOV Schemes". In: *Post-Quantum Cryptography - 15th International Workshop, PQCrypto 2024, Oxford, UK, June 12-14, 2024, Proceedings, Part II*. 2024. DOI: 10.1007/978-3-031-62746-0_5.

[PPS17] Ray A. Perlner, Albrecht Petzoldt, and Daniel Smith-Tone. "Total Break of the SRP Encryption Scheme". In: *Selected Areas in Cryptography - SAC 2017 - 24th International Conference, Ottawa, ON, Canada, August 16-18, 2017, Revised Selected Papers*. 2017. DOI: 10.1007/978-3-319-72565-9_18.

[PP21] Peter Pessl and Lukas Prokop. "Fault Attacks on CCA-secure Lattice KEMs". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2 (2021). DOI: 10.46586/TCHES.V2021.I2.37-60.

[PTBW11] Albrecht Petzoldt, Enrico Thomae, Stanislav Bulygin, and Christopher Wolf. "Small Public Keys and Fast Verification for Multivariate Quadratic Public Key Systems". In: *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*. 2011. DOI: 10.1007/978-3-642-23951-9_31.

[PPRS23] Rafael del Pino, Thomas Prest, Mélissa Rossi, and Markku-Juhani O. Saarinen. "High-Order Masking of Lattice Signatures in Quasi-linear Time". In: *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023*. 2023. DOI: 10.1109/SP46215.2023.10179342.

[PSN21] David Pokorný, Petr Socha, and Martin Novotný. "Side-channel attack on Rainbow post-quantum signature". In: *Design, Automation & Test in Europe Conference & Exhibition, DATE 2021, Grenoble, France, February 1-5, 2021*. 2021. DOI: 10.23919/DATE51398.2021.9474157.

[PS05] Thomas Pornin and Julien P. Stern. "Digital Signatures Do Not Guarantee Exclusive Ownership". In: *Applied Cryptography and Network Security, Third International Conference, ACNS 2005, New York, NY, USA, June 7-10, 2005, Proceedings*. 2005. DOI: 10.1007/11496137_10.

[QRC21] Quantum resistant cryptography research center. *Korean post-quantum cryptographic competition*. https://kpqc.or.kr/. 2021.

[RMR+21] Hany Ragab, Alyssa Milburn, Kaveh Razavi, Herbert Bos, and Cristiano Giuffrida. "CrossTalk: Speculative Data Leaks Across Cores Are Real". In: *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*. 2021. DOI: 10.1109/SP40001.2021.00020.

[RT25] Lars Ran and Monika Trimoska. "Shifting Our Knowledge of MQ-Sign Security". In: *Post-Quantum Cryptography - 16th International Workshop, PQCrypto 2025, Taipei, Taiwan, April 8-10, 2025, Proceedings, Part I*. 2025. DOI: 10.1007/978-3-031-86599-2_8.

[RCDB24] Prasanna Ravi, Anupam Chattopadhyay, Jan-Pieter D'Anvers, and Anubhab Baksi. "Side-channel and Fault-injection attacks over Lattice-based Post-quantum Schemes (Kyber, Dilithium): Survey and New Results". In: *ACM Trans. Embed. Comput. Syst.* 2 (2024). DOI: 10.1145/3603170.

[RRCB20] Prasanna Ravi, Sujoy Sinha Roy, Anupam Chattopadhyay, and Shivam Bhasin. "Generic Side-channel attacks on CCA-secure lattice-based PKE and KEMs". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 3 (2020). DOI: 10.13154/TCHES.V2020.I3.307-335.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". In: *Commun. ACM* 2 (1978). DOI: 10.1145/359340.359342.

[SSH11] Koichi Sakumoto, Taizo Shirai, and Harunaga Hiwatari. "Public-Key Identification Schemes Based on Multivariate Quadratic Polynomials". In: *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*. 2011. DOI: 10.1007/978-3-642-22792-9_40.

[SMA+24] Oussama Sayari, Soundes Marzougui, Thomas Aulbach, Juliane Krämer, and Jean-Pierre Seifert. "HaMAYO: A Fault-Tolerant Reconfigurable Hardware Implementation of the MAYO Signature Scheme". In: *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer. 2024.

[SPOG19] Tobias Schneider, Clara Paglialonga, Tobias Oder, and Tim Güneysu. "Efficiently Masking Binomial Sampling at Arbitrary Orders for Lattice-Based Crypto". In: *Public-Key Cryptography - PKC 2019 - 22nd IACR International Conference on Practice and Theory of Public-Key Cryptography, Beijing, China, April 14-17, 2019, Proceedings, Part II*. 2019. DOI: 10.1007/978-3-030-17259-6_18.

[Sch89] Claus-Peter Schnorr. "Efficient Identification and Signatures for Smart Cards". In: *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*. 1989. DOI: 10.1007/0-387-34805-0_22.

[Sch91] Claus-Peter Schnorr. "Efficient Signature Generation by Smart Cards". In: *J. Cryptol.* 3 (1991). DOI: 10.1007/BF00196725.

[SKA22] Kyung-Ah Shim, Jeongsu Kim, and Youngjoo An. *MQ-Sign: A New Post-Quantum Signature Scheme based on Multivariate Quadratic Equations: Shorter and Faster*. https://www.kpqc.or.kr/images/pdf/MQ-Sign.pdf. 2022.

[SK20] Kyung-Ah Shim and Namhun Koo. "Algebraic Fault Analysis of UOV and Rainbow With the Leakage of Random Vinegar Values". In: *IEEE Trans. Inf. Forensics Secur.* (2020). DOI: 10.1109/TIFS.2020.2969555.

[Sho94] Peter W. Shor. "Algorithms for Quantum Computation: Discrete Logarithms and Factoring". In: *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*. 1994. DOI: 10.1109/SFCS.1994.365700.

[SA02] Sergei P. Skorobogatov and Ross J. Anderson. "Optical Fault Induction Attacks". In: *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*. 2002. DOI: 10.1007/3-540-36400-5_2.

[SLKG23] Hauke Malte Steffen, Georg Land, Lucie Johanna Kogelheide, and Tim Güneysu. "Breaking and Protecting the Crystal: Side-Channel Analysis of Dilithium in Hardware". In: *Post-Quantum Cryptography - 14th International Workshop, PQCrypto 2023, College Park, MD, USA, August 16-18, 2023, Proceedings*. 2023. DOI: 10.1007/978-3-031-40003-2_25.

[SGB01] Rainer Steinwandt, Willi Geiselmann, and Thomas Beth. "A Theoretical DPA-Based Cryptanalysis of the NESSIE Candidates FLASH and SFLASH". In: *Information Security, 4th International Conference, ISC 2001, Malaga, Spain, October 1-3, 2001, Proceedings*. 2001. DOI: 10.1007/3-540-45439-X_19.

[SPMS02] Jacques Stern, David Pointcheval, John Malone-Lee, and Nigel P. Smart. "Flaws in Applying Proof Methodologies to Signature Schemes". In: *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*. 2002. DOI: 10.1007/3-540-45708-9_7.

[TPD21] Chengdong Tao, Albrecht Petzoldt, and Jintai Ding. "Efficient Key Recovery for All HFE Signature Variants". In: *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part I*. 2021. DOI: 10.1007/978-3-030-84242-0_4.

[Tec23] NewAE Technology. *Repository of ChipWhisperer tool chain - commit a9527b5*. https://github.com/newaetech/chipwhisperer. 2023.

[TII25] Technology Innovation Institute. *PQ-SORT: Post-Quantum Signatures On-Ramp Tests*. https://pqsort.tii.ae/. 2025. (Visited on 05/24/2025).

[TW12] Enrico Thomae and Christopher Wolf. "Cryptanalysis of enhanced TTS, STS and all its variants, or: Why cross-terms are important". In: *Progress in Cryptology-AFRICACRYPT 2012: 5th International Conference on Cryptology in Africa, Ifrance, Morocco, July 10-12, 2012. Proceedings 5*. Springer. 2012.

[VKS11] Ingrid Verbauwhede, Dusko Karaklajic, and Jörn-Marc Schmidt. "The Fault Attack Jungle - A Classification Model to Guide You". In: *2011 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2011, Tokyo, Japan, September 29, 2011*. 2011. DOI: 10.1109/FDTC.2011.13.

[WCD+23] Lih-Chung Wang, Chun-Yen Chou, Jintai Ding, Yen-Liang Kuan, Ming-Siou Li, Bo-Shu Tseng, Po-En Tseng, and Chia-Chun Wang. *SNOVA*. Tech. rep. National Institute of Standards and Technology, 2023.

[WPH+22] Yingchen Wang, Riccardo Paccagnella, Elizabeth Tang He, Hovav Shacham, Christopher W. Fletcher, and David Kohlbrenner. "Hertzbleed: Turning Power Side-Channel Attacks Into Remote Timing Attacks on x86". In: *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*. 2022.

[Wes24] Bas Westerbaan. *The state of the post-quantum Internet*. https://blog.cloudflare.com/pq-2024/. 2024. (Visited on 01/27/2025).

[YC05] Bo-Yin Yang and Jiun-Ming Chen. "Building Secure Tame-like Multivariate Public-Key Cryptosystems: The New TTS". In: *Information Security and Privacy, 10th Australasian Conference, ACISP 2005, Brisbane, Australia, July 4-6, 2005, Proceedings*. 2005. DOI: 10.1007/11506157_43.

[YL17] Haibo Yi and Weijian Li. "On the Importance of Checking Multivariate Public Key Cryptography for Side-Channel Attacks: The Case of enTTS Scheme". In: *Comput. J.* 8 (2017). DOI: 10.1093/COMJNL/BXX010.

[YN18] Haibo Yi and Zhe Nie. "Side-channel security analysis of UOV signature for cloud-based Internet of Things". In: *Future Gener. Comput. Syst.* (2018). DOI: 10.1016/J.FUTURE.2018.04.083.

A

Developing Physical Attacks

In this chapter, we include our contribution to the development of physical attacks against UOV-based signature schemes.

Contents

The paper in Appendix A.1 is the publication *Recovering Rainbow's Secret Key with a First-Order Fault Attack* [**A**KK+22] in AFRICACRYPT'22.

Appendix A.2 contains the paper *Separating Oil and Vinegar with a Single Trace* [**A**CK+23], published at TCHES'23.

In Appendix A.3 we include the FDTC'24 paper *MAYo or MAY-not: Exploring Implementation Security of the Post-Quantum Signature Scheme MAYO Against Physical Attacks* [**A**MS+24].

Recovering Rainbow's Secret Key with a First-Order Fault Attack

Thomas Aulbach[1(✉)], Tobias Kovats[2], Juliane Krämer[1], and Soundes Marzougui[3]

[1] Universität Regensburg, Regensburg, Germany
{thomas.aulbach,juliane.kraemer}@ur.de
[2] SBA Research, Vienna, Austria
tkovats@sba-research.org
[3] Technische Universität Berlin, Berlin, Germany
soundes.marzougui@tu-berlin.de

Abstract. Rainbow, a multivariate digital signature scheme and third round finalist in NIST's PQC standardization process, is a layered version of the unbalanced oil and vinegar (UOV) scheme. We introduce two fault attacks, each focusing on one of the secret linear transformations T and S used to hide the structure of the central map in Rainbow. The first fault attack reveals a part of T and we prove that this is enough to achieve a full key recovery with negligible computational effort for all parameter sets of Rainbow. The second one unveils S, which can be extended to a full key recovery by the Kipnis-Shamir attack. Our work exposes the secret transformations used in multivariate signature schemes as an important attack vector for physical attacks, which need further protection. Our attacks target the optimized Cortex-M4 implementation and require only first-order instruction skips and a moderate amount of faulted signatures.

Keywords: Rainbow · Fault injection attacks · Multivariate schemes · Post-quantum cryptography · Cortex M4 implementation

1 Introduction

Quantum computers pose a threat to public-key schemes based on the integer factorization or the discrete logarithm problem, like the widely deployed RSA and ECC. Since Shor published his famous algorithms [28] to solve these problems in polynomial time, their security depends on the technical feasibility of large scale quantum computers. Although there still is a certain amount of scepticism about the possibility of quantum computers being capable of factorizing integers and solving the discrete logarithm for cryptographically relevant instances in the upcoming decades [17,19,26], the National Institute of Standards and Technology (NIST) states that:

> "[...] regardless of whether we can estimate the exact time of the arrival of the quantum computing era, we must begin now to prepare our information security systems to be able to resist quantum computing" [1].

L. Batina and J. Daemen (Eds.): AFRICACRYPT 2022, LNCS 13503, pp. 348–368, 2022.
https://doi.org/10.1007/978-3-031-17433-9_15

Therefore, a standardization process for post-quantum, i.e., quantum-resistant cryptographic algorithms was initiated. Currently five families of post-quantum cryptography are being studied, each relying on different mathematical assumptions. Concerning signatures, the remaining candidates in the currently ongoing third round of the standardization process mainly consist of lattice-based, hash-based, and multivariate schemes.

The evaluation criteria of NIST are not only the security and performance of the candidates, but also other properties such as resistance to side channel attacks and misuse resistance. Hence, NIST asked for efficient implementations that are protected against physical attacks, such as side channel and fault attacks. In these attacks, an attacker does not exploit mathematical weaknesses of a cryptographic scheme. Instead, in a side channel attack an attacker measures physical information during the computation of a cryptographic algorithm that she then analyzes to reveal secret data. In a fault attack, an attacker intentionally introduces faults into the computation such that it results in faulty outputs that she can analyze to learn secret data. In a first-order fault attack, an attacker induces a single fault during a computation, while in higher-order fault attacks, at least two faults are induced in the same computation. Since this is technically more complex, first-order faults are generally believed to be more realistic and, hence, more practically relevant.

In this work, we study first-order fault attacks on Rainbow, a multivariate signature scheme that was selected as finalist in the NIST standardization [14,15]. We are aware of the significant improvements in mathematical cryptanalysis on the multivariate signature schemes GeMSS and Rainbow that have been published recently [3,4,30]. Especially the improved cryptanalytic approach presented by Beullens massively reduces the complexity of key recovery attacks against Rainbow, in particular against the parameter set for security level I [4]. However, NIST announced that there will be a fourth round where further post-quantum signature schemes can be submitted[1]. This became necessary since on the one hand, for security and practicality reasons diversity of (post-quantum) signatures is needed, while on the other hand the remaining signature candidates in the current third round are mostly based on structured lattices. Thus, for the fourth round NIST is especially interested in schemes that are not based on structured lattices and we expect that, despite the recent cryptanalytic results, other multivariate schemes will be submitted. For instance, the well-studied scheme UOV is likely to gain more attention soon, since it is explicitly not affected by Beullens latest approach. Hence, we believe that our results are relevant for the future development of multivariate signature schemes even if Rainbow turns out to be insecure or even broken. Our results reveal attack vectors for and weaknesses of a specific multivariate signature scheme, which have to be prevented in future developments or optimizations of other multivariate signature schemes, too.

[1] Announced by Dustin Moody at the third PQC Standardization Conference in June 2021 https://www.nist.gov/video/third-pqc-standardization-conference-session-i-welcomecandidate-updates.

Related Work. The number of publications on the physical security of multivariate cryptography has increased in recent years, but is still manageable. Some effort was put into side channel analysis (SCA) of signature schemes. To name a few of them, Steinwandt et al. theoretically conducted differential power analysis (DPA) to reveal the secret seed and subsequently the affine bijections S and T used in FLASH and SFLASH already in 2001 [29]. Some years later, Okeya et al. were the first to experimentally verify a DPA attack against SFLASH [23]. More recently, in 2017, Yi and Li presented a DPA on enTTS [32], a signature scheme that contains some common features with Rainbow, such as the layer structure of the central map and the enclosing affine transformations. Finally, there are side channel attacks by Park et al. [24] and by Pokorny et al. [25] on Rainbow, both targeting the affine transformations via correlation power analysis (CPA).

The literature on fault attacks on multivariate signatures is less extensive. In 2011, Hashimoto et al. described some general ideas that might be applicable to multivariate schemes [18]. However, their ideas remain rather high-level and refer to several schemes at once. The authors in [32] mentioned a fault model that is supposed to facilitate the DPA on the central map F of the enTTS scheme, but they also did not provide a detailed description of their approach. They merely stated to "cause a fault, to change the unknown items during the signature generation". Krämer and Loiero transferred two ideas of [18] to UOV and Rainbow [21]. First, they analyzed how a faulted coefficient in the central map propagates through the signature and can be utilized to regain information about the secret transformation S. Second, they discussed the effect of fixing the vinegar variables across multiple signatures and show how information about the secret transformation T can be revealed by this. Shim and Koo developed the latter approach further to a full key recovery attack [27]. But the algebraic post-processing method they used still has a significant complexity of 2^{40}, rendering the attack impractical. The most practical attack yet, called *Quantumhammer* [22] by Mus, Islam, and Sunar, was performed on LUOV. The authors randomly induce bit flips in the linear transformation T and learn one bit of the secret key through each faulty signature. They also append an algebraic attack to the online fault injection phase, but they manage to limit the effort in the range of hours. In summary, there is no fault attack on Rainbow that presents a full key recovery in reasonable time.

Contribution. We introduce the first two efficiently executable fault attacks on Rainbow that lead to full key recovery. Both fault attacks only require first-order instruction skips and a moderate number of faulted signatures to be executed. We target the optimized Cortex-M4 implementation from [12].

1. We revisit the already existing theoretical approach of fixing the vinegar variables via a fault injection attack. This attack leads to partial leakage of one of the secret transformations used in Rainbow. The authors of [27] suggested to exploit this leakage by speeding up the key recovery attack using the notion of good keys [31]. Although they can reduce the number of variables and equations, the remaining system of quadratic equations is still

of significant complexity of around 2^{40} [27, Table. VII] for the Rainbow level I parameter set. We introduce a cryptanalytical method for circumventing this costly procedure and show how it is possible to recover the remaining bits of the secret key by just solving linear equations. Contrary to the previously suggested key recovery attack, this method can be applied to any possible parameter set of Rainbow and leads to full key recovery.
2. We present a new fault attack that targets the application of the linear transformation S. By collecting a small number of faulted signatures, we obtain enough input and output values of S to completely recover it. By knowledge of S, the complexity of Rainbow can be reduced to a small UOV instance with reduced parameter sizes [3]. To complete the key recovery, we apply the Kipnis-Shamir attack[2] for unbalanced oil and vinegar values [20], with remaining complexity $\mathcal{O}(q^{v_1-o_1})$. Considering Rainbows security level I parameter set, it holds $q^{v_1-o_1} = 2^{16}$. Compared to the first fault attack, this attack works with half the number of faulted signatures.

We see the algebraic post-processing that is used to further exploit the information gained by the fault attacks as a contribution of its own. It can be used as a plug-and-play method for all kinds of physical attacks. For instance, Sect. 3.1 proves that if an attacker obtains the block T_1 of the secret Rainbow parameters through any kind of leakage, she can achieve full key recovery without any further physical or computational effort.

Furthermore, we verify our attacks on an emulated ARM M4 architecture. On the one hand, this implies that we execute the compiled binary of the source code as a real signing device would and, therefore, can target the specific instruction of the assembly code that needs to be skipped. On the other hand, it verifies the feasibility of our attacks and proves the claims we made above for a given Rainbow key pair.

Finally, we suggest efficient countermeasures to prevent the mentioned attacks and make implementations of multivariate schemes more resilient against fault attacks.

1.1 Organization

In Sect. 2, we develop the background that is necessary for the presented attacks. This includes the Rainbow signature scheme, relevant simplifications applied by the authors of the corresponding NIST submission, and background information on fault attacks. In Sect. 3, we present the two fault attacks, together with a detailed description of the algebraic post-processing. We uncover the low-level instructions that need to be skipped in the practical fault attack in Sect. 4 and present our simulation. In Sect. 5, we suggest countermeasures to the previously described fault attacks and Sect. 6 concludes the work.

[2] In a very recent paper Esser et al. claim that there is another way to complete the key recovery, instead of using the Kipnis-Shamir attack [16]. If their findings hold true, this works with significantly lower complexity $\mathcal{O}(n^3)$, which is efficient even for higher parameter sets.

2 Background

In this section, we recall useful background information for understanding the rest of the paper. This includes an overview on the Rainbow signature scheme and fault injection attacks.

Notation. Let $\mathbb{F}_q$ be a finite field with q elements. By $x = (x_1, \ldots, x_n) \in \mathbb{F}_q^n$ we denote a vector and by $T \in M_n(\mathbb{F}_q)$ we denote a matrix with entries in $\mathbb{F}_q$. The multivariate quadratic maps $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ are given by m quadratic polynomials $p^{(i)}$ in n variables. To concatenate two strings x and y, $x||y$ is written. $\mathcal{H}(x)$ represents the application of a hash function $\mathcal{H}$ on a value x.

2.1 The Rainbow Signature Scheme

The Rainbow signature scheme [15] can be seen as a generalization of the unbalanced oil and vinegar (UOV) scheme [20]. It consists of several layers, where the oil and vinegar variables of the i-th layer are used as vinegar variables of the subsequent layer. Inserting them into the central polynomials of this layer, leads to easily solvable linear equations, since there are no quadratic oil terms in each layer just as it is the case for UOV. Initially, the authors suggested to use $u = 5$ layers, but it turned out to work best for $u = 2$ which is used by all currently suggested parameter sets. The case $u = 1$ constitutes the original UOV, the scheme whose security and efficiency Rainbow is supposed to improve.

More formally, let $\mathbb{F}_q$ be a finite field with q elements and $v_1 < \ldots < v_{u+1} = n$ be integers. Set $V_i = \{1, \ldots, v_i\}$ and $O_i = \{v_i + 1, \ldots, v_{i+1}\}$ for $i \in \{1, \ldots, u\}$. Therefore, it holds $|V_i| = v_i$ and $|O_i| = o_i$ for $i \in \{1, \ldots, u\}$. The central map F of Rainbow consists of $m = n - v_1$ multivariate quadratic polynomials $f^{(v_1+1)}, \ldots, f^{(n)}$ of the form

$$f^{(k)}(x) = \sum_{i,j \in V_l} \alpha_{ij}^{(k)} x_i x_j + \sum_{i \in V_l, j \in O_l} \beta_{ij}^{(k)} x_i x_j + \sum_{i \in V_l \cup O_l} \gamma_i^{(k)} x_i + \delta^{(k)}, \quad (1)$$

where $l \in \{1, \ldots, u\}$ is the only integer such that $k \in O_l$. In each layer l there remain no quadratic terms in the polynomials $f^{(k)}$ after inserting the values of the vinegar variables x_i for $i \in V_l$. This leads to an easily invertible central map $F : \mathbb{F}^n \to \mathbb{F}^m$ consisting of the m coordinate functions f. This special structure of $\mathcal{F}$ facilitates the signature generation and must be hidden in the public key. To this end, two invertible linear maps $S : \mathbb{F}^m \to \mathbb{F}^m$ and $T : \mathbb{F}^n \to \mathbb{F}^n$ are concatenated to the central map in order to generate the *public key*

$$\mathcal{P} = S \circ \mathcal{F} \circ T : \mathbb{F}^n \to \mathbb{F}^m. \quad (2)$$

Since the composed maps look like a random system of multivariate quadratic equations, it is hard to find a preimage under $\mathcal{P}$. By holding the *private key* maps $S, \mathcal{F}$, and T, this task becomes feasible. The signing and verifying procedure for the Rainbow signature scheme can be briefly summarized as follows. For the signing procedure we also present the pseudo code in Algorithm 1.

Signature Generation: To generate a signature for a message (or hash value) $w \in \mathbb{F}_m$, one performs the following three steps.

1. Compute $x = S^{-1}(w) \in \mathbb{F}^m$.
2. Compute a pre-image y of x under the central map $\mathcal{F}$.
3. Compute the signature $z \in \mathbb{F}^n$ by $z = T^{-1}(y)$.

Signature Verification: To check, if $z \in \mathbb{F}^n$ is a valid signature for a message $w \in \mathbb{F}^m$, one simply computes $w_0 = \mathcal{P}(z)$. If $w_0 = w$ holds, the signature is accepted, otherwise rejected.

Since both fault attacks presented in this work target the signing procedure, we shortly present it in Algorithm 1 to give the reader a first intuition of the code lines that render the signature scheme vulnerable.

The algorithm is located in Sect. 3.1 to keep it close to the presented fault attacks. We use a similar description as [14, Section 3.5], but align the notation tighter to the actual implementation and simplify the representation of the secret maps, as they are all chosen to be homogeneous in [14, Section 4], anyway. The first fault attack in Sect. 3.1 targets the sampling of random vinegar variables in Line 2. The second fault attack in Sect. 3.2 bypasses the application of S in Line 8. Since both lines are located in a loop, one has to consider under which conditions the fault might be annihilated by a repeated execution of the respective lines. We exit the while loop if the matrices given by $\hat{F}_1$ and $\hat{F}_2$ are invertible. If we assume the entries to be uniformly distributed in $\mathbb{F}_q$, the probability that a matrix $M_n(\mathbb{F}_q)$ is invertible is given by

$$\prod_{i=0}^{n}\left(\frac{q^n - q^i}{q^{n^2}}\right) = \prod_{i=1}^{n}\left(1 - \frac{1}{q^i}\right). \tag{3}$$

For the given parameters this evaluates to approximately 93%. We will carefully analyze what impact the injected fault might have on the conditions of the while loop in Sect. 4.2.

Remark 1. Very recently, Beullens presented an improved cryptanalytic approach that massively reduces the complexity of key recovery attacks [4], in particular against the Rainbow parameter set for security level I. His paper builds on an earlier analysis of him [3], where he introduced a new description of the Rainbow scheme, avoiding the presence of the central map and rather considering secret subspaces that satisfy certain equations under the public map $\mathcal{P}$ and its polar form $\mathcal{P}'$. He defines

$$\begin{aligned} O'_1 \subset \mathbb{F}_q^n &:= \{x \in \mathbb{F}_q^n : x_i = 0 \text{ for } i \in \{1, \ldots, v_1\}\}, \\ O'_2 \subset \mathbb{F}_q^n &:= \{x \in \mathbb{F}_q^n : x_i = 0 \text{ for } i \in \{1, \ldots, v_1 + o_1\}\}, \\ W' \subset \mathbb{F}_q^m &:= \{x \in \mathbb{F}_q^m : x_i = 0 \text{ for } i \in \{1, \ldots, o_1\}\}. \end{aligned}$$

The interesting point about these (public) subspaces is that all polynomials of the central map vanish on O'_2 and the polynomials of the first layer vanish

even on O_1', i.e., it holds $\mathcal{F}(O_2') = 0$ and $\mathcal{F}(O_1') \subset W'$, respectively. The secret linear maps S and T now transform the given subspaces to the secret subspaces $O_1 = T^{-1}O_1', O_2 = T^{-1}O_2'$, and $W = SW'$. The new technique in [4] finds a vector in the secret subspace O_2 with way less computational effort than needed in previous works. The attack is completed in similar style as in [3], where he uses the vector in O_2 to recover W efficiently by using the polar form of the public key and finally, applying the Kipnis-Shamir attack to compute O_1. The important take-away for our analysis in Sect. 3.2 is that recovering the secret transformation S is equivalent to detecting the secret subspace W using this notation. For more details we refer to [3, Section 5].

2.2 Conventions in the Specification

In the Rainbow specification [14], several simplifications are made. They are introduced to speed up the key generation process and reduce the key sizes, while it is argued that they do not weaken the security of Rainbow. First, the secret transformations S and T are chosen to be of the form

$$S = \begin{pmatrix} I & S_1 \\ 0 & I \end{pmatrix} \quad \text{and} \quad T = \begin{pmatrix} I & T_1 & T_2 \\ 0 & I & T_3 \\ 0 & 0 & I \end{pmatrix}. \tag{4}$$

This is justified by the fact that, for every public map $\mathcal{P}$, there exists an equivalent secret key $(S, \mathcal{F}, T)$ with S and T as in Eq. (4). Consequently, the inverse maps have the same structure and are given by

$$S^{-1} = \begin{pmatrix} I & S_1 \\ 0 & I \end{pmatrix} \quad \text{and} \quad T^{-1} = \begin{pmatrix} I & T_1 & T_4 \\ 0 & I & T_3 \\ 0 & 0 & I \end{pmatrix}, \tag{5}$$

where $T_4 = T_1T_3 - T_2$. Furthermore, the Rainbow submitters restrict themselves to a homogeneous central map $\mathcal{F}$. As a result, the public map $\mathcal{P} = S \circ \mathcal{F} \circ T$ is homogeneous as well. Thus, the coefficients of every quadratic polynomial $f^{(i)}$ and $p^{(j)}$ for $i, j \in \{1, \ldots, m\}$ can be collected in $n \times n$ matrices, by defining $F_i \in M_{n\times n}(\mathbb{F}_q)$ and $P_j \in M_{n\times n}(\mathbb{F}_{\text{ıı}})$ as the matrices that satisfy $f^{(i)}(x) = x^\top F_i x$ and $p^{(j)}(x) = x^\top P_j x$, respectively. Following this notation, Eq. (2) can be turned into an equation of matrices of the form

$$F_i = \sum_{j=1}^{m} \tilde{s}_{ij}(\tilde{T}^\top P_j \tilde{T}). \tag{6}$$

Here, $\tilde{s}_{ij}$ denote the entries of S^{-1} and $\tilde{T} = T^{-1}$. This method of switching back and forth from public to secret matrices will play a major role in the analysis of our fault attacks. Interchanging the roles of F_i and P_j in Eq. (6) represents the basic procedure of computing the public key from the private key. In the above form, the equation occurs less frequently in the literature, but is used, e.g., by

Thomae in [31]. Due to the structure of the central polynomials in Eq. (1), several parts of the matrices F_i are forced to be zero and this is obvious to any attacker. In more detail, the zero blocks of the matrices are given as

$$F_i = \begin{pmatrix} F_i^{(1)} & F_i^{(2)} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \text{ for } i \in \{1, \ldots, o_1\} \tag{7}$$

and

$$F_i = \begin{pmatrix} F_i^{(1)} & F_i^{(2)} & F_i^{(3)} \\ 0 & F_i^{(5)} & F_i^{(6)} \\ 0 & 0 & 0 \end{pmatrix} \text{ for } i \in \{o_1 + 1, \ldots, o_2\}.$$

If an attacker somehow obtains secret information, she can use Eq. (6) together with the structure of the occurring matrices given by Eq. (5) and Eq. (7), to further exploit this leakage. In Sect. 3.1 we will show, e.g., how an attacker that has T_1 at hand, is able to recover S_1 and subsequently T_3 and T_4, just by solving linear equations. Although this assumes a strong leakage in the first place, to the best of our knowledge this is the first work that demonstrates such a result.

2.3 Fault Attacks

Fault attacks against cryptographic schemes were first described 25 years ago [8]. Since then, a great variety of fault attacks has been developed. All fault attacks have in common that an attacker actively and intentionally disturbs the computation of a cryptographic algorithm so as to gain secret information from the faulty output. Both the kind of physical fault and the effect of a fault can be manifold. For instance, a fault can be injected via clock glitching, e.g., [9], or laser fault injection, e.g., [11], and it can be either transient or permanent. Most published fault attacks result in a zeroed or randomized value or an instruction skip, see, e.g., [6]. Instruction skips as also used in this work correspond to skipping, i.e., ignoring, selected lines of the program code. Since fault attacks were first described, most cryptographic schemes have been analyzed with respect to them and in recent years, fault attacks have been published for all five families of post-quantum cryptography, e.g., [6,9–11,21]. These works range from purely theoretical publications [6] to more practical attacks [22].

3 Full Key Recovery Attacks

In this section we present two different fault attack scenarios. Both rely on a first-order skipping fault model, i.e., the attacker is assumed to be capable of introducing a single instruction skip during the signing procedure. Compared to fault attacks that require randomizing or zeroing values in memory, the instruction skips belongs to the more practical ones and it has been shown that even

higher-order skipping faults against real-world cryptographic implementations are possible [7]. The first attack already exists in the literature. It aims at fixing the vinegar variables across consecutive signature generations and leads to valid signatures. We significantly reduce the complexity of the post-processing that is necessary to achieve a full key recovery. Additionally, we introduce a completely new attack that benefits from the same efficient techniques and works with even less faulted signatures. They constitute the first fault attacks on the Rainbow scheme that lead to a complete revealing of the secret key which are executable on a desktop machine. In both cases, the messages that are chosen to generate the faulty signatures not need to fulfill special requirements, that are hard to control. The messages must be different from each other and should lead to linearly independent vectors at some point of the respective procedures, but the attacker can just discard a faulty signature if it does not meet this requirement and start over with a new message. We will emphasize this condition in the description of the two fault attacks.

In Sect. 3.1, we reinvestigate the case of fixed vinegar variables, which was presented in detail in [21]. The authors of [27] are the first to expand this to a complete key recovery attack. However, their approach is reliant on a costly post-processing step that involves solving a system of quadratic equations of moderate size. They investigate three different level I parameter sets of Rainbow, and even in the best case their complexities remain as high as 2^{38}. This is still considerably large to constitute another hurdle in the practicability of the fault attack. We introduce a method on how to bypass this step, leading to an easily executable full key recovery attack.

Furthermore, we present in Sect. 3.2 a new fault attack that gets along with significantly less faulted signatures. The faulted output reveals the secret transformation S and the attack can be completed to a full key recovery by a subsequent Kipnis-Shamir attack. Due to the knowledge of S, the parameters of the system to solve allow for an efficiently executable instance of the Kipnis-Shamir attack. After explaining the attacks, we also translate the procedure to the abstract secret subspace notation used by Beullens which we stated in Remark 1.

3.1 Attack 1: Full Key Recovery from Fixed Vinegar Variables

This fault attack aims to skip the random generation of vinegar variables during the signing process. Depending on the implementation, this results in either the vinegar variables from the previous signature being reused, or being equal to zero in case they are zeroed at the end of the signing process. Both cases have already been mentioned in [27] and are very similar. The main difference is that for the reuse model an additional, unfaulted reference signature is needed. In the following we focus on the optimized bitsliced implementation developed in [12]. Here, the vinegar variables are not zeroed at the end of the signing process and therefore skipping the generation of the new vinegar variables results in the same variables being used for successive signature computations.

Algorithm 1. Rainbow Sign

Input message d, private key $(S, \mathcal{F}, T)$, length l of the salt.
Output signature $\sigma = (z, salt) \in \mathbb{F}_q^n \times \{0,1\}^l$ s.t. $\mathcal{P}(z) = \mathcal{H}(\mathcal{H}(d)||salt)$.

repeat
  $(z_1, \ldots, z_{v_1}) \leftarrow_R \mathbb{F}_q^{v_1}$
  $\left(\hat{f}^{(v_1+1)}, \ldots, \hat{f}^{(n)}\right) \leftarrow \left(f^{(v_1+1)}(z_1, \ldots, z_{v_1}), \ldots, f^{(n)}(z_1, \ldots, z_{v_1})\right)$
until IsInvertible$(\hat{F}_1)$ == True
repeat
  $salt \leftarrow_R \{0,1\}^l$
  $y \leftarrow \mathcal{H}(\mathcal{H}(d)||salt)$
  $y \leftarrow S^{-1}(y)$
  $z_{v_1+1}, \ldots, z_{v_2} \leftarrow \hat{F}_1^{-1}(y_{v_1+1}, \ldots, y_{v_2})$
  $\left(\hat{f}^{(v_2+1)}, \ldots, \hat{f}^{(n)}\right) \leftarrow \left(f^{(v_2+1)}(z_{v_1+1}, \ldots, z_{v_2}), \ldots, f^{(n)}(z_{v_1+1}, \ldots, z_{v_2})\right)$
until IsInvertible$(\hat{F}_2)$ == True
$z_{v_2+1}, \ldots, z_n \leftarrow \hat{F}_2^{-1}(y_{v_2+1}, \ldots, y_n)$
$z \leftarrow T^{-1}(z)$
$\sigma = (z, salt)$
return σ

First, we show how the secret matrices $T_1 \in M_{o_1 \times o_2}(\mathbb{F}_q)$ and $T_2 \in M_{o_2 \times o_2}(\mathbb{F}_q)$ in Sect. 2.2 can be determined from the faulty signatures. Let z' be the error-free generated signature of an arbitrary message d'. According to the Rainbow specification, z' is defined by $z' = T^{-1} \circ \mathcal{F}^{-1} \circ S^{-1}(y)$, where $y = \mathcal{H}(\mathcal{H}(d')||salt) \in \mathbb{F}_q^m$. From an attacker's point of view, all intermediate values are unknown. What is known is that the first v_1 entries of $\mathcal{F}^{-1} \circ S^{-1}(y)$ consist of the generated vinegar values, whereas the remaining $m = o_1 + o_2$ entries are the corresponding solutions of the first and second layer of the central map under the chosen vinegar variables. Thus, we can write

$$z' = T^{-1} \begin{pmatrix} v \\ o_1' \\ o_2' \end{pmatrix},$$

with $v \in \mathbb{F}_q^{v_1}, o_1' \in \mathbb{F}_q^{o_1}$, and $o_2' \in \mathbb{F}_q^{o_2}$. By using the instruction skip indicated in Sect. 2.1 and elaborated in detail in Sect. 4.1, the attacker successively generates m signatures, all of which fall back to the same vinegar variables v as the reference signature z'. For $i \in \{1 \ldots m\}$, we denote these signatures by

$$z^{(i)} = T^{-1} \begin{pmatrix} v \\ o_1^{(i)} \\ o_2^{(i)} \end{pmatrix}.$$

The remaining entries $o_1^{(i)}$ and $o_2^{(i)}$ of the input of T^{-1} are under no control of the attacker and do not need to be considered in more detail. By subtracting the reference signature and multiplying with T, we receive

$$T(z^{(i)} - z') = \begin{pmatrix} v \\ o_1^{(i)} \\ o_2^{(i)} \end{pmatrix} - \begin{pmatrix} v \\ o_1' \\ o_2' \end{pmatrix} = \begin{pmatrix} 0 \\ \tilde{o}_1^{(i)} \\ \tilde{o}_2^{(i)} \end{pmatrix}, \tag{8}$$

for $i \in \{1 \dots m\}$. Let $Z \in M_{n \times m}$ be the matrix whose i-th column is defined by the vector $z^{(i)} - z'$. Then Equation (8) implies that the first v_1 rows of T map Z to $0_{v_1 \times m}$. If this linear system of equations can be solved uniquely, it reveals the first v_1 rows of T, more precisely the submatrices $T_1 \in M_{o_1 \times o_2}(\mathbb{F}_q)$ and $T_2 \in M_{o_2 \times o_2}(\mathbb{F}_q)$. Therefore, we need the columns of Z and thus the last m entries of $Tz^{(i)}$ to be linearly independent, since the first entries are identical to the entries of the reference signature. Following Eq. 3 this happens with high probability and in case we draw a faulted signature that is linearly dependent of the previous, we can just disregard it and draw a new one. We note that Eq. (8) does not provide any further information about the remaining rows of T.

Remark 2. The authors of [27] only utilize parts of the gained information for their algebraic key recovery attack. More precisely, they use certain entries of the submatrices to reduce the complexity of a key recovery attack introduced in [31] using the good key approach. However, this still requires solving a system of quadratic equations. In the following we show how this can be completely omitted by utilizing the whole submatrix T_1.

Recover the Secret Transformation S. We take a closer look at Eq. (6). By also dividing the public matrices P_j and the secret transformation T into 3×3 block matrices we receive

$$\tilde{F}_i = \sum_{j=1}^{m} \tilde{s}_{ij} \left[\begin{pmatrix} I & 0 & 0 \\ T_1^\top & I & 0 \\ T_4^\top & T_3^\top & I \end{pmatrix} \begin{pmatrix} P_j^{(1)} & P_j^{(2)} & P_j^{(3)} \\ 0 & P_j^{(5)} & P_j^{(6)} \\ 0 & 0 & P_j^{(9)} \end{pmatrix} \begin{pmatrix} I & T_1 & T_4 \\ 0 & I & T_3 \\ 0 & 0 & I \end{pmatrix} \right]. \tag{9}$$

The resulting matrices are not necessarily equal to the matrices F_i of the central map but the polynomials they represent are identical. Consequently, by denoting

$$\tilde{F}_i = \begin{pmatrix} \tilde{F}_i^{(1)} & \tilde{F}_i^{(2)} & \tilde{F}_i^{(3)} \\ \tilde{F}_i^{(5)} & \tilde{F}_i^{(5)} & \tilde{F}_i^{(6)} \\ \tilde{F}_i^{(7)} & \tilde{F}_i^{(8)} & \tilde{F}_i^{(9)} \end{pmatrix}, \tag{10}$$

it follows from Eq. (7) that $\tilde{F}_i^{(5)}$ needs to be skew symmetric and $\tilde{F}_i^{(7)\top} + \tilde{F}_i^{(3)} = 0_{v_1 \times o_2}$ and $\tilde{F}_i^{(8)\top} + \tilde{F}_i^{(6)} = 0_{0_1 \times o_2}$ holds for the central maps of the first layer, i.e., for $i \in \{1, \dots, o_1\}$.

Now, we solely focus on the middle block $\tilde{F}_i^{(5)} \in M_{o_1 \times o_1}(\mathbb{F}_q)$ and observe that T_1 is the only part of the secret transformation T contributing to that block. Thus, neglecting the other submatrices turns Eq. (9) into

$$\tilde{F}_i^{(5)} = \sum_{j=1}^{m} \tilde{s}_{ij} \left(T_1^\top P_j^{(1)} T_1 + T_1^\top P_j^{(2)} + P_j^{(4)} T_1 + P_j^{(5)} \right). \tag{11}$$

Note that the term inside the brackets is completely known to the attacker, since she has already recovered T_1. The remaining unknowns are now the entries of S^{-1}, in particular the $o_1 \cdot o_2$ entries of S_1. Since Eq. (11) holds for all $i \in \{1, \ldots, o_1\}$, the resulting linear system of equations is overdetermined and solving it provides exactly the entries of S_1.

Recover the Remaining Part of the Secret Transformation T. Having access to the complete transformation S, the attacker is able to exploit (9) even more. She now targets the $v_1 \times o_2$ block $F_i^{(3)}$ on the top right and the $o_2 \times o_1$ block $F_i^{(7)}$ on the bottom left. Similarly to (11), she derives

$$F_i^{(7)\top} + F_i^{(3)} = 0_{v_1 \times o_2} = \sum_{j=1}^{m} \tilde{s}_{ij} \left(P_j^{(1)\top} T_4 + P_j^{(1)} T_4 + P_j^{(2)} T_3 + P_j^{(3)} \right). \quad (12)$$

Now the attacker wants to solve for the unknowns in T_3 and T_4. By now, she has knowledge of all the entries $\tilde{s}_{ij}$, which turns (12) into a linear system of equations. Once more the number of equations exceeds the number of variables and its solution reveals the submatrices T_3 and T_4 and therefore the remaining part of the secret transformation T^{-1}. This finishes the key recovery attack. The algebraic post-processing of the fault attack can be summarized as follows.

Attack 1: Full Key Recovery from Fixed Vinegar Variables. After successful execution of the fault attack, the attacker takes the reference signature z' and m faulted signatures $z^{(1)}, \ldots, z^{(m)}$, obtained in the way described above and proceeds as follows.

1. Build the matrix $Z \in M_{n \times m}(\mathbb{F}_q)$ with columns $z^{(i)} - z'$ for $i \in \{1, \ldots, m\}$.
2. Compute the echelon form of the matrix $T' \in M_{v_1 \times n}(\mathbb{F}_q)$ that fulfills $T'Z = 0$. It holds $T' = \begin{pmatrix} I & T_1 & T_2 \end{pmatrix}$.
3. Insert T_1 into Eq. (11). Solve the resulting system of linear equations to recover S.
4. Insert S into Eq. (12). Solve the resulting system of linear equations to recover T_3 and T_4.
5. Use Eq. (6) to obtain $\mathcal{F}$. The attacker recovered the full secret key $(S, \mathcal{F}, T)$.

Remark 3. This attack can also be translated to the more abstract language established in Remark 1. The difference of two signatures $z^{(i)} - z'$ that are generated with identical vinegar variables, can be seen as a vector in the secret subspace O_1. This becomes obvious when considering Eq. (8), which shows that T maps this vector to a vector whose first v_1 entries are zero, i.e., an element in O_1'. Thus, the m linearly independent vectors of the matrix Z that are gained by our fault attack, together span the secret subspace O_1 from which the remaining secret subspaces can be deduced.

3.2 Attack 2: Secret Key Recovery by Skipping the Linear Transformation S

This fault attack aims to skip the application of the matrix S^{-1} during the generation of the signature. If the instruction skip is successful, the signing process evaluates to $\tilde{z} = T^{-1} \circ \mathcal{F}^{-1}(y)$. By inserting this faulted signature into the public map $\mathcal{P}$, an attacker receives $\mathcal{P}(\tilde{z}) = S \circ \mathcal{F} \circ T(\tilde{z}) = S(y) =: w \in \mathbb{F}_q^m$.

Since $y = \mathcal{H}(\mathcal{H}(d)||salt)$ is known to the attacker, this fault attack presents a method for deriving input-output pairs for the secret linear transformation S. Now, let $W \in M_{m \times o_2}(\mathbb{F}_q)$ be the matrix whose columns consist of vectors $w^{(i)} \in \mathbb{F}_q^m, i \in \{1 \dots o_2\}$, which are obtained in the manner described above, and $Y \in M_{m \times o_2}(\mathbb{F}_q)$ be the matrix whose columns consist of the corresponding starting vectors $y^{(i)}, i \in \{1 \dots o_2\}$. By dividing the matrices and vectors into blocks according to Eq. (4), we receive

$$SY = \begin{pmatrix} I & S_1 \\ 0 & I \end{pmatrix} \begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix} = \begin{pmatrix} Y_1 + S_1 Y_2 \\ Y_2 \end{pmatrix} = \begin{pmatrix} W_1 \\ W_2 \end{pmatrix}. \tag{13}$$

Thus, the secret submatrix S_1 can be obtained via $S_1 = (W_1 - Y_1) * Y_2^{-1}$. Consequently, for the attack to be successful o_2 faulty signatures are needed and the starting vectors $y^{(i)}$ need to be chosen s.t. $Y_2 \in M_{o_2 \times o_2}(\mathbb{F}_q)$ is invertible.

Recover T by Using S. Having access to the secret transformation S, an attacker can use the very same strategy to recover T_4 and T_3 as described in Sect. 3.1. By the time this step was applied during the post-processing of the first fault attack, the attacker had already learned T_1, which is not the case anymore. However, in order to exploit Eq. (12) it is enough to know S, i.e., the attacker does not need any of the entries of T_1 to recover T_3 and T_4.

This procedure - although presented somewhat differently - was already proposed by Park et al. in [24, Section 4.2]. In their work, they used Correlation Power Analysis to obtain S and thus, faced the same challenge for the subsequent algebraic evaluation, i.e., the recovery of T under the knowledge of S. In order to obtain T_1, they suggest to use a similar approach, namely by focusing on the $o_1 \times o_2$ block $F_i^{(6)}$ and the $o_2 \times o_1$ block $F_i^{(8)}$ of (10). However, it is not possible to continue the attack like this, as we sketch in the following. Therefore, observe

$$\begin{aligned} F_i^{(8)\top} + F_i^{(6)} = 0_{o_1 \times o_2} = \sum_{j=1}^{m} \tilde{s}_{ij} \Big(& T_1^\top (P_j^{(1)\top} + P_j^{(1)}) T_4 + T_1^\top (P_j^{(2)}) T_3 + \dots \\ & \dots P_j^{(2)\top} T_4 + (P_j^{(5)\top} + P_j^{(5)}) T_3 + T_1^\top P_j^{(3)} + P_j^{(6)} \Big). \end{aligned} \tag{14}$$

While it is true that only linear equations remain, after inserting the known values for T_3, T_4, and S, one can deduce from

$$T_1^\top \sum_{j=1}^{m} \tilde{s}_{ij} \left((P_j^{(1)\top} + P_j^{(1)}) T_4 + (P_j^{(2)}) T_3 + P_j^{(3)} \right) \overset{(12)}{=} T_1^\top 0,$$

that Eq. (14) does not provide further information about the block T_1, since it is satisfied independent of its choice. The authors of [31] and [24] confirmed our findings in this regard.

Thus, one way to proceed and obtain T_1, is to fall back to the well-known Kipnis-Shamir attack on UOV [20]. Note that the knowledge of S is equivalent to the recovery of the secret subspace W, referring to the notation in Remark 1. Following [3, Section 5.3], this reduces the problem of finding O_1 to a small UOV instance with reduced parameter $n' = n - o_2$ and $m' = m - o_2$ and complexity $\mathcal{O}(q^{n'-2m'}) = \mathcal{O}(q^{n+o_2-2m}) = \mathcal{O}(q^{v_1-o_1})$. In case of Rainbow parameter set I, this leads to a very efficient method to finish the key recovery attack, since it holds $n' \approx 2m'$. For higher parameter sets this approach still remains infeasible, as we have $n' \gg 2m'$, rendering the Kipnis-Shamir attack inefficient.

Very recently, Esser et al. published a work on partial key exposure attacks [16], in which they cover Rainbow, among other schemes. They also treat the task of exposing T_1 after the remaining part of the secret matrices are known. Their approach builds up on a work by Billet and Gilbert [5] and has complexity $\mathcal{O}(n^3)$, which would be very efficient, even for larger parameter sets of Rainbow.

Attack 2: Secret Key Recovery by Skipping the Linear Transformation S. After successful execution of the fault attack, the attacker takes the generated faulted signatures $z^{(1)}, \ldots, z^{(o_2)}$ and the used starting values $y^{(1)}, \ldots, y^{(o_2)}$ being of the form described above and performs the following steps:

1. Compute $w^{(i)} = \mathcal{P}(z^{(i)})$ for $i \in \{1, \ldots, o_2\}$.
2. Build the matrices $W \in M_{m \times o_2}(\mathbb{F}_q)$ and $Y \in M_{m \times o_2}(\mathbb{F}_q)$ as described for Eq. (13).
3. Recover S by computing $S_1 = (W_1 - Y_1) * Y_2^{-1}$.
4. Insert S into Eq. (12). Solve the resulting system of linear equations to recover T_3 and T_4.
5. Obtain T by applying the Kipnis-Shamir attack to the reduced UOV instance.
6. Use Eq. (6) to obtain $\mathcal{F}$. The attacker recovered the full secret key $(S, \mathcal{F}, T)$.

4 Code Analysis and Simulation

To implement the attacks described in Sect. 3, an in-depth analysis of the instruction code needs to be performed. The following section discusses how to uncover the low-level instructions that need to be skipped to achieve the

desired behaviour of the fault attacks as specified in Sect. 3 based on the source code of the ARM Cortex M4 optimized round 3 submission by the authors of [12][3]. Furthermore, we present an elaborated simulation of our results.

4.1 Attack 1: Fixing the Vinegar Variables

Listing 1.1 shows the relevant code snippet for our first attack. The implementation proposed by [12] does not set the vinegar variables to zero after signature generation. Therefore skipping the function call to *prng_gen* in line 55 will leave them with the same values due to the temporary variable being reallocated to the same address at each function call. This, of course, assumes that the respective memory region is not overwritten between two function calls, which holds if the device acts solely as a signing oracle.

By analyzing the disassembly of the compiled binary, we find the relevant instruction given in Listing 1.2. By skipping the branch performed in line 0xdfb2, the desired behaviour is achieved and the vinegars remain constant for subsequent signatures.

4.2 Attack 2: Skipping the Linear Transformation S

To prohibit the application of the linear transformation S^{-1} we aim at skipping the function call to *gf256v_add* in line 178 of the source code shown in Listing 1.1. However, for this function being *inlined* - meaning the compiler inserts the function body instead of a branch - a single instruction skip does not suffice. Therefore the beforehand executed call to *gf16mat_prod_16_32* in line 173 is skipped, leaving the variable *temp_o* at its initial all-zero value and rendering the subsequent call to *gf256v_add* without effect. To achieve this effect, we target line 0xe070 of the assembly code shown in Listing 1.2 with a first order fault.

Exiting the While Loop. In this paragraph, we discuss the probability of exiting the respective while loop on the first iteration, assuming that the fault injection was successful. Regarding the attack in Sect. 4.1, if the skip of the vinegar variables in Line 2 of Algorithm 1 is introduced successfully, the same vinegar variables are used again for consecutive signatures. Thus, the chosen vinegar variables already led to an invertible matrix $\hat{F}_1$ in the previous signature. Since $\hat{F}_1$ only depends on the vinegar variables $(y_1, \ldots, y_{v_1})$ and the polynomials of the first layer, the condition in Line 8 is always fulfilled. Regarding the attack in Sect. 4.2, the condition in Line 11 also depends on the solution of the first layer in Line 9. Thus the probability of $\hat{F}_2$ to be invertible can be approximated by the probability of that a randomly generated matrix with entries in $\mathbb{F}_q$ is invertible which is given by Eq. (3).

[3] The source code can be found at https://github.com/rainbowm4/rainbowm4.git.

```
 26  int rainbow_sign(...)
 27  {
...       ...
 51      while (!l1_succ) // until solution found
 52      {
...           ...
...           // skipped by Attack 1
 55           prng_gen(&prng_sign, vinegar, _V1_BYTE);
...           ...
 80      }
...       ...
...       // temp_o is initialized with zeros
155       uint8_t temp_o[_MAX_O_BYTE + 32] = {0};
...       ...
157       while (!succ) // until solution found
158       {
...        // skipped by Attack 2
173        gf16mat_prod_16_32(temp_o, sk->s1, _z + _O1_BYTE);
...        ...
...        // applying S
178        gf256v_add(y, temp_o, _O1_BYTE);
...        ...
228       }
...   ...
292   }
```

Listing 1.1. Relevant snippets of the rainbow_sign function in *rainbowm4/crypto_sign/rainbowI-classic/m4/rainbow.c*

```
...
0xdfb0  add         signature,sp,#0xe4
0xdfb2  bl          prng_gen
0xdfb6  add         _digest,sp,#0x6c
...
0xe06e  add         signature,sp,#0x144
0xe070  bl          gf16mat_prod_16_32
0xe074  ldr         sk,[sp,#y[4]]
...
```

Listing 1.2. Relevant snippets of the assembly code corresponding to line 55 and line 173 of the rainbow_sign function in Listing 1.1.

4.3 Simulation

To verify our assumptions and provide a first proof of concept, we implement a generic ARM M4 architecture simulation environment based on Unicorn [13], which itself is based on QEMU [2][4]. The validity of our results exceed the ones one would obtain by simple code modification - i.e., removing code lines one

[4] The codebase of our framework can be found at https://anonymous.4open.science/r/double_rainbow_submission-E3CC.

wishes to skip - as the compiled binary of the unmodified source code is executed within our simulation just as a real device would execute it. The 32-bit Reduced Instruction Set Computer (RISC) architecture as defined by ARM is emulated in its entirety.

The framework allows per-instruction execution of the compiled binary, cycle-accurate skipping faults and memory analysis at any given point during execution. This facilitates the validation of both attacks' feasibility through injection of the intended faults and subsequent analysis of the memory space mapped to the vinegar variables and y for the first and second attack, respectively. After verification of the skipping faults' effects on memory, signature collection is performed. Both attacks lead to successful recovery of the secret key, proving the feasibility of our attacks. In the following we give a brief overview of the core features of the simulation framework.

Key Generation. For the generation of the public and secret key being computationally expensive and very time consuming within the simulation, we implement it on the host machine and subsequently map the keys to the simulated device memory.

Signing. The signing algorithm runs entirely within the simulation. Upon executing the binary starting from the respective function's memory address, the secret key is mapped to the simulated device's memory. The address of the memory region holding the message to be signed, a buffer for the result and the key's address are written to the corresponding registers according to the calling convention. To implement the attacks, the simulation first stops at the address where we want to inject the fault. Then the instruction pointer is incremented as required by the length of the instruction to be skipped. Execution is subsequently resumed at the following instruction.

Verification. For completeness, verification inside the simulation is also implemented. Of course, the adversary may implement verification on any device. It is merely used to verify successful fault injection and extract temporary variables that facilitate executing Attack 2.

4.4 Applicability to Other Implementations

The attack we introduced in Section 3.1 is not directly applicable to the reference implementation of Rainbow that was submitted to the NIST Standardization Process [14]. This is due to the fact that the vinegar variables are zeroed at the end of the signing process there, so they can't be reused in a subsequent signing process by a first-order fault attack. See Sect. 5.1 for more details. The second attack, however, can be applied to the reference implementation, since the same steps as mentioned in Listing 1.1 are executed there.

5 Countermeasures

Countermeasures attempt to either verify the integrity of the executed algorithm before returning its result or ensuring that a system cannot leak secrets even when compromised. If the latter is the case, the returned value should either be a random number or an error constant. A traditional way to tackle this problem for the case of fault injections is to repeat the computation and compare the results. However, this approach is very expensive in terms of computation time and relies on the assumption that an attacker will fail to successfully inject faults in two subsequent runs of the algorithm. In this section we suggest countermeasures that can be adopted in order to avoid the attacks described in Sect. 3.

5.1 Countermeasures for Attack 1

For the first attack relying on keeping the vinegar variables constant, some countermeasures aiming for either zeroing or randomization can be employed.

Firstly, resetting the memory region mapped to the vinegars at the end of the function call to zero - as it is done in the original NIST submission in [15] - is the most straight-forward solution. Then, if the respective fault is injected, the system of equations is rendered non-solvable, leading to re-iteration of the loop until either a threshold number of iterations is reached and the function is exited or the fault injection fails and vinegar variables are sampled correctly. However, depending on the implementation, this can enable a different attack of higher complexity relying on partial zeroing of the vinegars, as described in [27].

Secondly, if the vinegars were to be saved in between subsequent function calls, they could be checked for equivalence before returning the signature. While this might seem a viable solution, care must be taken to ensure safe storage not to leak their values. Moreover, since simple checks are assumed to be easily skippable, the checking procedure has to be elegantly integrated in the signing procedure.

Thirdly, inlining the function call to *prng_gen* could prohibit the attack for some parameter sets. Depending on the implementation and the corresponding number of vinegars, the loop copying the random values to the vinegars can be exited earlier by injecting a fault, leaving them partially constant. While this prohibits our attack which requires all vinegars to stay constant, similar attacks with less stringent constraints might still be applicable. To further mitigate these, loop unrolling could provide remedy. However, this combined mitigation technique would introduce non-negligible overhead in code size which might render it inapplicable to constrained devices.

5.2 Countermeasures for Attack 2

For the second attack that aims to skip the application of the secret transformation S, an evident mitigation technique is to verify the signature before returning it. However, this leads to an overhead of around 25% [12], rendering this strategy very costly.

More practical, one could initialize the *temp_o* variable so that the skipping fault would result to an all-zero y after execution of *gf256v_add*. To achieve this, *temp_o* first $o_1/2$ bytes - i.e., the bytes that are affected by the subsequent addition - are initialized with y's first $o_1/2$ bytes (i.e., 16 for the parameter set of 32 oil variables in the bitsliced representation). The subsequent $\mathbb{F}_{256}$ addition - i.e., implemented as multiple consecutive binary XORs - then leads to an all-zero y, prohibiting leakage of the secret key through the collected signatures.

Furthermore, inlining the call to *gf16mat_prod_16_32* would prohibit a first-order skipping fault attack. However, due to this function being implemented in assembly language for optimization purposes, there is a discrepancy for the required build steps. Therefore this mitigation technique might not be trivial to implement.

6 Conclusion

This paper demonstrated how important it is to protect the secret transformations S and T in multivariate schemes against fault attacks. They are the only obstacles an attacker faces when trying to discover the structure of the central map $\mathcal{F}$. Due to their linearity, it is possible to recover them either partially (see Sect. 3.1) or in total (see Sect. 3.2), by collecting enough input and output vectors and analyzing their transformation. As the generated signature constitutes the output of T^{-1} and the hash value that is to be signed represents the input to S^{-1}, an attacker only needs to obtain an intermediate result, e.g., the input vector of T^{-1} or the output vector of S^{-1}, in order to gain secret information. If she is able to thoughtfully induce a fault that compromises one of these intermediate vectors, either by skipping a code line or forcing the algorithm to compute with the same values over and over again, the security of the scheme is no longer guaranteed.

For instance, it was already shown in [27] that UOV and LUOV are vulnerable to the attack that fixes the vinegar variables. Whereas the authors of [22] doubt that it is possible to fix a large portion of the vinegar variables by physical fault injection, we showed that this is indeed possible by a single instruction skip.

Specifically for Rainbow, we proved that it is not even necessary to recover the whole secret transformation T, by the means of a fault attack. The introduced algebraic attack restores the complete secret key of Rainbow on input of the submatrix T_1 by just solving linear equations. This is of course not limited to the evaluation of fault attacks, but also holds if T_1 is leaked through any other kind of side-channel analysis. In the light of the recent breakthrough in cryptanalysis [4], we acknowledge that the Rainbow parameter set for security level I is deprecated. However, the fault attacks we suggest, directly reveal either the secret subspaces O_1 (see Sect. 3.1) or W (see Sect. 3.2) and thus, work for any given parameter set, in particular for higher security levels and adapted parameters that are designed to meet new requirements.

Acknowledgement. This research work has been funded by the German Ministry of Education, Research and Technology in the context of the project Aquorypt (grant number 16KIS1022) and the project Full Lifecycle Post-Quantum PKI - FLOQI (grant number 16KIS1074). Furthermore, we want to thank Enrico Thomae and Namhun Koo for the helpful correspondence concerning the algebraic evaluation in Sect. 3.2.

References

1. Post-Quantum Cryptography. NIST Official Website (2021). https://csrc.nist.gov/projects/post-quantum-cryptography
2. Bellard, F.: QEMU, a fast and portable dynamic translator. In: Proceedings of the Annual Conference on USENIX Annual Technical Conference, ATEC 2005, p. 41, USA. USENIX Association (2005)
3. Beullens, W.: Improved cryptanalysis of UOV and rainbow. In: Canteaut, A., Standaert, F.-X. (eds.) EUROCRYPT 2021. LNCS, vol. 12696, pp. 348–373. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77870-5_13
4. Beullens, W.: Breaking rainbow takes a weekend on a laptop. Cryptology ePrint Archive, Report 2022/214 (2022). https://ia.cr/2022/214
5. Billet, O., Gilbert, H.: Cryptanalysis of rainbow. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 336–347. Springer, Heidelberg (2006). https://doi.org/10.1007/11832072_23
6. Bindel, N., Buchmann, J., Krämer, J.: Lattice-based signature schemes and their sensitivity to fault attacks. In: 2016 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2016, Santa Barbara, CA, USA, 16 August 2016, pp. 63–77. IEEE Computer Society (2016)
7. Blömer, J., Da Silva, R.G., Günther, P., Krämer, J., Seifert, J.P.: A practical second-order fault attack against a real-world pairing implementation. In: 2014 Workshop on Fault Diagnosis and Tolerance in Cryptography, pp. 123–136. IEEE (2014)
8. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of checking cryptographic protocols for faults. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 37–51. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-69053-0_4
9. Campos, F., Krämer, J., Müller, M.: Safe-error attacks on SIKE and CSIDH. In: Batina, L., Picek, S., Mondal, M. (eds.) SPACE 2021. LNCS, vol. 13162, pp. 104–125. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-95085-9_6
10. Castelnovi, L., Martinelli, A., Prest, T.: Grafting trees: a fault attack against the SPHINCS framework. In: Lange, T., Steinwandt, R. (eds.) PQCrypto 2018. LNCS, vol. 10786, pp. 165–184. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-79063-3_8
11. Cayrel, P.-L., Colombier, B., Drăgoi, V.-F., Menu, A., Bossuet, L.: Message-recovery laser fault injection attack on the *Classic McEliece* cryptosystem. In: Canteaut, A., Standaert, F.-X. (eds.) EUROCRYPT 2021. LNCS, vol. 12697, pp. 438–467. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77886-6_15
12. Chou, T., Kannwischer, M.J., Yang, B.Y.: Rainbow on cortex-M4. Cryptology ePrint Archive, Report 2021/532 (2021). https://ia.cr/2021/532
13. Quynh, N.A., Vu, D.H.: Unicorn: next generation CPU emulator framework (2015)
14. Ding, J., et al.: Rainbow. Technical report, National Institute of Standards and Technology (2020). https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions

15. Ding, J., Schmidt, D.: Rainbow, a new multivariable polynomial signature scheme. In: Ioannidis, J., Keromytis, A., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 164–175. Springer, Heidelberg (2005). https://doi.org/10.1007/11496137_12
16. Esser, A., May, A., Verbel, J., Wen, W.: Partial key exposure attacks on BIKE. Rainbow and NTRU, Cryptology ePrint Archive (2022)
17. Grimes, R.A.: Cryptography Apocalypse: Preparing for the Day When Quantum Computing Breaks Today's Crypto. Wiley, Hoboken (2019)
18. Hashimoto, Y., Takagi, T., Sakurai, K.: General fault attacks on multivariate public key cryptosystems. In: Yang, B.-Y. (ed.) PQCrypto 2011. LNCS, vol. 7071, pp. 1–18. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25405-5_1
19. Kalai, G.: The argument against quantum computers, the quantum laws of nature, and Google's supremacy claims. arXiv preprint arXiv:2008.05188 (2020)
20. Kipnis, A., Patarin, J., Goubin, L.: Unbalanced oil and vinegar signature schemes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 206–222. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_15
21. Krämer, J., Loiero, M.: Fault attacks on UOV and rainbow. In: Polian, I., Stöttinger, M. (eds.) COSADE 2019. LNCS, vol. 11421, pp. 193–214. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-16350-1_11
22. Mus, K., Islam, S., Sunar, B.: QuantumHammer: a practical hybrid attack on the LUOV signature scheme. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, pp. 1071–1084 (2020)
23. Okeya, K., Takagi, T., Vuillaume, C.: On the importance of protecting Δ in SFLASH against side channel attacks. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **88**(1), 123–131 (2005)
24. Park, A., Shim, K.A., Koo, N., Han, D.G.: Side-channel attacks on post-quantum signature schemes based on multivariate quadratic equations:-rainbow and UOV. IACR Trans. Cryptographic Hardware Embed. Syst. 500–523 (2018)
25. Pokornỳ, D., Socha, P., Novotnỳ, M.: Side-channel attack on rainbow post-quantum signature. In: 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 565–568. IEEE (2021)
26. Roetteler, M., Naehrig, M., Svore, K.M., Lauter, K.: Quantum resource estimates for computing elliptic curve discrete logarithms. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10625, pp. 241–270. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70697-9_9
27. Shim, K.-A., Koo, N.: Algebraic fault analysis of UOV and rainbow with the leakage of random vinegar values. IEEE Trans. Inf. Forensics Secur. **15**, 2429–2439 (2020)
28. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Rev. **41**(2), 303–332 (1999)
29. Steinwandt, R., Geiselmann, W., Beth, T.: A theoretical DPA-based cryptanalysis of the NESSIE candidates FLASH and SFLASH. In: Davida, G.I., Frankel, Y. (eds.) ISC 2001. LNCS, vol. 2200, pp. 280–293. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45439-X_19
30. Tao, C., Petzoldt, A., Ding, J.: Improved key recovery of the hfev-signature scheme. Cryptology ePrint Archive (2020)
31. Thomae, E.: A generalization of the rainbow band separation attack and its applications to multivariate schemes. Cryptology ePrint Archive (2012)
32. Yi, H., Li, W.: On the importance of checking multivariate public key cryptography for side-channel attacks: the case of enTTS scheme. Comput. J. **60**(8), 1197–1209 (2017)

IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2023, No. 3, pp. 221–245. DOI:10.46586/tches.v2023.i3.221-245

Separating Oil and Vinegar with a Single Trace

Side-Channel Assisted Kipnis-Shamir Attack on UOV

Thomas Aulbach[1], Fabio Campos[2,3], Juliane Krämer[1], Simona Samardjiska[3] and Marc Stöttinger[2]

[1] University of Regensburg, Regensburg, Germany
{thomas.aulbach,juliane.kraemer}@ur.de
[2] RheinMain University of Applied Sciences, Wiesbaden, Germany
campos@sopmac.de,marc.stoettinger@hs-rm.de
[3] Radboud University, Nijmegen, Netherlands
simonas@cs.ru.nl

Abstract.
Due to recent cryptanalytical breakthroughs, the multivariate signature schemes that seemed to be most promising in the past years are no longer in the focus of the research community. Hence, the cryptographically mature UOV scheme is of great interest again. Since it has not been part of the NIST process for standardizing post-quantum cryptography so far, it has not been studied intensively for its physical security.
In this work, we present a side-channel attack on the latest implementation of UOV. In the first part of the attack, a single side-channel trace of the signing process is used to learn all vinegar variables used in the computation. Then, we employ a combination of the Kipnis-Shamir attack and the reconciliation attack to reveal the complete secret key. Our attack, unlike previous work, targets the inversion of the central map and not the subsequent linear transformation. It further does not require the attacker to control the message to be signed.
We have verified the practicality of our attack on a ChipWhisperer-Lite board with a 32-bit STM32F3 ARM Cortex-M4 target mounted on a CW308 UFO board. We publicly provide the code and both reference and target traces. Additionally, we discuss several countermeasures that can at least make our attack less efficient.

Keywords: Multivariate signature schemes · UOV · Side-channel attack · Kipnis-Shamir attack · Reconciliation attack

1 Introduction

In July 2022, the National Institute of Standards and Technology (NIST) announced four post-quantum schemes which will soon be standardized, three of which are signature schemes. Two of these signature schemes - Dilithium [DKL+18] and Falcon [PFH+22] - are based on lattices and the third one - SPHINCS+ [BHK+19] - is hash-based. While lattice-based schemes are generally considered to be widely applicable, SPHINCS+ signatures are huge, consuming several KB even for the smallest parameter set. This makes SPHINCS+ useful mainly for specific applications.

Overall, given that currently five families of post-quantum cryptographic assumptions are studied, the three selected signature schemes from only two families imply a lack of diversity. This is problematic for at least two reasons: On the one hand, signature schemes from all five families feature different advantages such as small keys, small signatures, or small computation time. Hence, having standardized signature schemes from more than

Received: 2023-01-15 Accepted: 2023-03-15 Published: 2023-06-09

two families would allow to choose the optimal scheme for each application. On the other hand, future breakthroughs in (quantum) cryptanalysis might considerably decrease the hardness of post-quantum assumptions, hence decrease the security of the schemes built on them. Having standardized signature schemes from more than two families would provide alternative standardized signature schemes in case one class of assumptions turns out to provide less security than expected, instead of mainly relying on the security of lattices. NIST reacted to this lack of diversity by an explicit call for new signature proposals for the 4th round. For the reasons stated above, NIST is especially interested in signature schemes that are not based on lattices and have small signatures.

Multivariate signature schemes in general feature very small signatures, and the two multivariate signature schemes Rainbow [DCP+20] and GeMSS [CFM+20] also advanced to the third round of NIST's standardization process for post-quantum cryptography (PQC). However, powerful attacks against the third round alternate candidate GeMSS by Tao et al. [TPD21] and the third round finalist Rainbow by Beullens [Beu22a] showed that these two schemes should not be standardized. At this stage, the most relevant signature schemes in the field of multivariate cryptography are MAYO [Beu22b] and UOV [KPG99]. Although UOV has already been published at the end of the 1990s and is the basis for Rainbow, research concentrated on Rainbow after its publication because it is more efficient than UOV both in terms of required memory and computation time. Although Rainbow is a generalization of the oil-and-vinegar construction underlying UOV, Beullen's attack on Rainbow does not apply to UOV. This makes UOV again a very interesting signature scheme since it withstands cryptanalysis since nearly two decades and has very small signatures. Consequently, it will be submitted to the 4th round of NIST's PQC standardization process. Already now, a paper describing modern parameters and implementations of UOV exists [BCH+23].

Since UOV has initially not been submitted to the NIST PQC standardization process, UOV has also not been in the focus of physical security research until now. Therefore, we set out to analyze the physical security of UOV. In this paper, we propose the first single-trace side-channel attack (SCA) on UOV, targeting the latest UOV implementation [BCH+23, UOV23].

Related Work Since Rainbow was a third round finalist until 2022, most existing results on the physical security target Rainbow and not UOV. Although both schemes are very similar, not all known results for Rainbow can be transferred to UOV, since UOV does not use the layer structure and the second affine transformation S which mixes the quadratic polynomial equations. Interesting physical attacks have also been proposed against LUOV [BPSV19], an adaptation of the UOV signature scheme that advanced to the second round of NIST's PQC standardization process. We first state results for Rainbow and LUOV that cannot be transferred to UOV and then those which can be transferred to UOV or have been explicitly developed for UOV.

In 2020, Villanueva-Polanco described how to reveal a complete LUOV secret key by a cold boot attack [VP20], i.e., in a setting where an adversary can learn a noisy version of the secret key by cold booting the target device. In the same year, Mus et al. published a hybrid attack on LUOV [MIS20] where they first collect signatures that have been incorrectly computed due to Rowhammer fault injection and then reveal the complete secret key in a divide-and-conquer attack. A correlation power analysis on Rainbow was presented in 2021 by Pokorný et al. [PSN21]. The attack is based on a known message and needs a few hundred power traces to first recover the maps S and T and then reveal the central map F. In 2022, two fault attacks on Rainbow have been presented by Aulbach et al. [AKKM20]. The first attack leads to partial leakage of the secret transformation T by fixing vinegar variables. The second attack induces faults during the application of the linear transformation S. Both attacks eventually lead to full key recovery.

In 2018, Park et al. presented side-channel attacks on Rainbow and UOV [PSKH18]. The attacks are similar to the one described in [PSN21]. They use correlation power analysis together with algebraic key-recovery attacks and demonstrate the practical feasibility of their attack on an 8-bit AVR microcontroller. Again, the attack is based on known messages and first reveals the map S and then the map T. The attack is described for Rainbow but can be transferred to UOV when UOV is implemented with equivalent keys.

Regarding fault attacks not specific to Rainbow, three publications are interesting: Hashimoto et al. described general methods how to attack multivariate cryptography with fault attacks already in 2011 [HTS11]. Based on these ideas, Krämer and Loiero presented two fault attacks on UOV and Rainbow in 2019 [KL19]. In the first attack (to which UOV is immune), a coefficient of the central map F is randomized and in the second attack, vinegar variables are fixed. Just recently, another fault attack on UOV was published [FKNT22], again based on the ideas of [HTS11]. In this attack, a single coefficient of the secret key is faulted.

Contribution In this work, we present the first single-trace side-channel attack against UOV. Our attack targets the latest implementation of UOV [BCH+23, UOV23] and leads to full key recovery. Contrary to existing work, we target the inversion of the central map during signature generation, not the subsequent linear transformation. Since the target routine of our side-channel attack is the multiplication of (secret) vinegar variables with UOV's secret key, the message to be signed does not need to be controlled nor to be known by the attacker. The attack consists of three steps: First, a single side-channel trace of the inversion of the central map during signature generation allows us to recover the (secret) vinegar variables that are used during the signing process. Therefore, we make use of a correspondence between the public and the secret key, due to the special choice of the linear transformation T. This correspondence exists for both existing versions of UOV - the standard and the compressed version. Then, we use these vinegar variables to recover a vector that is annihilated by the public key, i.e., a vector of the secret linear oil space $\boldsymbol{O}$. In the third step this allows us to apply a modified version of the Kipnis-Shamir attack on reduced parameters that runs in polynomial time and reveals a second oil vector. For all given parameter sets these two oil vectors bear enough information to compute the remaining oil space, i.e., the secret key, by employing an efficient reconciliation attack that only requires us to solve linear equations.

We perform the attack practically on a ChipWhisperer-Lite board with a 32-bit STM32F3 ARM Cortex-M4 target mounted on a CW308 UFO board. The code can be found here: https://github.com/mstoetti/SCA_assisted_recon_UOV. We collected reference traces on a profiling device and attack traces on a target device.

Additionally, we provide scripts for collecting reference and target traces on the reader's own ChipWhisperer Setup, if available. The template attack can then be executed, either with the power traces we provide, or with the ones the reader collects. Furthermore, we adapted the Kipnis-Shamir attack [KS06] and the reconciliation attack to accept oil vectors as additional input in order to reduce the complexity. When the side-channel attack recovered an oil vector successfully, the script performs the algebraic attacks to obtain the complete oil space.

We suggest several countermeasures: First we introduce a countermeasure that breaks the correspondence between the public key and the secret key which makes our attack so strong. However, we present a detailed analysis of how the attack can be adapted in that case. Hence, removing this correspondence does not fully prevent the attack, but makes it considerably less powerful. Then we show how known countermeasures such as masking, shuffling, and using precomputations can be applied to UOV.

Organization The rest of this paper is organized as follows: In Section 2, we introduce the UOV algorithm and further background information that is relevant for this work. In Section 3, we present the theoretical part behind our side-channel attack. We present the critical correspondence between the secret key and the public key and show how we can reveal the complete secret UOV key from a single side-channel trace. In Section 4, we present the practical attack. We describe the setup, discuss the parameters we attacked, provide power traces, and discuss how we dealt with noise and what the expected noise resistance of our approach is. Finally, we present countermeasures against our attack in Section 5.

2 Background

In this section, we provide the background knowledge necessary to understand the announced attack. We will present the UOV signature scheme [KPG99], both in its traditional description and the one recently introduced by Beullens [Beu21]. They are equivalent, but facilitate our understanding of different aspects of the attack. Furthermore, we specify details of a major step of the signing process in UOV, the inversion of the central map. Since our attack targets a specific subroutine thereof, it is inevitable to examine this step more closely. The section is concluded by an elaborated presentation of the Kipnis-Shamir attack [KPG99] and the reconciliation attack [DYC+08]. The additional information gained through side-channel leakage, reduces the complexity of the algebraic attacks significantly and allows for an efficient recovery of the complete secret key.

Notation Let $\mathbb{F}_q$ be a finite field with q elements. Let n and m be two positive integers with $n > m$ and $v = n - m$. Subspaces of vector spaces over $\mathbb{F}_q$ are written in bold capital letters, e.g., $\boldsymbol{O} \subseteq \mathbb{F}_q^n$ and their elements have bold letters $\boldsymbol{x}$ with i-th coordinate x_i. Multivariate quadratic maps between those spaces are denoted in calligraphic font $\mathcal{P}$. Their coefficients can be stored in a collection of matrices with capital letters and enumerating superscripts $P^{(k)}$. The matrices we use often have block structure, so we use $P_i^{(k)}$ to denote the submatrices or $0_{m\times v}$ for the zero matrix and I_v for the identity matrix of a certain size.

2.1 Unbalanced Oil and Vinegar Signature Scheme

The essence of UOV consists of a multivariate quadratic map $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ that contains a certain elegant trapdoor, namely that $\mathcal{P}$ vanishes on a secret linear subspace $\boldsymbol{O} \subset \mathbb{F}_q^n$ of dimension $\dim(\boldsymbol{O}) = m$. This trapdoor information allows for efficiently obtaining solutions $\boldsymbol{x} \in \mathbb{F}_q^n$ of $\mathcal{P}(\boldsymbol{x}) = \boldsymbol{y} \in \mathbb{F}_q^m$. Without this trapdoor information, finding preimages of $\mathcal{P}$ boils down to solving the multivariate quadratic polynomial (MQ) problem for the system of quadratic equations given by $\mathcal{P}(\boldsymbol{x}) = \boldsymbol{y}$, which is assumed to be hard. There are two common ways to describe how this trapdoor function facilitates the construction of a signature scheme. We will present both of them in the following.

2.1.1 Traditional Description

For around two decades, researchers commonly utilized another multivariate quadratic map $\mathcal{F} : \mathbb{F}_q^n \to \mathbb{F}_q^m$, the so-called central map, in order to specify UOV. The polynomials of this map $\mathcal{F} = (f^{(1)}, \ldots, f^{(m)})$ are defined by

$$f^{(k)}(\boldsymbol{x}) = \sum_{1\le i\le n} \sum_{1\le j\le v} \alpha_{i,j}^{(k)} x_i x_j \tag{1}$$

for $k \in \{1, \ldots, m\}$, where $\alpha_{i,j}^{(k)} \in \mathbb{F}_q$ represent the coefficients of each quadratic polynomial. Observing the constrains on the indices in Equation (1), this reveals that the linear subspace of dimension m, which consists of vectors with zeros in the first v coordinates, is annihilated by $\mathcal{F}$. We denote this subspace by

$$\boldsymbol{O'} = \{\boldsymbol{x} \mid x_i = 0 \text{ for all } 1 \leq i \leq v\}.$$

The central map $\mathcal{F}$ is concatenated with a random linear transformation $T : \mathbb{F}_q^n \to \mathbb{F}_q^n$, that is supposed to hide the specific structure of $\mathcal{F}$ from anyone who has only access to the concatenation $\mathcal{P} = \mathcal{F} \circ T$. The resulting public key map $\mathcal{P}$ vanishes on the linear subspace $\boldsymbol{O} \coloneqq T^{-1}(\boldsymbol{O'})$. Since all available instantiations of UOV only consider homogeneous polynomials, the coefficients from Equation (1) can be stored in matrices $F^{(k)}$ such that evaluating the polynomial $f^{(k)}$ in $\boldsymbol{x}$ is equivalent to computing $\boldsymbol{x}^\top F^{(k)} \boldsymbol{x}$ for all $k \in \{1, \ldots, m\}$. Similarly, we can obtain $n \times n$ matrices $P^{(k)}$ with $p^{(k)}(\boldsymbol{x}) = \boldsymbol{x}^\top P^{(k)} \boldsymbol{x}$ for the public key polynomials $\mathcal{P} = (p^{(1)}, \ldots, p^{(m)})$. Thus, from $\mathcal{P} = \mathcal{F} \circ T$, we have

$$P^{(k)} = T^\top F^{(k)} T. \tag{2}$$

By fixing the first v entries $(\tilde{x_1}, \ldots, \tilde{x_v})$ - the so-called vinegar variables - in $\boldsymbol{x}$ to certain random elements in $\mathbb{F}_q$ and applying them to Equation (1), we receive polynomials $(\hat{f}^{(1)}, \ldots, \hat{f}^{(m)})$. These constitute a linear system of m equations in the remaining m variables $(x_{v+1}, \ldots, x_n)$ - the so-called oil variables - of $\boldsymbol{x}$. This system is solvable with quite a high probability and therefore explains why it is possible to find preimages under $\mathcal{F}$.

Key Generation We show a natural method for generating key pairs in Algorithm 1. For any public key map $\mathcal{P}$, there exists an equivalent secret key $(\mathcal{F}, T)$ with

$$T = \begin{pmatrix} I_{v \times v} & T_1 \\ 0_{m \times v} & I_{m \times m} \end{pmatrix}. \tag{3}$$

Thus, in order to obtain a key pair, we first randomly generate T_1. For the central map $\mathcal{F}$ it suffices to randomly generate upper triangular matrices $F^{(k)} \in \mathbb{F}_q^{n \times n}$ since the coefficients $\alpha_{i,j}$ and $\alpha_{j,i}$ can be grouped together. Furthermore, there is a zero block, since we have no quadratic oil terms in Equation (1). Consequently, we only need to generate upper triangular blocks $F_1^{(k)} \in \mathbb{F}_q^{v \times v}$ and blocks $F_2^{(k)} \in \mathbb{F}_q^{v \times m}$ for $k \in \{1, \ldots, m\}$. Then, compute $P^{(k)}$ by applying Equation (2) and store the final coefficients again in upper triangular form. There are ways to reduce the key sizes, by, e.g., storing a seed instead of the matrices or storing only T_1 as the secret key. This will be discussed in more detail in Section 3.1.

Signature Generation and Verification Signature generation is displayed in Algorithm 2. To sign a message d, one needs to find a preimage of $\boldsymbol{y} = \mathcal{H}(\mathcal{H}(d)||salt) \in \mathbb{F}_q^m$. This can be done as described above, by turning $\mathcal{F}$ into an invertible linear map $\hat{\mathcal{F}}$. The vinegar and oil variables together represent a solution of $\boldsymbol{x} = \mathcal{F}^{(-1)}(\boldsymbol{y})$. In the next section we will present some algorithmic details of how the described linear system is generated. Finally, we obtain $\boldsymbol{z} \in \mathbb{F}_q^n$ that fulfills $\mathcal{P}(\boldsymbol{z}) = \boldsymbol{y}$ by computing $\boldsymbol{z} = T^{-1}(\boldsymbol{x})$.

Signature Verification boils down to verifying that $\boldsymbol{z}$ is indeed a preimage of $\boldsymbol{y}$ under $\mathcal{P}$.

2.1.2 Beullens' Description

In [Beu21] the author introduces an approach that omits the central map. Here, signing is facilitated directly by knowledge of the secret linear oil space $\boldsymbol{O}$ of dimension m. To this end, consider the polar form of a homogeneous quadratic polynomial defined by

Algorithm 1 UOV Key Generation

Input: Parameters (q, n, v, m)
Output: Key pair (pk, sk)

$T_1 \leftarrow_R \mathbb{F}_q^{v \times m}$
for $k = 1$ `to` m **do**
  $(F_1^{(k)}, F_2^{(k)}) \leftarrow_R (\mathbb{F}_q^{v \cdot (v+1)/2}, \mathbb{F}_q^{v \times m})$
  Build T and $F^{(k)}$ from its blocks, according to Equation (3) and (7)
  Compute $P^{(k)} = T^\top F^{(k)} T$
  $P^{(k)} \leftarrow \texttt{Upper}(P^{(k)})$
end for
$pk \leftarrow \mathcal{P} = (\{P^{(k)}\}_{k=\{1,\dots,m\}})$
$sk \leftarrow (T, \mathcal{F}) = (T, \{F^{(k)}\}_{k=\{1,\dots,m\}})$
return (pk, sk)

Algorithm 2 UOV Signature Generation

Input: message d, private key $(T, \mathcal{F})$, length l of the salt.
Output: signature $\sigma = (\boldsymbol{z}, salt) \in \mathbb{F}_q^n \times \{0,1\}^l$ s.t. $\mathcal{P}(\boldsymbol{z}) = \mathcal{H}(\mathcal{H}(d)||salt)$.

$(x_1, \dots, x_v) \leftarrow_R \mathbb{F}_q^v$
$\hat{F} = \big(\hat{f}^{(1)}, \dots, \hat{f}^{(m)}\big) \leftarrow \big(f^{(1)}(x_1, \dots, x_v), \dots, f^{(m)}(x_1, \dots, x_v)\big)$
if $\operatorname{rank}(\hat{F}) \neq m$ **then**
  return to step 1
end if
$salt \leftarrow_R \{0,1\}^l$
$y \leftarrow \mathcal{H}(\mathcal{H}(d)||salt)$
$x_{v+1}, \dots, x_n \leftarrow \hat{F}^{-1}(y_{v+1}, \dots, y_n)$
$\boldsymbol{z} \leftarrow T^{-1}(\boldsymbol{x})$
$\sigma = (\boldsymbol{z}, salt)$
return σ

$$p'(x, y) \coloneqq p(x + y) - p(x) - p(y) + p(0). \tag{4}$$

The map $p' : \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q^m$ is a symmetric bilinear form. Assume that P is the matrix associated to the polynomial $p(x)$, then there exists a matrix satisfying $p'(x, y) = x^\top P' y$ and this matrix is given by $P' = P + P^\top$. This notion can be extended to the map $\mathcal{P}$ and we write $\mathcal{P}'$ for the corresponding map.

Given a target $\boldsymbol{t} \in \mathbb{F}_q^m$, we can use $\mathcal{P}'$ to find a preimage under $\mathcal{P}$. Therefore, fix an arbitrary vector $\boldsymbol{v} \in \mathbb{F}_q^n$ and solve the system $\mathcal{P}(\boldsymbol{v} + \boldsymbol{o}) = \boldsymbol{t}$ for a vector $\boldsymbol{o} \in \boldsymbol{O}$. This boils down to solving the following linear system of equations

$$\mathcal{P}(\boldsymbol{v} + \boldsymbol{o}) = \underbrace{\mathcal{P}(\boldsymbol{v})}_{\text{constant}} + \underbrace{\mathcal{P}(\boldsymbol{o})}_{=0} + \underbrace{\mathcal{P}'(\boldsymbol{v}, \boldsymbol{o})}_{\text{linear in } o} = \boldsymbol{t}, \tag{5}$$

because by definition $\mathcal{P}$ vanishes on the secret oil space $\boldsymbol{O}$, i.e., $\mathcal{P}(\boldsymbol{o}) = 0$, for all $\boldsymbol{o} \in \boldsymbol{O}$. Since $\dim(O) = m$, Equation (5) is a linear system of m equations in m variables. If it is solvable, we have found a solution $\boldsymbol{v} + \boldsymbol{o} = \boldsymbol{z} \in \mathbb{F}_q^n$, otherwise, one restarts with a new random vinegar vector $\boldsymbol{v}$.

Remark 1. The characterization of UOV from [Beu21] is an instantiation of what is known as (s, t)-linearity in symmetric key cryptography, first introduced by Boura and Canteaut [BC14] and adapted to multivariate cryptography by Samardjiska and Gligoroski [SG14]. In essence, a function $\mathcal{F} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ is said to be (s, t)-linear if there exist two linear

subspaces $\boldsymbol{V} \subset \mathbb{F}_q^n$, $\boldsymbol{W} \subset \mathbb{F}_q^m$ with $\dim(\boldsymbol{V}) = s$, $\dim(\boldsymbol{W}) = t$ such that for all $\boldsymbol{w} \in \boldsymbol{W}$, $\boldsymbol{w}^\intercal \cdot f$ has degree at most 1 on all cosets $\boldsymbol{x} + \boldsymbol{V}$ of $\boldsymbol{V}$. It was shown in [SG14] that UOV is (m, m)-linear, i.e., the public map is linear on any coset of the oil space. This is exactly what Equation (5) tells us.

Boura and Canteaut [BC14] further give a characterization of (s, t)–linearity through second order derivatives defined by $D_{\boldsymbol{a},\boldsymbol{b}} f = D_{\boldsymbol{a}} D_{\boldsymbol{b}} f = D_{\boldsymbol{b}} D_{\boldsymbol{a}} f$ where the first order derivative is simply $D_{\boldsymbol{a}} f = f(\boldsymbol{x} + \boldsymbol{a}) - f(\boldsymbol{x})$. Here, f is (s, t)–linear with respect to $\boldsymbol{V}, \boldsymbol{W}$ if and only if all second order derivatives $D_{\boldsymbol{a},\boldsymbol{b}} w^\intercal \cdot f$, with $\boldsymbol{a}, \boldsymbol{b} \in \boldsymbol{V}$, $\boldsymbol{w} \in \boldsymbol{W}$ vanish.

For UOV, this means that $D_{\boldsymbol{o}_1,\boldsymbol{o}_2} \mathcal{P} = 0$ for all oil vectors $\boldsymbol{o}_1, \boldsymbol{o}_2 \in \boldsymbol{O}$. This result was used in [SG14] to provide an alternative description of the reconciliation attack by Ding et al. [DYC+08], that we describe in Section 2.3.2.

Remark 2. We want to point out a difference in notation to prevent potential misunderstanding. In the traditional description, a vector $\boldsymbol{x} \in \mathbb{F}_q^n$ is split by its entries into vinegar and oil variables. In the above description, on the other hand, there are two vectors $\boldsymbol{v}$ and $\boldsymbol{o} \in \mathbb{F}_q^n$, that are called vinegar vector and oil vector, respectively.

Key Generation First, the user generates an oil space $\boldsymbol{O}$ by sampling a uniformly random matrix $O \in \mathbb{F}_q^{v \times m}$, and letting $\boldsymbol{O}$ be the rowspace of (OI_m). Consequently, a multivariate quadratic polynomial $p_k(\boldsymbol{x})$ vanishes on $\boldsymbol{O}$, if for the associated matrix $P^{(k)}$ it holds

$$\begin{pmatrix} O \\ I_m \end{pmatrix}^\top \begin{pmatrix} P_1^{(k)} & P_2^{(k)} \\ 0_{m \times v} & P_3^{(k)} \end{pmatrix} \begin{pmatrix} O \\ I_m \end{pmatrix} = 0. \tag{6}$$

Thus, in order to reduce the key size, we can expand $P_1^{(k)}$ and $P_2^{(k)}$ from a seed and compute $P_3^{(k)}$, such that Equation (6) is fulfilled. This implies the following algorithm for key generation.

Algorithm 3 UOV Key Generation according to [Beu21]

Input: Parameters (q, n, v, m)
Output: Key pair (pk, sk)

1: $O \leftarrow_R \mathbb{F}_q^{v \times m}$
2: $seed \leftarrow_R \{0,1\}^\lambda$
3: **for** $k = 1$ `to` m **do**
4: $\quad (P_1^{(k)}, P_2^{(k)}) \leftarrow$ `Expand`$(seed||k)$
5: $\quad P_3^{(k)} \leftarrow -(OP_1^{(k)}O^\top + OP_2^{(k)})$
6: **end for**
7: $pk \leftarrow (seed, \{P_3^{(k)}\}_{k=\{1,\ldots,m\}})$
8: $sk \leftarrow (seed, O)$
9: **return** (pk, sk)

Signature Generation and Verification The signing process can be directly derived from Equation (5) and is presented in Algorithm 4. Similar to the traditional description, it is possible that the resulting system of linear equations has no solution. In this case, a new vinegar vector has to be generated. Verification is basically done by evaluating $\mathcal{P}(\boldsymbol{z})$, so there is no need for any adaptions.

2.2 Detailed Description of the Central Map Inversion

In Line 2 of Algorithm 2 the randomly generated vinegar variables are inserted into the central map $\mathcal{F}$ to generate a linear system of equations. In the following we give

Algorithm 4 UOV Signature Generation according to [Beu21]

Input: message d, private key $(seed, O)$, length l of the salt.
Output: signature $\sigma = (\boldsymbol{z}, salt) \in \mathbb{F}_q^n \times \{0,1\}^l$ s.t. $\mathcal{P}(\boldsymbol{z}) = \mathcal{H}(\mathcal{H}(d)||salt)$.

1: $salt \leftarrow_R \{0,1\}^l$
2: $\boldsymbol{y} \leftarrow \mathcal{H}(\mathcal{H}(d)||salt)$
3: $\boldsymbol{v} \leftarrow_R \mathbb{F}_q^{n-m} \times \{0\}^m$
4: **if** $\text{rank}(\mathcal{P}(\boldsymbol{v}+\boldsymbol{o})) \neq m$ **then**
5: return to step 3
6: **end if**
7: Solve $\mathcal{P}(\boldsymbol{v}+\boldsymbol{o}) = \boldsymbol{y}$ for $\boldsymbol{o} \in \boldsymbol{O}$
8: $\sigma \leftarrow (\boldsymbol{z} = \boldsymbol{v}+\boldsymbol{o}, salt)$
9: **return** σ

more details on this procedure, as it contains the routine we target in our suggested side-channel attack later on. As described in Section 2.1, the coefficients of $\mathcal{F}$ are stored in the matrices $F^{(k)}$. Due to the structure of the polynomials in Equation (1), i.e., they do not have quadratic oil terms, these matrices are of the form

$$F^{(k)} = \begin{pmatrix} F_1^{(k)} & F_2^{(k)} \\ 0 & 0 \end{pmatrix}. \tag{7}$$

Thus, substituting the vinegar variables $\tilde{\boldsymbol{x}}$ into the central map amounts to computing $\tilde{\boldsymbol{x}}^\top F_1^{(k)} \tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{x}}^\top F_2^{(k)} \boldsymbol{x'}$, where $\boldsymbol{x'} = \{x_{v+1}, \ldots, x_n\}$ are the remaining oil variables of the linear system of equations. Consequently, inserting the vinegar variables into the central map amounts to performing the algebraic operations indicated above for all matrices $F^{(k)}$, where $k \in \{1, \ldots, m\}$.

2.3 Attacks on the Oil & Vinegar Construction

In the following we describe two well-known algebraic attacks on signature schemes that are based on the Oil and Vinegar principle, since we need modified versions of them to complete our side-channel attack.

2.3.1 Kipnis-Shamir Attack

In the original Oil and Vinegar scheme, the two sets of variables were of the same size, i.e., $m = v$ and $n = 2v$. This revealed some weaknesses that were used by Kipnis and Shamir to break the scheme [KS06]. The main observation of the attack is that the secret oil space is an invariant subspace (an eigenspace) of $P'_{ij} = P'^{-1}_j P'_i$, for any two invertible matrices P'_i and P'_j of the map $\mathcal{P}'$, i.e., $P'^{-1}_j P'_i O = O$. The invariant subspace of the matrices can be found efficiently, for example by looking at the characteristic polynomial of one such matrix P'_{ij}. If the characteristic polynomial factors into two irreducible factors C_1 and C_2 of degree v, then the oil space can be found as the kernel of the matrix $C_1(P'_{ij})$ or $C_2(P'_{ij})$. Kipnis and Shamir [KS06] argue that the probability for such a factorization is high, thus obtaining an efficient algorithm for distilling the oil space.

The unbalanced version, with $n > 2m$ [KPG99] was constructed to prevent this attack. Indeed, in this case two matrices P'_i and P'_j do not necessarily map the oil space into the exact same subspace of the vinegar space. Nevertheless, the intersection of $P'_i O$ and $P'_j O$ is still very high. The authors of [KPG99] show the following crucial result.

Theorem 1 ([KPG99]). *Let Q' be an invertible linear combination of the matrices $P'_1, \ldots, P'_m$. Then for any invertible P'_j, the matrix $P'^{-1}_j Q'$ has a non-trivial invariant subspace, which is also a subspace of O with probability at least $\frac{q-1}{q^{2(n-2m)}-1}$.*

When the difference $n-2m$ is not very big, this probability is high, and one can expect to find this subspace efficiently. In order to do that, the original Kipnis-Shamir attack can be generalized to look for a small invariant subspace of the matrices P'_{ij} that is also a subspace of O. Thus, one can use any factorization of the characteristic polynomial of P'_{ij}, and check whether for any of the factors C, the kernel of $C(P'_{ij})$ is in the oil subspace. Recall that we can efficiently test membership of a vector in the oil space by checking whether the public map vanishes on the said vector.

Note that the procedure needs to be repeated, in order to find the whole oil space, or one can use other methods such as the reconciliation attack (see Section 2.3.2), once a few oil vectors are known.

2.3.2 Reconciliation Attack

Recall from Remark 1 that the public map of UOV is (m,m)-linear. Thus, in order to break the scheme, it is necessary to find a vector space - the oil space $\boldsymbol{O}$, such that $\mathcal{P}$ is (m,m)–linear with respect to $(\boldsymbol{O}, \mathbb{F}_q^m)$.

Ding et al. in [DYC+08] propose an algorithm that sequentially performs a change of basis that reveals gradually the space O. They call the algorithm *Reconciliation Attack on UOV*. In Algorithm 5, we present an equivalent version of the attack interpreted in terms of (s,t)–linearity (cf. Algorithm 2 [DYC+08]).

Algorithm 5 Reconciliation Attack on UOV in terms of (s,t)–linearity

Input: UOV public key $\mathcal{P}: \mathbb{F}_q^n \to \mathbb{F}_q^m$.
Output: An oil space $\boldsymbol{O} = \boldsymbol{O}_m$ of dimension m.

1: $\boldsymbol{O}_0 \leftarrow$ the zero-dimensional vector space
2: **for** $k := 1$ to m **do**
3: Find $\mathbf{o}_k = (o_1^{(k)}, ..., o_v^{(k)}, 0, ..., 0, 1_{n-k+1}, 0, ..., 0) \in \mathbb{F}_q^n$, where 1_{n-k+1} denotes that the $(n-k+1)$-th coordinate is 1, by solving

$$\begin{aligned} \mathbf{o}_j P'_i \mathbf{o}_k &= 0, \quad \text{for } i \in \{1,\dots,m\} \text{ and } j < k \\ \mathbf{o}_k P_i \mathbf{o}_k &= 0, \quad \text{for } i \in \{1,\dots,m\}. \end{aligned}$$

4: Construct the space $\boldsymbol{O}_k = \boldsymbol{O}_{k-1} \oplus \mathit{Span}\{\mathbf{o}_k\}$
5: **end for**
6: **return** $\boldsymbol{O} = \boldsymbol{O}_m$

Note that the form of the oil vectors $\mathbf{o}_k$ is chosen to assure they are all independent, i.e., the algorithm finds a basis of the oil space. Other forms are possible, and they are all equivalent up to some column permutation. Actually, there is a small probability that a chosen form can't be the basis of the oil space, in case of which we choose randomly another form, i.e., we perform a randomization of the coordinates.

At the k-th iteration, the algorithm solves a system of m quadratic and $(k-1)m$ linear equations in v variables. This means that in the first iteration, there are no available linear relations, so finding the first oil vector is computationally the dominating step. At each subsequent step, we have m additional linear relations that basically reduce the problem to solving a quadratic system of m less variables. As soon as the algorithm reaches $k > v/m + 1$, there are enough linear equations to solve the system, and finding the rest of the oil vectors becomes easy.

Remark 3. The given description of the reconciliation attack from Algorithm 5 is very similar to the recent description given by Beullens [Beu21].

3 Strategy for a Complete Secret Key Recovery

In this section, we summarize the attack strategy. First, we highlight a correspondence between private and public keys in UOV, invoked by the design choices of the scheme. This correspondence partially reveals the input of the subroutine that is the target of our side-channel attack. We explain what information we get from the power measurements and how to exploit them to recover a vector from the secret oil space. Finally, we add a brief complexity analysis of the reconciliation attack with one (or two) known oil vectors, which is used to obtain the remaining oil space and therefore, the complete secret key.

3.1 Overlap in Public and Private Key

One possible way to generate a valid UOV key pair is given in Algorithm 1. This is what we refer to as 'Standard UOV' in the following, as there are no compression techniques applied and the secret key can be used for signing right away. It is also possible to not store large parts of the coefficients of the matrices and instead either expand them from a seed or calculate back and forth between them via Equation (2). Thereby, the key sizes can be reduced massively, at the expense of signing and verification time. This will be noted as 'Compressed UOV'. We will now show, that due to the relation given by Equation (2), in both cases[1], the entries of the sub-matrices $F_1^{(i)}$ of Equation (7) are obvious to any person with access to the public key.

Standard UOV First, the secret key $sk = (T, F^{(1)}, \ldots, F^{(m)})$ is randomly generated. The matrices are of the block-matrix structure given in Equations (7) and (3). Now, the public key $pk = (P^{(1)}, \ldots, P^{(m)})$ is computed by evaluating $P^{(i)} = T^\top F^{(i)} T$ and bringing the resulting matrices to upper triangular form. Since the blocks $F_1^{(i)}$ are already upper triangular matrices, this operation has no impact on them. From

$$\begin{pmatrix} I & 0 \\ T_1^\top & I \end{pmatrix} \begin{pmatrix} F_1^{(i)} & F_2^{(i)} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} I & T_1 \\ 0 & I \end{pmatrix} = \begin{pmatrix} F_1^{(i)} & F_1^{(i)} T_1 + F_2^{(i)} \\ T_1^\top F_1^{(i)} & T_1^\top F_1^{(i)} T_1 + T_1^\top F_2^{(i)} \end{pmatrix}$$

we deduce

$$P^{(i)} = \begin{pmatrix} P_1^{(i)} & P_2^{(i)} \\ 0 & P_3^{(i)} \end{pmatrix} = \begin{pmatrix} F_1^{(i)} & (F_1^{(i)} + F_1^{\top(i)}) T_1 + F_2^{(i)} \\ 0 & \texttt{Upper}(T_1^\top F_1^{(i)} T_1 + T_1^\top F_2^{(i)}) \end{pmatrix}. \tag{8}$$

We notice that the special structure of T leads to $P_1^{(i)} = F_1^{(i)}$, which implies that a considerable amount of the public and private key is identical.

Compressed UOV Here the order is reversed. First, the matrices $P_1^{(i)}$ and $P_2^{(i)}$ are expanded from a random seed pk_{seed}. Then, after T is randomly generated from a secret seed sk_{seed}, the relation shown in Equation (8) is used to compute $F_1^{(i)}$ and $F_2^{(i)}$. Thus, the secret key only consists of two seeds $sk = (pk_{seed}, sk_{seed})$. Finally $P_3^{(i)}$ is computed, again following Equation (8) to complete the key generation. Note, that again $P_1^{(i)} = F_1^{(i)}$ is satisfied. This time $P_1^{(i)}$ is not directly included in the public key, but can be recovered by expanding the public key seed pk_{seed}, which is part of the public key $pk = (pk_{seed}, P_3^{(i)})$.

[1] We want to point out that there are more possible ways to generate valid UOV key pairs. We will bring up one of them in Section 5, where we discuss countermeasures. We focus on these two in the attack description, as they are mainly considered and employed by current implementations.

3.2 Single-Trace Recovery of the Vinegar Variables

The previous findings help us to identify a vulnerability in terms of side-channel resistance in the signing process. Namely, it is the sub-routine responsible for setting up the constant part of the system of linear equations, indicated in Section 2.2. It is given by computing

$$\tilde{\boldsymbol{x}} F_1^{(k)} \tilde{\boldsymbol{x}}^\top = (\tilde{x}_1, \ldots, \tilde{x}_v) \begin{pmatrix} \alpha_{1,1}^{(k)} & \cdots & \alpha_{1,v}^{(k)} \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \alpha_{v,v}^{(k)} \end{pmatrix} (\tilde{x}_1, \ldots, \tilde{x}_v)^\top, \tag{9}$$

using the secret key matrices $F_1^{(k)}$, for all $k \in \{1, \ldots, m\}$. To abbreviate, we will write $\tilde{\boldsymbol{x}} \boldsymbol{F_1} \tilde{\boldsymbol{x}}^\top$ if we want to compute Equation (9) for all $k \in \{1, \ldots, m\}$. Here, the randomly generated vinegar variables $(\tilde{x}_1, \ldots, \tilde{x}_v)$ are multiplied with a considerable amount of known values $\alpha_{i,j}^{(k)}$. In more detail, for every $i \in \{1, \ldots, v\}$ the product

$$\alpha_{i,i}^{(k)} \cdot \tilde{x}_i \tag{10}$$

is computed for all $k \in \{1, \ldots, m\}$. The power consumption of this multiplication depends on the exact value of the respective vinegar variable, which makes them an apparent target for power analysis. In Section 4, we present the details of the suggested side-channel attack. In fact, the attack vector is so strong, that we are able to recover all v vinegar variables from measuring the power consumption of just one signing process with large probability. In the next section, we show that a whole set of vinegar variables, together with the corresponding signature leads to a secret oil vector.

3.3 Obtaining a Secret Oil Vector

As stated in Section 2.2, the vinegar variables $\tilde{\boldsymbol{x}} = (\tilde{x}_1, \ldots, \tilde{x}_v)$ in combination with the secret key generate a linear system of equations. Its solution $\boldsymbol{x'} = (x_{v+1}, \ldots, x_{v+m})$, the so-called oil variables, are concatenated to the vinegar variables. To finalize the signature generation, the resulting vector $(\tilde{\boldsymbol{x}}, \boldsymbol{x'})^\top$ is transformed by T^{-1}, as depicted in Line 9 of Algorithm 2. The result constitutes the signature $(\boldsymbol{s_1}, \boldsymbol{s_2})^\top$. Thus, it holds

$$\begin{pmatrix} \boldsymbol{s_1} \\ \boldsymbol{s_2} \end{pmatrix} = \begin{pmatrix} I & -T_1 \\ 0 & I \end{pmatrix} \begin{pmatrix} \tilde{\boldsymbol{x}} \\ \boldsymbol{x'} \end{pmatrix} = \begin{pmatrix} \tilde{\boldsymbol{x}} - T_1(\boldsymbol{x'}) \\ \boldsymbol{x'} \end{pmatrix}.$$

Obtaining $\tilde{\boldsymbol{x}}$ by a side-channel attack, enables us to choose a vector $\boldsymbol{o} = (T_1(\boldsymbol{x'}), \boldsymbol{x'})^\top$ with the property, that its first $n - m = v$ entries are zero, after it is transformed by T. From the structure of the secret key matrices F_i in Equation (8) one can see that $\mathcal{P}(\boldsymbol{o}) = \mathcal{F} \circ T(\boldsymbol{o}) = 0$. Consequently, we found a vector of the secret oil space $\boldsymbol{o} \in \boldsymbol{O}$. This reduces the complexity of algebraic key recovery attacks significantly.

Remark 4. The previous section basically concludes that if one is in possession of a signature and the corresponding vinegar vector used to build this signature, one is able to determine a secret oil vector by subtracting the two from each other. Using the description given in Section 2.1.2, this is quite obvious, since the signature is of the form $\boldsymbol{s} = \boldsymbol{v} + \boldsymbol{o}$. However, the implementations currently considered follow the description in Section 2.1.1, so we added the former result for clarification.

3.4 Recovering the Secret Oil Space Using (a Combination of) the Kipnis-Shamir Attack and the Reconciliation Attack

As discussed in Section 2.3.2 (also illustrated in [Beu22b, Section 4.1]), when one oil vector $\mathbf{o}_1$ is known in the reconciliation attack, we obtain m linear equations in the entries of a

second $\mathbf{o}_2$. Thus finding a second vector $\mathbf{o}_2 \in O$ can be achieved by solving a quadratic system of m equations in $n-m-m$ variables which can be done using any general system solver, for example the XL algorithm and its variants [CKPS00, Die04, YC05], the (Hybrid) F4/F5 algorithm [Fau99, Fau02, BFP12] or the Joux-Vitse Crossbred algorithm [JV17]. Since $n = 2,5m$, the complexity of the attack is significantly reduced, however, it is still exponential in the number of variables. Instead of using an algebraic solver and the reconciliation attack, we can first use a modification of the Kipnis-Shamir attack to find a second vector $\mathbf{o}_2 \in \boldsymbol{O}$. Alternatively, we can obtain $\mathbf{o}_2$ by an SCA in the same way we recovered $\mathbf{o}_1 \in \boldsymbol{O}$. Note that a second oil vector again provides us with m linear equations and $n < 3m$, so once two oil vectors are known, the remaining oil space can be recovered in polynomial time using the reconciliation attack. We measured the time consumption of this step with the parameters of different security levels and present it in the last column of Table 1. We describe next the procedure for recovering the second oil vector using the Kipnis-Shamir attack.

Using the known oil vector $\mathbf{o}_1$, we form the linear equations $\mathcal{P}'(\mathbf{o}_1, \mathbf{x}) = 0$, i.e.,

$$\mathbf{o}_1 P'_i \mathbf{x} = 0, \quad \text{for } i \in \{1, \ldots, m\}$$

that characterize the space of the remaining oil vectors. We can use these to replace m variables $x_{n-m+1}, \ldots, x_n$ in the equations of the public system $\mathcal{P}(\mathbf{x}) = 0$. Furthermore, for a nonzero coordinate $i \in \{1, \ldots, n-m\}$ of the oil vector $\mathbf{o}_1$, fixing $x_i = 0$ restricts the oil space of the system to $O \setminus \{\mathbf{o}_1\}$. Without loss of generality, let $i = n - m$. We can now apply efficiently the Kipnis-Shamir attack on the obtained system as long as the number of variables $n - m - 1$ is close to two times the dimension of the oil space $dim(O \setminus \{\mathbf{o}_1\}) = m - 1$.

However, the attack can't be directly applied, because the symmetric matrices of the obtained system are not full rank. As a matter of fact, the rank is only $2n-4m$. The reason is that for each P_i, the linear equations above add $m-1$ additional constrains on the oil $\times$ vinegar part, which means the kernel of the matrices is at least $2m-1-(n-m) = 3m-n-1$. This situations can be remedied, by fixing additional $3m - n - 1$ variables, but this will also reduce the dimension of the oil space by $3m - n - 1$. In total, we obtain a system

$$\mathcal{M}_i(x_1, \ldots, x_{2n-4m}) = 0, \ i \in \{1, \ldots, m\}$$

whose oil space $O_{\mathcal{M}}$ is of dimension $n - 2m$, i.e., we obtain a balanced oil and vinegar instance, for which the Kipnis-Shamir attack works best.

As said above, we only need to find one additional oil vector, because then the reconciliation attack can very efficiently recover the rest of the oil space O. Thus, the generalization of Kipnis-Shamir attack from [KPG99] for finding a small invariant subspace of $M'^{-1}_j M'_i$ works best, and we expect to find an oil vector in approximately q trials.

We have implemented and verified both parts of the algebraic attack for all NIST security levels. The results are summarized in Table 1.

Table 1: Practical experiments on different parameter sets when only one oil vector is available. We show the time for finding a second oil vector with the KS attack, and the time for finding $m-2$ more basis vectors of O with the reconciliation attack.

Security level	Kipnis-Shamir step in *hh:mm:ss*	Reconciliation step in *hh:mm:ss*
Reduced parameters $(v, m) = (42, 28)$	00:00:25	00:01:07
Security level I $(v, m) = (68, 44)$	00:08:00	00:11:34
Security level II $(v, m) = (112, 72)$	00:44:36	02:23:19
Security level III $(v, m) = (148, 96)$	00:58:16	10:42:51

4 Executing the Side-Channel Attack

For the side-channel attack presented here, we exploit the data-dependent power consumption of the subroutine discussed in Section 3.2 to compute $\tilde{\boldsymbol{x}}\boldsymbol{F_1}\tilde{\boldsymbol{x}}^\top$. In [PSKH18, PSN21], the authors have already published side-channel attacks on UOV and Rainbow based on correlation power analysis (CPA). However, in all these attacks, the attacker model assumes some kind of public data under control, in order to perform a classic CPA with different messages, i.e., the attack needs control over the digest of the salted and hashed message $\mathcal{H}(\mathcal{H}(d)||salt)$. Furthermore, their attacks require a considerable amount of power traces, taken from the target device. As described in [PSKH18, Section 4.1] around 30 traces are needed to recover elements of the secret linear transformation. Moreover, they are attacking a generic matrix-vector multiplication deviating from the specific implementation that is used in recent UOV implementations.

Our proposed side-channel attack does not require control over the message d during the attack phase to extract vinegar variables. Instead of conducting a CPA, we perform a template-based profiling attack to extract individual bits of the processed vinegar variables while performing the operation $\tilde{\boldsymbol{x}}\boldsymbol{F_1}\tilde{\boldsymbol{x}}^\top$. Therefore, we need a learning phase with access to an identical device running the same implementation of UOV that we are targeting. We also investigate a transferable SCA scenario where the training device in the template building phase is not identical to the target device in the attack phase. We used two STM32F3 core-based platforms, and were also able to extract the secret $\tilde{\boldsymbol{x}}$ in the attack phase successfully. In the learning phase, we recorded reference traces for every possible value in $\mathbb{F}_q$, i.e., in our case $q = 256$. We fix every entry in $\boldsymbol{c} \in \mathbb{F}_q^v$ to the same element in $\mathbb{F}_q$ and execute the operation $\boldsymbol{c}\boldsymbol{F_1}\boldsymbol{c}^\top$ a single time. Subsequently, for every entry $i \in \{1, \ldots, v\}$, we cut out the region of the trace where the entry $c_i \in \boldsymbol{c}$ is processed, so that it can be compared to the corresponding region of $x_i \in \tilde{\boldsymbol{x}}$ that is measured in the attack phase. Because of the dependencies between the private and public keys discussed in Section 3.1, we know the α-values of the secret key of the signature operation we are targeting by considering the available public key. Hence, we can create a template for the multiplication operation in Equation (10) with all the necessary knowledge.

In detail, we exploit execution patterns in the power consumption caused by a bit-value dependent execution within the bit-sliced implementation of the targeted operation. In Section 4.2, we discuss the side-channel exploitation of the implementation in detail.

4.1 Practical Setup

All practical experiments were implemented using the ChipWhisperer tool chain [Tec23] (version 5.6.1) in Python (version 3.8.10) and performed on a ChipWhisperer-Lite board with the CW308T UFO board. The victim board, containing a 32-bit STM32F303RCT7 microcontroller with ARM Cortex-M4 architecture, is mounted on the UFO board. The ARM Cortex-M4 implementation was compiled with `arm-none-eabi-gcc` (version 10.1.0) and the C reference implementation for x86 compiled with `gcc` (version 10.1.0). For running the experiments on our setup, slight modifications of the available implementations [UOV23] were required. The side-channel exploited sections of the code remain unchanged.

Due to the SRAM size of 40 KB on the target STM32F3 core, we specify a reduced parameter set aimed at adapting the attacked routine to the limitations of the target device. For this, we chose the parameters $(v, m) = (42, 28)$ as shown in Table 2 resulting in ≈ 25 KB of used memory for the matrices in $\boldsymbol{F_1}$. Further, our ARM implementation only includes the required functions from Section 3.2 for generating traces. Thus, the execution time for a single run is reduced and more experiments can be conducted within a certain time frame. We note that by adapting the parameter set, our implementation can be used to attack other instantiations, e.g., higher security levels (see Table 2) on suitable target boards. Every coefficient is a field element in $\mathbb{F}_q$ and thus, consumes one byte if $q = 256$.

Table 2: Required memory size for computing $\tilde{\boldsymbol{x}}\boldsymbol{F_1}\tilde{\boldsymbol{x}}^\top$ for different security levels.

Security level	Number of coefficients in $\boldsymbol{F_1}$ $m \cdot v \cdot (v+1)/2$
Reduced parameters $(v, m) = (42, 28)$	25.284
Security level I $(v, m) = (68, 44)$	103.224
Security level II $(v, m) = (112, 72)$	455.616
Security level III $(v, m) = (148, 96)$	1.058.496

Additionally, we adapted the C reference implementation for the x86 architecture from [UOV23] in order to output the public key and signature in a file, since we need them for the attack later on. Furthermore, we output the part of the public key, that contains the coefficients of $\boldsymbol{F_1}$, and the used vinegar variables $\tilde{\boldsymbol{x}}$ in another file. Note, that the latter are used solely to execute the function from Equation (9) on the target device. They are not used in the analysis of our measurements, nor in the subsequent algebraic attack, since they are not known to an attacker. The reference traces are recorded on a profiling device identical to the target device. Hereby, the used vinegar variables are fixed to a certain value $c \in \mathbb{F}_q$ while recording the corresponding trace.

4.2 Exploitable Side-Channel Information

The vinegar variables $\tilde{\boldsymbol{x}}$ are passed via the function parameter `uint8_t b` to the function `gf256v_mul_u32(uint32_t a, uint8_t b)` and get internally processed bitwise. This function performs the basic multiplication operation stated in Equation (10). This operation is required during the computation of $\tilde{\boldsymbol{x}}\boldsymbol{F_1}\tilde{\boldsymbol{x}}^\top$ to set up the constant part of the generated linear system during signing. Then, depending on the value of each of the 8 bits of $\tilde{x}_i$ (resp. parameter `uint8_t b`), a multiplication will be executed, see Line 2 in Algorithm 6 or, for more details, Line 5, 10, 15, 20, 25, 30, 35, 40 in Listing 1 in the appendix. As shown in Figure 1, the spots for these multiplications are easily recognizable for every single bit within the captured power traces.

Algorithm 6 Algorithmic representation of `gf256v_mul_u32`$(\alpha^k_{i,i}, \tilde{x}_i)$ from [UOV23]

Input: $\alpha^k_{i,i}, \tilde{x}_i$
Output: $\alpha^k_{i,i} \cdot \tilde{x}_i$
1: **for** $z_0 = 0$ `to` 7 **do**
2: temp = temp **xor** $\alpha^k_{i,i} \cdot ((\ \tilde{x}_i \ggg z_0)$ **and** $1)$
3: α_msb $= \alpha^k_{i,i}$ **and** 0x80808080
4: $\alpha^k_{i,i} = \alpha^k_{i,i}$ **xor** α_msb
5: $\alpha^k_{i,i} = (\alpha^k_{i,i} \lll 1\ )$ **xor** $((\ \alpha^k_{i,i}$_msb $\ggg 7\ ) \cdot \tilde{x}_i)$
6: **end for**
7: **return** temp

In the learning phase of our attack, we collect and store reference power traces for all 256 possible values of `uint8_t b`, i.e., for the values $\tilde{x}_i$. The collected traces represent the average over $m/4$ runs (since four α-values are processed simultaneously) and include 1000 analog digital converter (ADC) samples recorded in a single capture during the execution of the function `gf256v_mul_u32`. As shown in Figure 1, this number of ADC samples is sufficient to capture the entire execution of the targeted function. The reference traces must be captured only once for a certain target device and public key, and can be reused for further attacks.

After collecting the reference traces, we can create template traces based on the

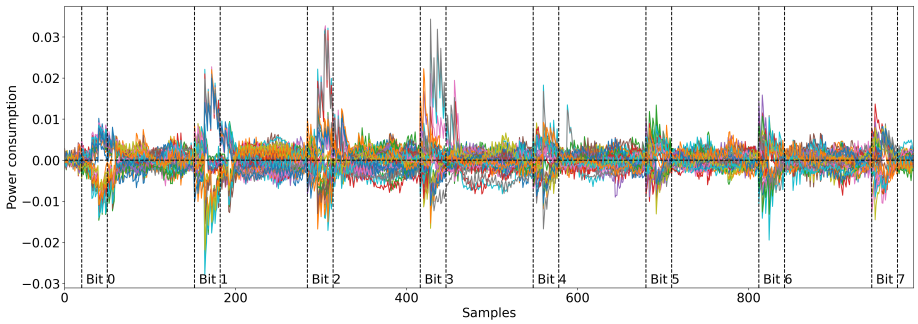


Figure 1: Captured power traces during the execution of the attacked function `gf256v_mul_u32(uint32_t a, uint8_t b)` for random `b` for all 42 vinegar variables.

reference traces by extracting and concatenating the points of interest. Thus, the template trace consists only of the samples marked in Figure 1. As can be seen from Figure 1, we obtain for each of the possible 256 vinegar candidates one unique sequence of value per reference trace based on each processed bits. With this construction of a template, we can perform a correlation analysis that follows more the idea of a horizontal attack as known from CPA attacks against elliptic-curve cryptography (ECC), cf. [NC17].

In the leak extraction phase, usually referred to as the attack phase in the context of a side-channel template attack, we use Pearson correlation [BCO04] to compare all generated reference traces with the acquisition power consumption of a signature creation operation. Of course, we also need to prepare the recorded power traces similarly to the reference traces and extract only the time points of interest of the captured trace for each processing of vinegar variable within the operation of $(\alpha_{i,i}^{(k)} \cdot \tilde{x}_i)$. As shown in Figure 2, the correct candidate can be extracted by calculating the Pearson correlation coefficient between the reference trace and the captured power trace section where the vinegar variable under investigation is processed.

As in general for single trace attacks the extractable information are quite limited. Thus, the correct candidate might not be clearly identifiable due to the signal-to-noise ratio, since there might be one or more field elements that have corresponding power traces with correlation coefficient similar high to the maximum. There is of course no way to verify the correctness of the vinegar variables guess individually, but as described in Section 3.3 there is an efficient way to check if all vinegar variables are correct. We subtract them from the signature and test if the corresponding oil vector candidate is annihilated by $\mathcal{P}$. If this is not the case, we apply a small trial-and-error replacement, where we successively substitute vinegar variables that have a few reference traces with similar high correlation. Following this strategy, we managed to find a valid oil vector, even if our initial guess that uses the values with maximum correlation coefficients was not correct. We implemented the trial-and-error replacement such that it aborts after a few seconds, since we wanted to keep the analysis fast and it was enough in most of the cases. If we have not found an oil vector by then, the attack is considered as unsuccessful, which happened in around 2% of the cases.

4.3 Generalizing the Attack

The achieved results depend strongly on the analyzed implementation, which is shown in Listing 1. The given algorithm processes the targeted vinegar variables bitwise, which implies that the power consumption is directly related to the single bits of the value we want to recover. In the following, we assume that there is a protected implementation in place, that disallows us to draw conclusions about the single bits or even the Hamming

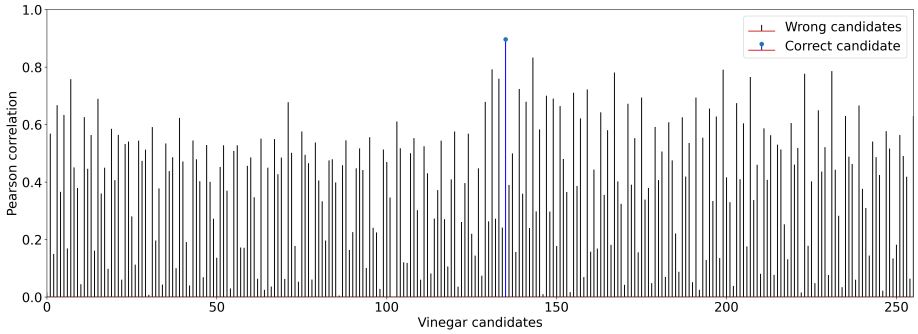


Figure 2: Recovering the value of one vinegar variable by correlation value.

weight (HW) of the vinegar variables $\tilde{x}_i$ itself. We will now discuss theoretically a more relaxed attack model in which we can only obtain information about the Hamming weight of the products $\alpha_{i,i}^{(k)} \cdot \tilde{x}_i$ via side channels. We prove that this would still allow us to recover the vinegar variables and perform the remaining algebraic attack as described in Section 3, up to a certain noise level.

Focus on a specific entry $\tilde{x}_i$, assuming we have m measurements $w^{(k)} = \mathrm{HW}(\alpha_{i,i}^{k} \cdot \tilde{x}_i)$ and knowledge of the respective values $\alpha_{i,i}^{k}$ for $k = 1, \ldots, m$. Then, this entry can be recovered very precisely, since for every k, there is only a small number of field elements $c \in \mathbb{F}_q$, such that $\alpha_{i,i}^{k} \cdot c$ has the measured Hamming weight $w^{(k)}$. A natural method to recover the right value of $\tilde{x}_i$, is to go through $k = 1, \ldots, m$ and count for every $c \in \mathbb{F}_q$ the number of times the Hamming weight of the product $\alpha_{i,i}^{k} \cdot c$ does not coincide with the measured Hamming weight. The element with the smallest number of misses is likely to be the $\tilde{x}_i$ we are looking for.

We have carried out several simulations to test at which noise levels the described method has a high probability of success. To simulate a certain noise level, we took the correct Hamming weights and added an error value $\epsilon^{(k)}$ following a normal distribution with deviation σ. Consequently, adding error values following such a distribution with, e.g., $\sigma = 1$, implies around 68.3% of the conducted Hamming weight measurements $w^{(k)}$ are correct and the others are faulty. Table 3 summarizes the results for various noise levels and the parameter sets of three different security levels[2]. Here, p_i states the probability for correctly obtaining one vinegar variable $\tilde{x}_i$ and $\boldsymbol{p}$ states the probability for a successful recovery of the complete vinegar vector $\tilde{\boldsymbol{x}}$.

Table 3: Success probability of recovering the right vinegar variable(s) at different noise levels of the generalized Hamming weight attack.

Noise level		Security level I $(v, m) = (68, 44)$		Security level II $(v, m) = (112, 72)$		Security level III $(v, m) = (148, 96)$	
dev	acc in %	p_i	$\boldsymbol{p}$	p_i	$\boldsymbol{p}$	p_i	$\boldsymbol{p}$
$\sigma = 1$	68.3	99.9	$93,3$	> 99.9	> 99.9	> 99.9	> 99.9
$\sigma = 1.3$	55.8	98.8	43.3	99.9	93.3	> 99.9	> 99.9
$\sigma = 1.4$	52.5	97.9	20.0	99.8	83.3	99.9	93.3

There are two main conclusions we can draw from this. First, even if only around two thirds of the conducted Hamming weight measurements $w^{(k)}$ are correct, we can still recover a whole vinegar vector with high probability. This can be transformed to an oil

[2]The script for simulating the noise and determining the success probability of the recovery of the used values can also be found at https://github.com/mstoetti/SCA_assisted_recon_UOV.

vector, that enables an efficient Kipnis-Shamir attack, similar to our strategy in Section 3. Second, this probability even grows for larger parameter sets, since there are more available measurements for each vinegar variable, as m grows.

These are just theoretical bounds stating what noise level would be acceptable to still run the generalized attack on the Hamming weight of the products $\alpha_{i,i}^k \cdot \tilde{x}_i$. The real noise level would depend on the target architecture and on the concrete implementation that is chosen to protect the vinegar variables from our attack in Section 4.2.

5 Countermeasures

In this section, we discuss possible countermeasures to prevent the side-channel attack proposed in this work. We will discuss in total four countermeasures, one of which is specific to the UOV algorithm while the other countermeasures rely on established techniques. The concrete implementation of the countermeasures is not discussed in this section and is reserved for future work.

5.1 UOV-Specific Countermeasure

The attack described in this work relies on the correspondence between the public key and the private key, which results from the choice of the linear transformation T (Section 3.1, Equation (3)). This kind of key generation has been coined as *equivalent keys* and is the usual way of generating UOV keys. Another method exists, however, which is referred to as *random affine T* [PSKH18]. In this method the whole matrix T is generated randomly such that it does not feature three blocks of all zeros or the identity matrix. When T is chosen like this, both the $(\alpha_{i,j}^k)$ and the vinegar variables are secret. Hence, the attacker can observe side-channel leakage of only a multiplication of two unknown factors. In the following, we show how our attack adapts to this case. We show that, since both the vinegar variables $\tilde{x}_i$ and the secret key elements $\alpha_{i,j}^k$ have to be revealed, an attacker needs considerably more side-channel traces and the attack is less noise-resistant. Still, an attacker can reveal secret information.

In our target routine, the first operation performed is $\boldsymbol{F_1} \cdot \tilde{\boldsymbol{x}}^\top$, followed by $\tilde{\boldsymbol{x}} \cdot \boldsymbol{F_1} \cdot \tilde{\boldsymbol{x}}^\top$. The intermediate results of the first matrix vector multiplication are accumulated in the considered implementation:

$$F_1^{(k)} \cdot \tilde{\boldsymbol{x}}^\top = \begin{pmatrix} \alpha_{1,1}^k \tilde{x}_1 & + & \alpha_{1,2}^k \tilde{x}_2 & + & \cdots & + & \alpha_{1,v}^k \tilde{x}_v \\ & & \alpha_{2,2}^k \tilde{x}_2 & + & \cdots & + & \alpha_{2,v}^k \tilde{x}_v \\ & & & \ddots & & + & \vdots \\ & & & & & & \alpha_{v,v}^k \tilde{x}_v \end{pmatrix}.$$

In a second step, $\tilde{\boldsymbol{x}}$ is multiplied to this vector from the left:

$$\tilde{\boldsymbol{x}} \cdot F_1^{(k)} \cdot \tilde{\boldsymbol{x}}^\top = \left(\sum_{j=1}^{v} \alpha_{1,j}^k \tilde{x}_j \right) \tilde{x}_1 + \left(\sum_{j=2}^{v} \alpha_{2,j}^k \tilde{x}_j \right) \tilde{x}_2 + \cdots + \alpha_{v,v}^k \tilde{x}_v \tilde{x}_v.$$

During both computations, we can observe the Hamming weight of all individual summands, i.e., the product of two or three unknown values. This is specific to the implementation we are attacking in this work, which is the latest implementation of UOV [UOV23]. Other implementations might lead to other exploitable information.

The general idea underlying this attack is that the Hamming weight of a product carries information about potential values of the factors. First, we focus on the HW pairs $HW(\tilde{x}_v \cdot \alpha_{v,v}^k)$ and $HW(\tilde{x}_v \cdot \tilde{x}_v \cdot \alpha_{v,v}^k)$,

corresponding to the last vinegar variable $\tilde{x}_v$. Hence, by analyzing these Hamming weights, we learn the HW of a product $(\tilde{x}_v \cdot \tilde{x}_v \cdot \alpha^k_{v,v})$ and also of one of the factors $(\tilde{x}_v \cdot \alpha^k_{v,v})$, which helps us reveal information about the co-factor $\tilde{x}_v$. We emphasize that both multiplications are computed for m different values $\alpha^k_{v,v}$, i.e., all multiplications using the same vinegar vector $\tilde{\boldsymbol{x}}$ are computed with m different secret matrices. This increases the information one gets about the vinegar variables. By passing through all provided $k = 1, \ldots, m$ pairs, we can step by step reduce the number of candidates and eventually reduce the number of candidates to a small number (between 1 and 3) with high probability. Consequently, we can almost uniquely learn the value of the vinegar variable $\tilde{x}_v$.

In the next step, we repeat this analysis for several signatures, which is why more side-channel traces are needed in this scenario. Each new signature uses different vinegar variables, but the same matrices $(\alpha^k_{i,j})$. Hence, we obtain different sets of candidates for the last vinegar variables $\tilde{x}^{(i)}_v$ used to generate the corresponding signature. Together with the knowledge of the weights

$$HW(\tilde{x}^{(i)}_v \cdot \alpha^1_{j,v}), \ldots, HW(\tilde{x}^{(i)}_v \cdot \alpha^m_{j,v})$$

for all i and j, these candidates facilitate the recovery of all $\alpha^1_{j,v}, \alpha^2_{j,v}, \ldots, \alpha^m_{j,v}$, for every j, i.e., the last column of all the matrices $(\alpha^k_{i,j})$ with $k \in \{1, \ldots, m\}$. Assuming that we choose the amount of signatures high enough, which also highly depends on the noisiness of the traces, this will leave us with unique values $\alpha^l_{j,v}$ instead of only a list of candidates. With these exact values we can revisit the list of candidates for $\tilde{x}^{(i)}_v$ and also obtain the exact value for $\tilde{x}^{(i)}_v$ that meets the requirement for the corresponding Hamming weights. We proceed to the previous column, and so on. In general for column $v - s$:

1. Recover first the value of $\tilde{x}^{(i)}_{v-s}$ (almost uniquely) from the knowledge of the Hamming weight of the first summand, the exact knowledge of the remaining summands, and the Hamming weight of the product of $v^{(i)}_{v-s}$ with this sum:

$$\begin{array}{ll}
HW(\tilde{x}^{(i)}_{v-s} \cdot \alpha^1_{v-s,v-s}) + \cdots + \tilde{x}^{(i)}_v \cdot \alpha^1_{v-s,v} & HW(\tilde{x}^{(i)}_{v-s}(\tilde{x}^{(i)}_{v-s} \cdot \alpha^1_{v-s,v-s} + \cdots + \tilde{x}^{(i)}_v \cdot \alpha^1_{v-s,v})) \\
HW(\tilde{x}^{(i)}_{v-s} \cdot \alpha^2_{v-s,v-s}) + \cdots + \tilde{x}^{(i)}_v \cdot \alpha^2_{v-s,v} & HW(\tilde{x}^{(i)}_{v-s}(\tilde{x}^{(i)}_{v-s} \cdot \alpha^2_{v-s,v-s} + \cdots + \tilde{x}^{(i)}_v \cdot \alpha^2_{v-s,v})) \\
\vdots & \vdots \\
HW(\tilde{x}^{(i)}_{v-s} \cdot \alpha^m_{v-s,v-s}) + \cdots + \tilde{x}^{(i)}_v \cdot a^m_{v-s,v} & HW(\tilde{x}^{(i)}_{v-s}(\tilde{x}^{(i)}_{v-s} \cdot \alpha^m_{v-s,v-s} + \cdots + \tilde{x}^{(i)}_v \cdot \alpha^m_{v-s,v})).
\end{array}$$

2. As previously, now recover all $\alpha^1_{j,v-s}, \alpha^2_{j,v-s}, \ldots, \alpha^m_{j,v-s}$, for all j.
3. Now, use the $\alpha^l_{j,v-s}$ to determine $\tilde{x}^{(i)}_{v-s}$ uniquely, for all i.

The previous procedure recovers all vinegar variables and the entire matrices $\boldsymbol{F_1}$. It demands a high number of traces and is very susceptible to noise, but shows that choosing T as affine random T does not completely prevent the attack presented in this work.

5.2 Generic Countermeasures

In contrast to the previous discussed countermeasure that exploits the mathematical structure and properties of UOV the here discussed countermeasure concepts can be applied to various cryptographic schemes. In detail, we will discuss three different possible concepts to prevent our proposed attack. These countermeasure techniques are masking, shuffling and pre-computation. Also we will only focus on prevention of first order side-channel leakage.

Masking is a well-established countermeasure against side-channel attacks and can also be used to prevent our proposed attack. In [PSKH18] the authors already propose a multiplicative masking scheme for the operation of a matrix-vector multiplication. This

proposal can be directly applied to the computation of $\alpha^k_{i,i} \cdot \tilde{x}_i$ to prevent first-order leakage exploitation. The basic idea is to multiply a random non-zero value Ψ to the secret vinegar variable $\hat{x}_i = \tilde{x}_i \cdot \Psi$. Hence, the operation $\hat{y}_i = \alpha^k_{i,i} \cdot \hat{x}_i$ is performed with the randomized vinegar variable $\hat{x}_i$. Afterwards, the original value can be reconstructed by multiplying the inverse of the random value Ψ^{-1} by the multiplication result $y_i = \hat{y}_i \cdot \Psi^{-1}$. In this multiplicative masking setting, an attacker would only obtain the masked vinegar variable $\hat{x}_i$ by applying our template-based CPA to the captured traces. Of course, an attacker could perform our proposed attack twice to first guess the Ψ and then reconstruct $\tilde{x}_i = \hat{x}_i \cdot \Psi^{-1}$. To perform such a second-order template attack, the noise in the power traces must be very low to correctly guess Ψ.

In praxis shuffling is implemented in parallel to masking to decrease the signal-to-noise ration of the captured traces. This noise introduced by shuffling will not cancel the side-channel leakage, but more measurements needs to be conducted to extract the side-channel information. Due to [HOM06] the number of required traces for a successful attack increases quadratically. In case of the presented attack on UOV we can only captured one trace with the same settings of α. Therefore, shuffling can be considered as a sufficient countermeasure against our proposed single trace template attack. To counteract our analysis, shuffling can be applied at two different places. First, shuffling can be applied on the vector-matrix multiplication given in Equation (9). The authors of [PSKH18] already proposed a shuffling scheme on matrix-vector multiplication to prevent their proposed CPA attack. In the attack we introduce, the shuffling approach only needs to be applied to the multiplication involving the diagonal elements of the central map ($\alpha^k_{i,i} \cdot \tilde{x}_i$). Thus, depending on the security parameters of UOV, the number of possible execution sequences corresponds to the factorial of m. Thereby the proposed randomization of the index i does not provide as much permutation as the scheme proposed by [PSKH18], but it requires less randomness to execute the permutation.

Second, execution order of the bit-dependent operation in the bit-sliced implementation of `gf256v_mul_u32` can be shuffled. The proposed shuffle scheme in Algorithm 7 exploits a modulus operation and a random value Φ to change the starting index of the cyclic execution over the 8 bits. Thereby not all potential permutations of the index sequence by !8 are exploited, but it provides a minimal computational overhead. By just applying the alternating starting point of the sequence in Line 2 of Algorithm 7 we have only 8 different sequences.

Algorithm 7 Shuffled conditional move version of Algorithm `gf256v_mul_u32`($\alpha^k_{i,i}, \tilde{x}_i, \Phi$)

Input: $\alpha^k_{i,i}, \tilde{x}_i, \Phi$
Output: $\alpha^k_{i,i} \cdot \tilde{x}_i$

1: **for** $z_0 = 0$ `to` 7 **do**
2: $z_1 = (\ (z_0 + \Phi)$ **mod** 7)
3: **if** (($\tilde{x}_i$ » z_1) **and** 1) **then**
4: tempT = tempT **xor** $\alpha^k_{i,i}$
5: tempF = tempF **xor** 0x00
6: **else**
7: tempF = tempF **xor** $\alpha^k_{i,i}$
8: tempT = tempT **xor** 0x00
9: **end if**
10: $\alpha^k_{i,i}$_msb = $\alpha^k_{i,i}$ **and** 0x80808080
11: $\alpha^k_{i,i}$ ^= $\alpha^k_{i,i}$ **xor** $\alpha^k_{i,i}$_msb
12: $\alpha^k_{i,i} = (\alpha^k_{i,i}$ « 1) **xor** (($\alpha^k_{i,i}$_msb » 7) $\cdot$ $\tilde{x}_i$)
13: **end for**
14: **return** tempT

Hence, an attacker can reconstruct all 8 possible values form the guessed bits. However, with 8 possible candidates per vinegar variable, the complexity increases to a computational complexity of the 8th power to the numbers of entries in $\tilde{\boldsymbol{x}}$, i.e v.
In addition, we adopted the idea of *Always-Double-and-Add*, known as ECC side-channel countermeasure, to hide the conditional execution of the operation in Line 2 of Algorithm 6. In Algorithm 7 the previous conditional operation of temp = temp **xor** $\alpha_{i,i}^k$ in Line 2 of Algorithm 6 is always executed, but conditionally stored in different registers, see Line 3 to 9. Thereby, conditional execution based leakage does not exist anymore. Still this scheme is attackable with an Adress-PDA [IIT02] to distinguish the storage location of the intermediate values, but therefore more than one measurement is required. In general, we propose to apply a combination of the above mentioned countermeasures to prevent our proposed attack on UOV.

Finally, we observe that if the vinegar variables are generated message independent, then their insertion into the central polynomials, i.e., the vulnerable subroutine we identified, might be part of a precomputation step. In case there is an offline phase in place, where message independent operations are precomputed, like suggested by Shim et al. in [SLK22], then this protects also against our proposed attack, since there is no way anymore to obtain necessary side-channel information.

6 Conclusion

In this paper we present a novel side-channel attack against UOV. It exploits leakage, that appears by inserting the vinegar variables into the secret polynomials. This leakage becomes substantial, since we can inherently deduce a large amount of the coefficients of these polynomial. This facilitates a template attack, where we only need a single attack trace to recover a complete set of vinegar variables (resp. an oil vector). We have implemented the attack with the ChipWhisperer Setup and a STM32F3 target board. The success probability thereof, lies above 97% even though we took the reference traces for the template and the attack traces on different devices, i.e., we separated carefully into profiling and target device. From an attacker point of view, it is easy to verify if the side-channel attack was successful, since the retrieved oil vector is annihilated by the public-key map. With the knowledge of one (single-trace) additional oil vector, we can recover the secret key in polynomial time by means of the Kipnis-Shamir attack and the reconciliation attack.

We theoretically extended our approach to a potentially protected implementation that screens the vinegar variables, where we might have only access to the Hamming weights of the products of the vinegar variables and the coefficients of the secret polynomials. We showed that the attack is still feasible and even has a certain noise resistance. Contrary to existing side-channel attacks on UOV, we do not attack the linear transformation. This indicates that our attack is still viable when the implementation is adapted to the description we stated in Section 2.1.1, which omits the usage of the central map and linear transformation. Here, the vinegar variables are inserted directly into the public key map, so we do not need the correspondence derived in Section 3.1.

This also provides ideas for future work, where we want to apply the presented attack to the MAYO signature scheme [Beu22b]. During signing in MAYO, several sets of vinegar variables are inserted into the public key map, that is very similar to the one used in UOV. This indicates the possibility to recover oil vectors in a similar fashion, and with the parameters used in the available public implementation[3], one known oil vector is enough to start a very efficient reconciliation attack. Furthermore, there might be room for improvement regarding our classification technique. We applied a straight forward template attack, where we just clued together the regions of interest and sought for the

[3] https://github.com/WardBeullens/MAYO

trace of the field element that had the highest correlation. We managed to achieve good results on the available implementation, but it might be necessary to apply tools like a machine learning classifier to attack more protected implementations.

Acknowledgement

We want to thank Ward Beullens sincerely for pointing us to the applicability of the Kipnis-Shamir attack in our scenario. In a previous version of this work, we needed two traces to be able to construct a linear system of equations, while a single-trace attack required a more complex analysis phase.
This research work has been funded by the German Ministry of Education, Research and Technology in the context of the project Aquorypt (grant number 16KIS1022).

A Source code of `gf256v_mul_u32`

```
static inline uint32_t gf256v_mul_u32(uint32_t a, uint8_t b) {
    uint32_t a_msb;
    uint32_t a32 = a;
    uint32_t b32 = b;
    uint32_t r32 = a32*(b32&1);  // Exploit Bit 0

    a_msb = a32&0x80808080;
    a32 ^= a_msb;
    a32 = (a32<<1)^((a_msb>>7)*0x1b);
    r32 ^= (a32)*((b32>>1)&1);  // Exploit Bit 1

    a_msb = a32&0x80808080;
    a32 ^= a_msb;
    a32 = (a32<<1)^((a_msb>>7)*0x1b);
    r32 ^= (a32)*((b32>>2)&1);  // Exploit Bit 2

    a_msb = a32&0x80808080;
    a32 ^= a_msb;
    a32 = (a32<<1)^((a_msb>>7)*0x1b);
    r32 ^= (a32)*((b32>>3)&1);  // Exploit Bit 3

    a_msb = a32&0x80808080;
    a32 ^= a_msb;
    a32 = (a32<<1)^((a_msb>>7)*0x1b);
    r32 ^= (a32)*((b32>>4)&1);  // Exploit Bit 4

    a_msb = a32&0x80808080;
    a32 ^= a_msb;
    a32 = (a32<<1)^((a_msb>>7)*0x1b);
    r32 ^= (a32)*((b32>>5)&1);  // Exploit Bit 5

    a_msb = a32&0x80808080;
    a32 ^= a_msb;
    a32 = (a32<<1)^((a_msb>>7)*0x1b);
    r32 ^= (a32)*((b32>>6)&1);  // Exploit Bit 6

    a_msb = a32&0x80808080;
    a32 ^= a_msb;
    a32 = (a32<<1)^((a_msb>>7)*0x1b);
    r32 ^= (a32)*((b32>>7)&1);  // Exploit Bit 7

    return r32;
}
```

Listing 1: Vulnerable bit-sliced multiplication operation used for the calculation of $(\alpha_{i,i}^k \cdot \tilde{x}_i)$ in Equation (1) from the available implementation [UOV23].

References

[AKKM20] Thomas Aulbach, Tobias Kovats, Juliane Krämer, and Soundes Marzougui. Recovering Rainbow's Secret Key with a First-Order Fault Attack. In *Progress in Cryptology - AFRICACRYPT 2020 - 12th International Conference on Cryptology in Africa, Cairo, Egypt, July 20-22, 2020, Proceedings*, volume 12174 of *Lecture Notes in Computer Science*. Springer, 2020.

[BC14] Christina Boura and Anne Canteaut. A new criterion for avoiding the propagation of linear relations through an Sbox. In *FSE 2013 - Fast Software Encryption*, LNCS, Singapore, 2014. Springer.

[BCH+23] Ward Beullens, Ming-Shing Chen, Shih-Hao Hung, Matthias J. Kannwischer, Bo-Yuan Peng, Cheng-Jhih Shih, and Bo-Yin Yang. Oil and Vinegar: Modern Parameters and Implementations. Cryptology ePrint Archive, Report 2023/059, 2023. https://ia.cr/2023/059, accepted for publication at TCHES'23.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.

[Beu21] Ward Beullens. Improved cryptanalysis of UOV and Rainbow. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 348–373. Springer, 2021.

[Beu22a] Ward Beullens. Breaking Rainbow Takes a Weekend on a Laptop. In *CRYPTO*, volume 13508 of *Lecture Notes in Computer Science*, pages 464–479. Springer, 2022.

[Beu22b] Ward Beullens. Mayo: Practical post-quantum signatures from oil-and-vinegar maps. In *International Conference on Selected Areas in Cryptography*, pages 355–376. Springer, 2022.

[BFP12] Luk Bettale, Jean-Charles Faugère, and Ludovic Perret. Solving polynomial systems over finite fields: improved analysis of the hybrid approach. In Joris van der Hoeven and Mark van Hoeij, editors, *Proceedings of the 37th International Symposium on Symbolic and Algebraic Computation – ISSAC '12*, pages 67–74. ACM, 2012. https://hal.inria.fr/hal-00776070/document.

[BHK+19] Daniel J. Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. The SPHINCS$^+$ Signature Framework. In *CCS*, pages 2129–2146. ACM, 2019.

[BPSV19] Ward Beullens, Bart Preneel, Alan Szepieniec, and Frederik Vercauteren. LUOV. Technical report, National Institute of Standards and Technology, 2019. available at https://github.com/WardBeullens/LUOV/blob/master/Supporting_Documentation/luov.pdf.

[CFM+20] A. Casanova, J.-C. Faugère, G. Macario-Rat, J. Patarin, L. Perret, and J. Ryckeghem. GeMSS. Technical report, National Institute of Standards and Technology, 2020.

[CKPS00] Nicolas Courtois, Er Klimov, Jacques Patarin, and Adi Shamir. Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807, pages 392–407, 2000. www.iacr.org/archive/eurocrypt2000/1807/18070398-new.pdf.

[DCP+20] Jintai Ding, Ming-Shing Chen, Albrecht Petzoldt, Dieter Schmidt, Bo-Yin Yang, Matthias Kannwischer, and Jacques Patarin. Rainbow. Technical report, National Institute of Standards and Technology, 2020.

[Die04] Claus Diem. The XL-algorithm and a conjecture from commutative algebra. In *Advances in Cryptology – ASIACRYPT 2004*, volume 3329, pages 323–337, 2004. https://www.iacr.org/archive/asiacrypt2004/33290320/33290320.pdf.

[DKL+18] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):238–268, 2018.

[DYC+08] Jintai Ding, Bo-Yin Yang, Chia-Hsin Owen Chen, Ming-Shing Chen, and Chen-Mou Cheng. New Differential-Algebraic Attacks and Reparametrization of Rainbow. In *ACNS*, volume 5037 of *LNCS*, pages 242–257, 2008.

[Fau99] Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases (F4). *Journal of Pure and Applied Algebra*, 139:61–88, 1999. http://www-polsys.lip6.fr/~jcf/Papers/F99a.pdf.

[Fau02] Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation – ISSAC '02*, pages 75–83. ACM, 2002. http://www-polsys.lip6.fr/~jcf/Papers/F02a.pdf.

[FKNT22] Hiroki Furue, Yutaro Kiyomura, Tatsuya Nagasawa, and Tsuyoshi Takagi. A New Fault Attack on UOV Multivariate Signature Scheme. In *PQCrypto*, volume 13512 of *Lecture Notes in Computer Science*, pages 124–143. Springer, 2022.

[HOM06] Christoph Herbst, Elisabeth Oswald, and Stefan Mangard. An AES smart card implementation resistant to power analysis attacks. In Jianying Zhou, Moti Yung, and Feng Bao, editors, *Applied Cryptography and Network Security, 4th International Conference, ACNS 2006, Singapore, June 6-9, 2006, Proceedings*, volume 3989 of *Lecture Notes in Computer Science*, pages 239–252, 2006.

[HTS11] Yasufumi Hashimoto, Tsuyoshi Takagi, and Kouichi Sakurai. General fault attacks on multivariate public key cryptosystems. In *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29 - December 2, 2011. Proceedings*, pages 1–18, 2011.

[IIT02] Kouichi Itoh, Tetsuya Izu, and Masahiko Takenaka. Address-bit differential power analysis of cryptographic schemes OK-ECDH and OK-ECDSA. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 129–143. Springer, 2002.

[JV17] Antoine Joux and Vanessa Vitse. A Crossbred Algorithm for Solving Boolean Polynomial Systems. In Jerzy Kaczorowski, Josef Pieprzyk, and Jacek Pomykala, editors, *Number-Theoretic Methods in Cryptology - First International Conference, NuTMiC 2017, Warsaw, Poland, September 11-13, 2017, Revised Selected Papers*, volume 10737 of *Lecture Notes in Computer Science*, pages 3–21. Springer, 2017.

[KL19] Juliane Krämer and Mirjam Loiero. Fault Attacks on UOV and Rainbow. In *COSADE*, volume 11421 of *Lecture Notes in Computer Science*, pages 193–214. Springer, 2019.

[KPG99] Aviad Kipnis, Jacques Patarin, and Louis Goubin. Unbalanced Oil and Vinegar Signature Schemes. In Jacques Stern, editor, *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 206–222. Springer, 1999.

[KS06] Aviad Kipnis and Adi Shamir. Cryptanalysis of the oil and vinegar signature scheme. In *Advances in Cryptology—CRYPTO'98: 18th Annual International Cryptology Conference Santa Barbara, California, USA August 23–27, 1998 Proceedings*, pages 257–266. Springer, 2006.

[MIS20] Koksal Mus, Saad Islam, and Berk Sunar. QuantumHammer: a Practical Hybrid Attack on the LUOV Signature Scheme. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1071–1084, 2020.

[NC17] Erick Nascimento and Lukasz Chmielewski. Horizontal Clustering Side-Channel Attacks on Embedded ECC Implementations (Extended Version). *IACR Cryptol. ePrint Arch.*, page 1204, 2017.

[PFH+22] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. FALCON. Technical report, National Institute of Standards and Technology, 2022.

[PSKH18] Aesun Park, Kyung-Ah Shim, Namhun Koo, and Dong-Guk Han. Side-channel attacks on Post-Quantum Signature Schemes based on multivariate quadratic equations:-Rainbow and UOV. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 500–523, 2018.

[PSN21] David Pokorný, Petr Socha, and Martin Novotný. Side-Channel Attack on Rainbow Post-Quantum Signature. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 565–568. IEEE, 2021.

[SG14] Simona Samardjiska and Danilo Gligoroski. Linearity Measures for Multivariate Public Key Cryptography. In *SECURWARE 2014, The Eighth International Conference on Emerging Security Information, Systems and Technologies*, pages 157–166, 2014.

[SLK22] Kyung-Ah Shim, Sangyub Lee, and Namhun Koo. Efficient Implementations of Rainbow and UOV using AVX2. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 245–269, 2022.

[Tec23] NewAE Technology. Repository of chipwhisperer tool chain - commit 6bf3bac, 2023. https://github.com/newaetech/chipwhisperer.

[TPD21] Chengdong Tao, Albrecht Petzoldt, and Jintai Ding. Efficient Key Recovery for All HFE Signature Variants. In *CRYPTO (1)*, volume 12825 of *Lecture Notes in Computer Science*, pages 70–93. Springer, 2021.

[UOV23] Repository of Oil and Vinegar: Modern Parameters and Implementations - commit eeedc68, 2023. https://github.com/pqov/pqov-paper.

[VP20] Ricardo Villanueva-Polanco. Cold Boot Attacks on LUOV. *Applied Sciences*, 10:4106, 06 2020.

[YC05] Bo-Yin Yang and Jiun-Ming Chen. All in the XL family: Theory and practice. In Choon sik Park and Seongtaek Chee, editors, *Information Security and Cryptology – ICISC 2004*, pages 67–86, 2005. http://by.iis.sinica.edu.tw/by-publ/recent/xxl.pdf.

MAYo or MAY-not: Exploring Implementation Security of the Post-Quantum Signature Scheme MAYO Against Physical Attacks

Thomas Aulbach*
University of Regensburg, Germany
thomas.aulbach@ur.de

Soundes Marzougui*
STMicroelectronics, Belgium
soundes.marzougui@st.com

Jean-Pierre Seifert*
TU Berlin – SECT, Germany
Fraunhofer SIT, Darmstadt, Germany
jean-pierre.seifert@tu-berlin.de

Vincent Quentin Ulitzsch*
TU Berlin – SECT, Germany
vincent@sect.tu-berlin.de

2024 Workshop on Fault Detection and Tolerance in Cryptography (FDTC) | 979-8-3503-8036-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/FDTC64268.2024.00012

***Abstract*—MAYO is a multivariate signature scheme notable for its efficiency and compact key size. Targeting NIST security level I, MAYO features a public key size of 1168 bytes and a signature size of 321 bytes, making it more compact than leading lattice-based signature schemes like Falcon and Dilithium, thereby easing integration into embedded systems. With the deployment of MAYO in embedded systems, studying the resilience of MAYO implementations against fault injection attacks is of increasing importance. In this paper, we investigate the security of MAYO against fault injection attacks, and present the first end-to-end fault injection attack on the multivariate scheme. The attack introduces a loop-abort fault in the sampling of the vinegar vector. We present two variants: A zero-ing attack, in which the skipped sampling results in an all-zero vinegar vector, and a differential fault attack. In both variants, the faulted signature reveals an oil vector, allowing for full key recovery through techniques borrowed from the reconciliation attack in a few seconds.**

***Index Terms*—fault-injection, MAYO, multivariate schemes, post-quantum cryptography**

I. Introduction

Quantum attacks can compromise the computational problems foundational to classical cryptography, such as factorization and discrete logarithms. This underscores the urgent need to develop cryptographic methods resilient to quantum attacks. In response, researchers have proposed new mathematical assumptions and computational problems that remain intractable for quantum computers, thereby giving rise to the field of post-quantum cryptography. These new assumptions are grouped into different families, such as lattice-based, code-based, hash-based, and multivariate cryptography. Multivariate cryptography schemes, referred to as MQ schemes, primarily depend on the computational difficulty of solving large systems of multivariate quadratic equations. Since the first MQ encryption scheme was introduced by Imai and Matsumoto, most of these schemes have been broken [Beu22], [Pat00], [KS98], with the exception of a few signature schemes like Unbalanced Oil-and-Vinegar (UOV) variants and HFEv-variants. MQ schemes are build on relatively simple operations, relying mostly on vector-matrix operations and solving linear systems over small finite fields without the need for multi-precision arithmetic. They are also considered to be highly efficient, especially for resource-constrained devices with modest computational power, and do not require a cryptographic coprocessor. Additionally, the size of the signatures of MQ schemes is among the shortest, among all known post-quantum signature schemes. However, the major disadvantage of multivariate schemes was recognized to be in the size of the keys. Multivariate schemes suffer from relatively large key sizes (in the order of hundreds of kilobytes). As a consequence, the integration of multivariate schemes was limited to devices with sufficient memory capabilities such as smartphones.

Rainbow [DS05] and LUOV [BPSV19] were multivariate schemes present in the first rounds of the NIST standardization process. The LUOV scheme was already excluded from the NIST competition, despite it being simpler and having a smaller attack surface when compared to Rainbow. On the other hand, in 2022, Rainbow was broken by Beullens [Beu22]. To defend against this attack, one solution is to increase the parameters used in the Rainbow scheme, but this results in larger key and signature sizes.

The NIST standardization competition is still ongoing and continuously evolving, with new submissions and emerging attacks regularly influencing the landscape. Notably, MAYO was submitted in response to a call for more diverse cryptographic approaches to signature schemes. MAYO is a post-quantum signature scheme submitted by Beullens. According to [Beu21b], MAYO is considered to be the most efficient multivariate scheme due to its small public key size compared to other multivariate schemes, such as Rainbow and UOV.

* Authors are listed in alphabetical order.

The authors acknowledge the financial support by the Federal Ministry of Education and Research of Germany in the programme of "Souverän. Digital. Vernetzt." Joint project 6G-RIC, project identification number: 16KISK030. Furthermore, this work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - project number 50550035.

2995-0252/24/$31.00 ©2024 IEEE
DOI 10.1109/FDTC64268.2024.00012

With parameters targeting NISTPQC security level I, the public key size of MAYO is 1168 bytes, and the signature size is 321 bytes, making it even more compact than other state-of-the-art lattice-based signature schemes such as Falcon and Dilithium.

With the introduction of MAYO, current efforts have been directed towards optimizing its implementation, as seen in studies like [HSMR23], [SMA+24]. Given these optimizations, MAYO is likely to see real-world deployment on embedded devices. This introduces a new threat-landscape; cryptographic schemes deployed on embedded devices face the threat of physical side-channel and fault injection attacks. It is therefore important to safeguard the implementations against these attacks. The authors of the MAYO scheme acknowledge this threat in their NIST submission end emphasize that the use of a salt in their implementation reduces the attack surface against active physical attacks [BCC+23].

To understand how MAYO can be secured against physical attackers, it is important to study a physical attacker's capabilities and points of attack in MAYO implementations. However, so far, there is little research of end-to-end attack on multivariate schemes. Existing research focuses on theoretical or emulated attacks, such as [KL19] and [FKNT22]. Historically, multivariate schemes have been hindered by large key sizes, making their deployment in embedded devices challenging. For instance, deploying Rainbow on a Cortex-M4 target board on a ChipWhisperer was not feasible. This limitation has likely contributed to the mentioned absence of end-to-end attacks on multivariate schemes in the literature. As of writing of this paper, no end-to-end fault-injection attack has been reported against multivariate schemes in general, and against MAYO specifically.

a) Contribution: In this paper, we present multiple fault injection attacks on the multivariate signature scheme MAYO. We show that by attacking the random sampling of the vinegar values, an attacker can recover MAYO's secret key given only one faulted signature. The key insight is that the attacker can recover a vector from the secret oil space from a a faulted signature and then recover the oil space through techniques borrowed from the reconciliation attack. The techniques presented here are akin to fault injection attacks on Rainbow — as such, they demonstrate a critical section in the MAYO implementation that needs to be protected against fault injection for real-world deployment.

In Section II, we give a background on multivariate schemes. Section III describes the fault injection attacks, following by experimental evaluation of the attacks in Section IV. We conclude the paper with Section V where we discuss the possible countermeasures that can present against the presented attack.

b) Related work: Prior work demonstrated fault injection attacks against multivariate cryptography, but the attack surface of MAYO implementations has been unexplored so far. Fault injection attacks on multivariate public-key crypto systems were first introduced by Hashimoto et. al in [HTS11]. The authors demonstrated how to recover the secret key with a single fault. In [KL19], Krämer and Loiero based on the results of [HTS11] to comprehensively analyse how the attacks can be applied to UOV and Rainbow. First, they analyzed the propagation of a faulted coefficient in the central map through the signature, demonstrating how this can be exploited to gain information about the secret transformation. Second, they examined the impact of fixing the vinegar variables across multiple signatures, revealing how this process can disclose information about the secret transformation. Shim and Koo further developed this latter approach into a full key recovery attack [SK20]. However this method has high computational complexity. In [AKKM22], Aulbach et al. demonstrate how Rainbow can be attacked through fault injection.

II. Background

A. Multivariate Signature Schemes

The core of multivariate signature schemes in general and MAYO signature scheme [Beu21b] specifically is a quadratic map. For example, this map can be defined as the following $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ with m multivariate quadratic polynomials as a sequence $p_1(\mathbf{x}), \ldots, p_m(\mathbf{x})$ in n variables $\mathbf{x} = (x_1, \ldots, x_n)$ and the coefficients of the polynomials are in $\mathbb{F}_q$, a finite field with q elements.

Herein, the hardness of multivariate schemes is based on the hardness of finding a preimage $\mathbf{s} \in \mathbb{F}_q^n$ of a target vector $\mathbf{t} \in \mathbb{F}_q^m$ under a given multivariate quadratic map $\mathcal{P}$, i.e., solving a multivariate system of quadratic equations. This task is often referred to as the MQ problem.

In a multivariate cryptography scheme, the map $\mathcal{P}$ is given as the public key, while the secret key is a trapdoor in the public map $\mathcal{P}$, allowing to invert $\mathcal{P}$. To sign a message, the signer first hashes the message into a target vector $\mathbf{t} \in \mathbb{F}_q^m$, and then finds a pre-image $\mathbf{s}$ of $\mathbf{t}$ under $\mathcal{P}$. The verifier then simply checks that $\mathcal{P}(\mathbf{s}) = \mathbf{t}$.

B. The Trapdoor in UOV

The Mayo signature scheme is a special modification of the UOV signature scheme. The main idea behind oil and vinegar schemes is to introduce a trapdoor into the set of equations to allow efficient sampling of preimages. In UOV, the trapdoor information is a basis of a secret linear subspace $\mathcal{O} \subset \mathbb{F}_q^n$ of dimension $\dim(\mathcal{O}) = m$, the so-called oil space [Beu21a]. This oil space has the property that it vanishes on the public key map, i.e., $\mathcal{P}(\mathbf{o}) = \mathbf{0}_m$ for all $\mathbf{o} \in \mathcal{O}$.

One can define the *polar form* or *differential* of the multivariate quadratic polynomials $p_i(\mathbf{x})$ as

$$p_i'(\mathbf{x}, \mathbf{y}) \coloneqq p_i(\mathbf{x} + \mathbf{y}) - p_i(\mathbf{x}) - p_i(\mathbf{y}) + p_i(\mathbf{0}).$$

Note, that the multivariate quadratic polynomials constitute the map $\mathcal{P}$ via $\mathcal{P}(\mathbf{x}) = p_1(\mathbf{x}), \ldots, p_m(\mathbf{x})$.

Since these quadratic polynomials are homogeneous polynomials, the term $p_i(\mathbf{0})$ will be omitted in the following. The differential or polar form of $\mathcal{P}$ can now be defined as

$$\mathcal{P}'(\mathbf{x}, \mathbf{y}) = p_1'(\mathbf{x}, \mathbf{y}), \ldots, p_m'(\mathbf{x}, \mathbf{y}).$$

The map $\mathcal{P}' : \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q^m$ is a symmetric and bilinear map, see [Beu21a, Theorem 1].
Furthermore, if an attacker has knowledge of the secret oil space, she can easily generate a legitimate signature of a message of her choice. This proceeds by using the oil space to efficiently find preimages $\mathbf{x} \in \mathbb{F}_q^n$ of a given target $\mathbf{t} \in \mathbb{F}_q^m$ (derived from the message to sign) such that $\mathcal{P}(\mathbf{x}) = \mathbf{t}$. To do so, one can randomly pick a vinegar vector $\mathbf{v} \in \mathbb{F}_q^n$ and solve the system $P(\mathbf{v}+\mathbf{o}) = \mathbf{t}$ for $\mathbf{o} \in \mathcal{O}$. This is possible since in

$$\mathbf{t} = \mathcal{P}(\mathbf{v}+\mathbf{o}) = \mathcal{P}(\mathbf{v}) + \mathcal{P}(\mathbf{o}) + \mathcal{P}'(\mathbf{v},\mathbf{o}) \tag{1}$$

the term $\mathcal{P}(\mathbf{v})$ is constant and $\mathcal{P}(\mathbf{o})$ vanishes, so whenever the linear map $\mathcal{P}'(\mathbf{v},\cdot)$ is non-singular, the system has a unique solution $\mathbf{o} \in \mathcal{O}$, which can be computed efficiently. Without the knowledge of the oil space $\mathcal{O}$, the term $\mathcal{P}(\mathbf{o})$ implies that Equation 1 constitutes a system of quadratic equations, which remains hard to solve.

Building a signature scheme directly from this setting has, however, one important disadvantage influencing the performance of the scheme. To be secure against algebraic attacks introduced in the literature, the oil space and the image space of the multivariate quadratic map $\mathcal{P}$ need to be picked carefully and should be equally large, i.e., dim $\mathcal{O} = m$. Moreover, the parameter n representing the variables needs to be sufficiently larger than m, with $n \approx 2,5m$ being used in all currently considered implementations. This is mandatory to counter the Kipnis-Shamir attack [KS98]. Additionally, the parameter m itself needs to be of a certain size as well, to provide security against the intersection attack [Beu21a]. Those conditions lead to key pairs of enormous size, which is considered the main drawback of multivariate signatures. The MAYO scheme aims at tackling this problem.

C. Description of MAYO

The huge size of the key pairs of multivariate schemes is linked to the parameters m and n, representing the number of multivariate quadratic polynomials and variables, respectively. The MAYO scheme relies on the downsizing of the dimension of the oil space to dim $\mathcal{O} = o < m$. However, this oil space is now too small to sample signatures, since the system $\mathcal{P}(\mathbf{v}+\mathbf{o}) = \mathbf{t}$ given in Equation 1 consists consequently of m linear equations in o variables and is unlikely to have any solutions. The approach taken by the authors of MAYO in [Beu21b] is to stretch the public key map into a larger whipped map $\mathcal{P}^* : \mathbb{F}_q^{kn} \to \mathbb{F}_q^m$, such that it accepts k input vectors $\mathbf{x} \in \mathbb{F}_q^n$. This is realized by defining

$$\mathcal{P}^*(\mathbf{x}_1,...,\mathbf{x}_k) := \sum_{i=1}^{k} \mathbf{E}_{ii}\mathcal{P}(\mathbf{x}_i) + \sum_{1\le i<j\le k} \mathbf{E}_{ij}(\mathcal{P}'(\mathbf{x}_i,\mathbf{x}_j)), \tag{2}$$

where the matrices $\mathbf{E}_{ij} \in \mathbb{F}_q^{m\times m}$ represent multiplications by $1, X, X^2, \ldots, X^{\binom{k}{2}-1}$ in $\mathbb{F}_q[X]/f(X)$, for some monic irreducible polynomial $f(X)$ of degree m. This is possible for parameter sets that satisfy $\binom{k}{2} < m$.
Figure 1 gives the pseudocode of the MAYO algorithms, as described in [Beu21b]. We stick to this description, since it contains all the key ideas, but remains very compact at the same time, omitting a lot of details like the de- and encoding of the matrices given in the pseudocode of the actual NIST submission [BCC+23].

The oil and vinegar property holds for MAYO as well. Hence, one can notice that $\mathcal{P}^*$ vanishes on the subspace $\mathcal{O}^k = \{(\mathbf{o}_1,\ldots,\mathbf{o}_k) |$ with $\mathbf{o}_i \in \mathcal{O}$ for all $i \in [k]\}$ of dimension ko. By choosing the parameters such that $ko \ge m$, the k copies of the oil space are large enough to construct preimages of a target vector $\mathbf{t} \in \mathbb{F}_q^m$ under the whipped map $\mathcal{P}^*$. In more detail, the signer randomly samples $(\mathbf{v}_1,\ldots,\mathbf{v}_k) \in \mathbb{F}_q^{kn}$, and then solves

$$\mathcal{P}^*(\mathbf{v}_1+\mathbf{o}_1,\ldots,\mathbf{v}_k+\mathbf{o}_k) = \mathbf{t} \tag{3}$$

for $(\mathbf{o}_1,\ldots,\mathbf{o}_k) \in \mathcal{O}^k$. Observe from Equation 2 that this system remains linear in the presence of the linear emulsifier maps $\mathbf{E}_{ij} \in \mathbb{F}_q^{m\times m}$. Thus, the signer can efficiently compute a preimage $\{\mathbf{s}_i = \mathbf{v}_i+\mathbf{o}_i\}_{i\in[k]}$ of $\mathbf{t}$. To verify a MAYO signature, one needs to check if the given $\{\mathbf{s}_i\}_{i\in[k]}$ satisfy Equation 3.

D. Introduction to Fault Injection Attacks

Fault injection attacks are active attacks where the attacker is required to interact physically with the victim's device to instantly tamper with a processor's electrical inputs (e.g., voltage or clock). By violating the safe ranges of these operating parameters, a fault can occur within the processor, resulting in skipped instructions or corrupt memory transactions. Therefore, the attacker exploits the faulted output to reveal the secret information.

A fault attack is described through a fault attack model. This model summarizes the type of error it produces (e.g. skipping the execution of a specific instruction or flipping a bit in data), the timing precision, or the number of faults. The latter characteristic defines the order of the attack. For instance, first-order attacks assume that an attacker can inject only one error per execution of the target algorithm. Similarly, second-order attacks assume that an attacker can induce two errors per execution, and so forth. It is plausible, that the feasibility of the fault injection attack decreases with the order of the attack i.e., the number of injected faults.

III. Fault-Injecion Attack against MAYO

The main idea of our attack is to perturb the sampling of the vinegar variables in a way that the resulting faulted signature reveals a vector from the secret oil space. This oil vector enables an attacker to recover the entire oil space in polynomial time via techniques borrowed from the reconciliation attack. As described in Section II, the secret key is solely given by the secret linear oil space O. Thus, an attacker is able to forge signatures, as soon as she recovered O. This Section first recalls the details of the reconciliation attack, and then describes scenarios in which an attacker can reveal an oil vector through fault injection attacks.

Algorithm 1 KeyGen()

$\mathbf{O} \leftarrow \mathbb{F}_q^{o \times (n-o)}$
seed $\leftarrow \{0,1\}^\lambda$
for i from 1 to m **do**
  $\mathbf{P}_i^{(1)} \leftarrow \text{Expand}(\text{seed} \| \text{P1} \| i)$
  $\mathbf{P}_i^{(2)} \leftarrow \text{Expand}(\text{seed} \| \text{P2} \| i)$
  $\mathbf{P}_i^{(3)} \leftarrow -\mathbf{O}\mathbf{P}_i^{(1)}\mathbf{O}^\top - \mathbf{O}\mathbf{P}_i^{(2)}$
end for
$\text{pk} = (\text{seed}, \{\mathbf{P}_i^{(3)}\}_{i \in \{1,\dots,m\}})$
$\text{sk} = (\text{seed}, \mathbf{O})$
return (pk, sk)

Algorithm 2 Sign(M, sk)

$(\text{seed}, \mathbf{O}) \leftarrow \text{sk}$
$\text{salt} \leftarrow \{0,1\}^{2\lambda}$
$t \leftarrow \text{Hash}(M \| \text{salt})$
$\mathcal{P}^*(\mathbf{x}_1, \dots, \mathbf{x}_k) \leftarrow \sum_{i=1}^k E_{ii}\mathcal{P}(\mathbf{x}_i) + \sum_{1 \le i < j \le k} E_{ij}\mathcal{P}'(\mathbf{x}_i, \mathbf{x}_j)$
$\mathbf{v}_i \leftarrow \mathbb{F}_q^{n-m} \times \{0\}^m$
if $\mathcal{P}^*(\mathbf{v}_1 + \mathbf{o}_1, \dots, \mathbf{v}_k + \mathbf{o}_k)$ does not have full rank **then**
  return to step 5
end if
Solve $\mathbf{P}^*(\mathbf{v}_1 + \mathbf{o}_1, \dots, \mathbf{v}_k + \mathbf{o}_k) = \mathbf{t}$ for $\mathbf{o}_1, \dots, \mathbf{o}_k \in \text{RowSpace}(\mathbf{O} \| \mathbf{I}_o)$.
return $\sigma = (\text{salt}, \{\mathbf{s}_i = \mathbf{v}_i + \mathbf{o}_i\}_{i \in [k]})$

Algorithm 3 Verify(M, pk, σ)

$(\text{salt}, \{s_i\}_{i \in [k]}) \leftarrow \sigma$
$\mathbf{t} \leftarrow \text{Hash}(M \| \text{salt})$
$\mathbf{t}' \leftarrow \sum_{i=1}^k E_{ii}\mathcal{P}(\mathbf{s}_i) + \sum_{1 \le i < j \le k} E_{ij}\mathcal{P}'(\mathbf{s}_i, \mathbf{s}_j)$
if $\mathbf{t} = \mathbf{t}'$ **then**
  return accept
else
  return reject
end if

Fig. 1: The MAYO signature algorithm [Beu21b].

A. Reconciliation Attack

The reconciliation attack tries to find a collection of linearly independent vectors $\boldsymbol{o}_i \in O$ until a complete basis of the secret oil space O is found. The first vector $\boldsymbol{o}_1$ has to be obtained by solving the system $\mathcal{P}(\boldsymbol{o}_1) = \mathbf{0}$ consisting of m quadratic polynomials. As $dim(O) = o$, we expect a unique solution, if o affine constraints are fixed in the entries of $\boldsymbol{o}_1$. Thus, we need to find a solution to a quadratic system of m equations in $n - o$ variables. Obviously, the parameters of MAYO are chosen in a way, that the required computational complexity is beyond the respective security level.

However, as soon as a first vector $\boldsymbol{o}_1 \in O$ is recovered, finding additional vectors becomes significantly more efficient. E.g. for $\boldsymbol{o}_2$, we now have the combined system

$$\begin{cases} \mathcal{P}(\boldsymbol{o}_2) = 0 \\ \mathcal{P}'(\boldsymbol{o}_1, \boldsymbol{o}_2) = 0 \end{cases}$$

at hand. For fixed $\boldsymbol{o}_1$, the bilinear differential map $\mathcal{P}'(\boldsymbol{o}_1, \boldsymbol{o}_2) = \mathbf{0}$ imposes m linear equations in the entries of $\boldsymbol{o}_2$. This follows from the fact that $\mathcal{P}'(\boldsymbol{o}_1, \boldsymbol{o}_2)$ consists of m bilinear polynomials $p_i'(\boldsymbol{o}_1, \boldsymbol{o}_2)$, each providing a linear equation in n variables - the entries of $\boldsymbol{o}_2$. Since $\dim(O) = o > n - m$, we can fix $n - m$ of those variables, solve the linear system and receive a new oil vector $\boldsymbol{o}_2 \in O$. This process can be iterated for $\boldsymbol{o}_3, \dots, \boldsymbol{o}_o$ to recover a complete basis of O with complexity $\mathcal{O}(o \cdot m^3)$. One just needs to be careful that the newly recovered oil vectors are linearly independent of the previous ones. For further reading, we refer to [DYC+08] and [Beu21b].

Since it is impracticable to recover the starting vector $\mathbf{o}_1$ by algebraic methods, we instead apply the fault injection attack described in Section III-B to obtain $\mathbf{o}_1$. As described above, the computations to retrieve the remaining oil space are indeed trivial. We implemented the key recovery algorithm using the Sagemath [The24] package. For the MAYO_1 parameter set submitted to NIST which sets $q = 16, n = 66, m = 64, o = 8, k = 9$, the secret oil space O can be recovered in a matter of seconds on an consumer-grade laptop, given that the attacker already knows one oil vector $\boldsymbol{o}_1 \in O$.

B. Fault Injection Attack

We now turn our attention to fault injection scenarios that can reveal a vector from the secret oil space.

a) Assumptions: The attacks suggested in the following are first-order fault injection attacks and assume an attacker to be able to skip one specific instruction during the signing process. The resulting faulted signature is used to recover the secret key.

b) Key Idea: The main idea of our attack is to inject a loop-abort fault during the sampling of the vinegar variables (Line 5 in the Sign function, Figure 1), disturbing the sampling in a way that an oil vector can be recovered from the resulting faulted signature. Assuming that the vinegar vectors are sampled in sequential order, a loop-abort fault results in the sampling of one or more vinegar vectors being skipped. Note that the sampling of the vinegar vectors is implemented in a sequential manner in both the NIST reference implementation submitted as well the implementation provided with the original paper [Beu21b], [BCC+23].

We present two scenarios in which such a loop-abort fault can lead to key recovery. First, assume that the vinegar vectors are initialized to a constant value before being sampled. In that case, a loop-abort fault will lead to one or more vinegar values being set to this constant. For simplicity, assume that value to be zero. Then, skipping the sampling of the k'th vinegar vector will lead to the faulted faulted signature $\boldsymbol{s} = (\boldsymbol{s}_1, \dots, \boldsymbol{s}_k) = (\boldsymbol{v}_1 + \boldsymbol{o}_1, \dots, \boldsymbol{v}_k + \boldsymbol{o}_k) = (\boldsymbol{v}_1 + \boldsymbol{o}_1, \dots, \mathbf{0} + \boldsymbol{o}_k)$, directly revealing the oil vector $\boldsymbol{o}_k$.

The second scenario uses a differential fault attack. Here, the attacker first lets the device compute a signature $\boldsymbol{s}$ correctly, and then injects a loop-abort fault during the computation of a second signature $\boldsymbol{s}'$. If this loop-abort fault results in at least

one vinegar vector $\boldsymbol{v}'_i$ being assigned the same value as in the first (correct) signature, then we can launch an attack as follows. The attacker subtracts an obtained correct (not faulted) signature $\boldsymbol{s}$ and the faulted signature $\boldsymbol{s}'$ and receives $\boldsymbol{s} - \boldsymbol{s}' = (\boldsymbol{s}_1 - \boldsymbol{s}'_1, \ldots, \boldsymbol{s}_k - \boldsymbol{s}'_k)$. Observe that for the entry i, where $\boldsymbol{v}_i = \boldsymbol{v}'_i$ holds, we have $\boldsymbol{s}_i - \boldsymbol{s}'_i = \boldsymbol{v}_i + \boldsymbol{o}_i - \boldsymbol{v}'_i - \boldsymbol{o}'_i = \boldsymbol{o}_i - \boldsymbol{o}'_i$. As a result, we again have obtained a vector from the secret oil space. We highlight that this scenario does not require the other vinegar values to be the same (and therefore, works even if the signature generation is randomized), but only that at least one vinegar value of the faulted signature is equal to a vinegar value of the correct signature. This could happen, for example, if the variable holding the vinegar vectors is merely allocated on the stack, but not set to an initial value before sampling. In this case, the vinegar vectors are set to whatever value is currently in the respective location – which could be the values of the vinegar vector from the prior signature generation.

In both scenarios, the attacker can use the obtained vector from the secret oil space to recover the entire oil space in polynomial time. Note that the attacker must be careful not fault the sampling of *all* vinegar vectors, as otherwise the equation $\mathcal{P}^*(\boldsymbol{v}_1 + \boldsymbol{o}_1, \ldots, \boldsymbol{v}_k + \boldsymbol{o}_k) = \boldsymbol{t}$ might not have a solution.

IV. Experimental Validation

a) Simulations: We have verified the described attacks through both Sagemath simulations as well as a fault injection executed on an STM32F4 board. For the Sagemath simulations, we modified the Sage reference implementation submitted as part of MAYO's NIST submission [BCC+23] to simulate the fault injection scenarios described in Section III-B. For both fault injection scenarios, we were able to recover the key in a matter of seconds on a consumer-grade laptop using our Sagemath implementation of the reconciliation attack.

b) Practical Fault Injection Attack: To verify that our attack works in practice, we launched the fault injection attack against the proof of concept implementation [BC22] provided with the original MAYO paper [Beu21b], executed on an STM32F4 target board having 1 MB of Flash memory and 192 KB of RAM. The target board is mounted on an UFO board which has three 20-pin female headers into which the target board fits. The proof of concept implementation of the original MAYO paper calls a function that samples random vinegar values during signature generation. This `sample_vinegar` function, depicted in Listing 1, samples vinegar values in a nested loop. In our attack, the attacker tries to trigger a loop-abort fault in this loop, causing an early exit out of the outer sampling loop of the vinegar values (Line 5 in Listing 1).

```
void sample_vinegar(unsigned char *inputs){[...]
  int c = 0;
  for (int i = 0; i < K; ++i){
//Trigger the CW' GPIO pin here if i==K-1
    for (int j = 0; j < paramN-O; ++j){
          while(randomness[c]%(1<<PRIME_BITS) >= PRIME){
        c++;
      }
          inputs[i*paramN + j] = randomness[c];
          c++;
        }

  }
}
```

Listing 1: Sampling the vinegar values

To bypass the sampling loop, we perform a clock-glitching attack using the ChipWhisperer. The ChipWhisperer generates the base clock for the microcontroller, with the clock frequency set to 7.327 MHz. The devices under test can raise a GPIO pin to trigger the clock modification. Upon detecting the GPIO as raised, the ChipWhisperer modifies the clock pulse, causing the microcontroller to skip instructions. This GPIO pin is triggered by the code under test. In our experiments, we modified the MAYO code to raise the GPIO pin when sampling the vinegar values (i.e., in the `sample_vinegar` function). The clock modification by the ChipWhisperer system occurs within a single clock cycle. It adjusts the clock as follows: during the targeted clock period, the ChipWhisperer delays the rising clock edge for `width` percent of the clock period after leaving the rising edge unchanged for `offset` percent of the clock period, with the parameters being adjustable by the attacker.

For simplicity, we modified the code to initialize the vinegar values to zero. We then triggered a loop-abort fault, with parameters and trigger adjusted to skip the sampling of just one vinegar vector. This reveals an oil vector through the resulting signature – we can recover the oil space through a reconciliation attack as described earlier. Our results confirm the efficacy of the presented fault-injection attacks.

V. Countermeasures

Our security analysis shows that the implementation of MAYO [BC22] is vulnerable to fault injection attacks, and an attacker might be able to reveal the full secret key with one fault. Securing an implementation against the presented attacks is non-trivial. First, note that the attacks lead to valid signatures, and therefore, cannot be mitigated by a signature check. Second, note that zeroing the vinegar values is also sufficient to perform key recovery, and thus, a simple loop counter inside the loop sampling the vinegar value is not sufficient. The differential fault attack can also not be mitigated through randomness in the signature generation, since only the faulted vinegar vector needs to be the same, all other vinegar vectors can differ between the faulted and the non-faulted signature. Our analysis therefore helps to isolate critical regions in the MAYO code that need dedicated protection, but further research is required to design dedicated and efficient countermeasures against the presented fault injection attacks.

VI. Conclusion

In conclusion, this paper presents the first end-to-end fault injection attack on the MAYO scheme, a candidate in the ongoing NIST post-quantum cryptography standardization process. We demonstrate that a single fault injection can successfully disclose the secret key. This underscores the need for protective measures against such physical attacks. By studying the attack surface of MAYO, we are able to propose lightweight and easy-to-implement countermeasures to enhance the scheme's resilience, emphasizing the necessity of implementing these protections to safeguard the integrity and security of the MAYO scheme in practical applications. Our work serves as a starting point for future work, extending our analysis of MAYO's attack surface against physical attacks, thereby enabling the design of secure, but still efficient implementations.

References

[AKKM22] Thomas Aulbach, Tobias Kovats, Juliane Krämer, and Soundes Marzougui. Recovering Rainbow's Secret Key with a First-Order Fault Attack. In *International Conference on Cryptology in Africa*. Springer, 2022.

[BC22] Ward Beullens and Sofía Celi. Mayo implementation. https://github.com/WardBeullens/MAYO, 2022.

[BCC+23] Ward Beullens, Fabio Campos, Sofía Celi, Basil Hess, and Matthias Kannwischer. MAYO-algorithm specifications. MAYO team. https://pqmayo.org/assets/specs/mayo.pdf, 2023.

[Beu21a] Ward Beullens. Improved cryptanalysis of UOV and Rainbow. In *Advances in Cryptology–EUROCRYPT 2021*. Springer, 2021.

[Beu21b] Ward Beullens. MAYO: Practical Post-Quantum Signatures from Oil-and-Vinegar Maps. In *Selected Areas in Cryptography*. Springer, 2021.

[Beu22] Ward Beullens. Breaking Rainbow Takes a Weekend on a Laptop. In *Annual International Cryptology Conference*, pages 464–479. Springer, 2022.

[BPSV19] Ward Beullens, Bart Preneel, Alan Szepieniec, and Frederik Vercauteren. Reference implementation of the LUOV signature scheme. https://github.com/WardBeullens/LUOV, 2019.

[DS05] Jintai Ding and Dieter Schmidt. Rainbow, a new Multivariable Polynomial Signature Scheme. In *ACNS*, volume 5, pages 164–175. Springer, 2005.

[DYC+08] Jintai Ding, Bo-Yin Yang, Chia-Hsin Owen Chen, Ming-Shing Chen, and Chen-Mou Cheng. New differential-algebraic attacks and reparametrization of rainbow. In *Applied Cryptography and Network Security*, pages 242–257. Springer, 2008.

[FKNT22] Hiroki Furue, Yutaro Kiyomura, Tatsuya Nagasawa, and Tsuyoshi Takagi. A new fault attack on uov multivariate signature scheme. In *International Conference on Post-Quantum Cryptography*. Springer, 2022.

[HSMR23] Florian Hirner, Michael Streibl, Ahmet Can Mert, and Sujoy Sinha Roy. Whipping the mayo signature scheme using hardware platforms. Cryptology ePrint Archive, Paper 2023/1267, 2023. https://eprint.iacr.org/2023/1267.

[HTS11] Yasufumi Hashimoto, Tsuyoshi Takagi, and Kouichi Sakurai. General fault attacks on multivariate public key cryptosystems. In *Post-Quantum Cryptography*, pages 1–18. Springer Berlin Heidelberg, 2011.

[KL19] Juliane Krämer and Mirjam Loiero. Fault attacks on uov and rainbow. In Ilia Polian and Marc Stöttinger, editors, *Constructive Side-Channel Analysis and Secure Design*, 2019.

[KS98] Aviad Kipnis and Adi Shamir. Cryptanalysis of the Oil and Vinegar Signature Scheme. In *Advances in Cryptology—CRYPTO'98*. Springer, 1998.

[Pat00] Jacques Patarin. Cryptanalysis of the matsumoto and imai public key scheme of eurocrypt'98. *Designs, codes and cryptography*, 20(2):175–209, 2000.

[SK20] Kyung-Ah Shim and Namhun Koo. Algebraic fault analysis of uov and rainbow with the leakage of random vinegar values. *IEEE Transactions on Information Forensics and Security*, 15:2429–2439, 2020.

[SMA+24] Oussama Sayari, Soundes Marzougui, Thomas Aulbach, Juliane Krämer, and Jean-Pierre Seifert. Hamayo: A fault-tolerant reconfigurable hardware implementation of the mayo signature scheme. In *COSADE*. Springer, 2024.

[The24] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version x.y.z)*, 2024. `https://www.sagemath.org`.

B

Enhancing Implementation Security

In this chapter, we include our contributions towards more implementation security of UOV-based signature schemes.

Contents

Appendix B.1 contains a copy of the PQCRYPTO'25 paper *SoK: On the Physical Security of UOV-based Signature Schemes* [**A**CK25].

In Appendix B.2, we include the COSADE'24 paper *HaMAYO: A Fault-Tolerant Reconfigurable Hardware Implementation of the MAYO Signature Scheme* [SM**A**+24].

SoK: On the Physical Security of UOV-Based Signature Schemes

Thomas Aulbach[1( )], Fabio Campos[2], and Juliane Krämer[1]

[1] University of Regensburg, Regensburg, Germany
{thomas.aulbach,juliane.kraemer}@ur.de
[2] Bonn-Rhein-Sieg University of Applied Sciences, Sankt Augustin, Germany
campos@sopmac.de

Abstract. Multivariate cryptography currently centres mostly around UOV-based signature schemes: All multivariate round 2 candidates in the selection process for additional digital signatures by NIST are either UOV itself or close variations of it: MAYO, QR-UOV, SNOVA, and UOV. Also schemes which have been in the focus of the multivariate research community, but are broken by now - like Rainbow and LUOV - are based on UOV. Both UOV and the schemes based on it have been frequently analyzed regarding their physical security in the course of the NIST process. However, a comprehensive analysis regarding the physical security of UOV-based signature schemes is missing.

In this work, we want to bridge this gap and create a comprehensive overview of physical attacks on UOV and its variants from the second round of NIST's selection process for additional post-quantum signature schemes, which just started. First, we collect all existing side-channel and fault attacks on UOV-based schemes and transfer them to the current UOV specification. Since UOV was subject to significant changes over the past few years, e.g., adaptions to the expanded secret key, some attacks need to be reassessed. Next, we introduce new physical attacks in order to obtain an overview as complete as possible. We then show how all these attacks would translate to MAYO, QR-UOV, and SNOVA. To improve the resistance of UOV-based signature schemes towards physical attacks, we discuss and introduce dedicated countermeasures. As related result, we observe that certain implementation decisions, like key compression techniques and randomization choices, also have a large impact on the physical security, in particular on the effectiveness of the considered fault attacks. Finally, we provide implementations of UOV and MAYO for the ARM Cortex-M4 architecture that feature first-order masking and protection against selected fault attacks. We benchmark the resulting overhead on a NUCLEO-L4R5ZI board and validate our approach by performing a TVLA on original and protected subroutines, yielding significantly smaller t-values for the latter.

Author list in alphabetical order; see https://www.ams.org/profession/leaders/CultureStatement04.pdf. This work has been supported by the German Federal Ministry of Education and Research (BMBF) under the project SASPIT (ID 16KIS1858). Furthermore, this work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - project number 50550035. Date of this document: 2025-01-31.

R. Niederhagen and M.-J. O. Saarinen (Eds.): PQCrypto 2025, LNCS 15577, pp. 199–231, 2025.
https://doi.org/10.1007/978-3-031-86599-2_7

Keywords: Multivariate Cryptography · Physical Security · Fault Attacks · Side-channel Analysis · Masking · ARM Cortex-M4 · TVLA

1 Introduction

At the latest since the standards FIPS 203, FIPS 204, and FIPS 205 have been published, post-quantum cryptography can be considered mature enough for practical use. For practical applications, however, not only standardized schemes are interesting, but also other post-quantum schemes which offer useful alternatives, for instance, regarding key sizes or computation times. Depending on the application, schemes that are not (yet) standardized might be therefore also in demand.

A very promising post-quantum family for signature schemes with interesting properties is multivariate cryptography. Two multivariate signature schemes, including Rainbow [17], also advanced to the third round of NIST's standardization process for post-quantum cryptography (PQC).[1] However, powerful attacks against both schemes showed that these two schemes should not be standardized [8,42]. The two attacks, especially the one on Rainbow by Beullens, brought the unbalanced oil and vinegar (UOV) scheme [29] back into the interest of the research community. Although UOV had already been published at the end of the 1990s and is the basis for Rainbow, research concentrated on Rainbow after its publication because it seemed to be more efficient than UOV both in terms of required memory and computation time. This is also why UOV has initially not been submitted to the NIST PQC standardization process. However, although Rainbow is a generalization of the oil-and-vinegar construction underlying UOV, the sweeping attack on Rainbow does not apply to UOV. This makes UOV again a very interesting signature scheme since it withstands cryptanalysis since more than two decades.

UOV in particular and multivariate signature schemes in general feature very small signatures. Short signatures (and fast verification) were also features of signature schemes NIST was explicitly interested in for their call for additional post-quantum signatures.[2] The goal of this process is to diversify the post-quantum signature standards by selecting additional general-purpose signature schemes not based on structured lattices and schemes that are particularly suitable for certain applications. Hence, it was not surprising that ten out of fourty submissions to the call in 2023 have been based on multivariate cryptography.[3] For three of these schemes - 3WISE, DME, and HPPC - rapidly efficient attacks have been found. The remaining seven schemes all rely on the oil-and-vinegar principle, i.e., are UOV-based: MAYO [11], PROV [23], QR-UOV [20], SNOVA [44], TUOV [19], UOV [12], and VOX [35]. Hence, since then, all multivariate

[1] https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[2] https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/call-for-proposals-dig-sig-sept-2022.pdf.

[3] https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures.

signature schemes which are in the focus of the research community are based on the UOV principle. Very recently, on October 25, 2024 NIST announced the 14 schemes to advance to the next round.[4] The share of multivariate signature schemes even increased slightly, since four of the schemes advanced to round 2: MAYO, QR-UOV, SNOVA, and UOV.

When cryptographic schemes are used in practical applications, not only their mathematical security and efficiency, but also their resistance towards physical attacks is important. Hence, post-quantum schemes have been analyzed with respect to their physical security in recent years, and there is also a line of research targeting multivariate cryptography in general and UOV-based signature schemes in particular.

1.1 State of the Art and Related Work

Starting in 2011 [24], UOV-based signature schemes have been analyzed both with respect to passive, i.e., side-channel, and active, i.e., fault attacks. There are results that specifically target UOV [2,21] or another UOV-based scheme [3, 4,25,32,39,41], but also publications that analyze several schemes [24,31,34].

However, by reading these publications, one does not get a comprehensive picture of the state of the art of the physical security of UOV-based signature schemes. This has several reasons: 1) Some of the schemes have received more attention from the research community than others; simply because of their age, but also because of the existence of a mature implementation. To the best of our knowledge, there was no effort yet to study the transferability of all attacks to all UOV-based signature schemes. 2) Some of the attacks [3,32] targeted schemes that have since been proven to be insecure, like LUOV and Rainbow. Also for these attacks, it is often unknown if and how they transfer to other schemes. 3) Both the specifications of the schemes and their implementations have been subject to various changes and optimizations over time, e.g., the method of generating compressed public keys introduced in [37]. Hence, it is not even clear if the older attacks still remain a realistic threat to the current version of the algorithms. 4) Although attacks are usually published with descriptions of countermeasures, there are limited results about implemented countermeasures and their overhead, or physically secure implementations of multivariate schemes in general. This might also be due to the fact that for a long time there were no common reference implementations that could serve as a basis. 5) Moreover, since the recent history of multivariate signatures is characterized by major breaks and fixes, e.g., [8,18,42], the research community focused initially on the mathematical security of the schemes, which is natural and reasonable.

Now, with NIST's process for standardizing additional digital signature schemes commencing the second round, we have the chance to study all remaining UOV-based schemes on the basis of consistent specifications and implementations. Our goal is to advance their resistance against physical attacks by creating an extensive survey of possible attack vectors, implementing countermeasures, and measuring their overhead directly in comparison to existing implementations.

[4] https://csrc.nist.gov/Projects/pqc-dig-sig/round-2-additional-signatures.

1.2 Contribution

In this project, we provide a comprehensive overview on the physical security of today's most relevant UOV-based signature schemes, i.e., the schemes MAYO, QR-UOV, SNOVA, and UOV, that are being analyzed in the second round of NIST's standardization process for additional signature schemes. While we analyze all four schemes in this work, we specifically focus on UOV and MAYO, since they provide the more advanced implementation: The more practical a physical attack is carried out, the greater its relevance. To practically perform a physical attack, however, a target implementation is needed. Therefore, UOV and MAYO can be considered the most relevant signature schemes in the field of multivariate cryptography with respect to physical attacks. They both provide optimized implementations for the ARM Cortex-M4 architecture and both are analyzed several times in the literature, already.

In this work, we provide a complete overview of known side-channel and fault attacks against UOV-based schemes and derive their core attack vectors. We set out to understand if further vulnerabilities exist. In finding new vulnerabilities, we concentrate on side-channel attacks, since so far most attacks are based on faults. We provide a complete overview for all considered schemes regarding their susceptibility towards physical attacks: For all attacks, both known and new, we analyze if and how they can be transferred to all considered schemes. For all attacks, we provide existing and newly developed countermeasures. Moreover, we describe the effect of certain implementation decisions on the physical security of the schemes and derive implementation guidelines from this.

For UOV and MAYO, based on existing optimized M4 implementations, we provide first-order masked implementations for the ARM Cortex-M4 architecture. Additionally, the implementations include protection against the most relevant fault attacks. We benchmark the resulting overhead on a NUCLEO-L4R5ZI board and validate our approach by performing a test vector leakage assessment (TVLA) on original and protected subroutines, yielding significantly smaller t-values for the latter. Our implementation is available to the public at https://github.com/SoK-Psec-UOV-based/code.

1.3 Organization

In Sect. 2, we discuss the notation used in this work, describe the UOV scheme and the main differences between UOV and MAYO, QR-UOV, and SNOVA, and explain why an attack on a UOV-based scheme often leads to full key recovery once a single oil vector is found. In Sects. 3 and 4, we present a comprehensive collection of fault attacks and side-channel attacks, respectively, on UOV-based signature schemes. We present both known and new attacks, and analyze if they can be transferred to the current specifications of UOV, MAYO, QR-UOV, and SNOVA. In Sect. 5, we describe implementation guidelines that we derived from the analysis of the attacks. In Sect. 6, we present implementations of UOV and MAYO that include protection against selected fault attacks from Sect. 3 and countermeasures against all side-channel attacks from Sect. 4, i.e., are first-order masked. In Sect. 7, we conclude this work.

2 Background

2.1 Notation

In this paper we describe physical attacks and countermeasures to existing UOV-based signature schemes, which are all submitted to NIST's call for additional digital signatures for the PQC standardization process. Thus, we deem it reasonable to use exactly the notations and conventions introduced in the specification of the designated signature scheme. E.g., the discussions on UOV follow the notation given in [12], the findings about MAYO follow the notation in [11], etc. This requires the reader to be cautious at time, since the respective specifications might use different names or variables for similar objects. Nevertheless, we think this is the correct way, since using a fixed notation in here, would force the reader to adjust the notation on their own when referring to different schemes.

2.2 UOV

Here, we would like to recall UOV in its current form and the most important properties. We will also link its abstract mathematical description to the steps listed in the pseudo code.

The main objects in multivariate cryptography are multivariate quadratic maps $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$. In general, it is hard to find a solution $\mathbf{s} \in \mathbb{F}_q^n$ to a given target $\mathbf{t} \in \mathbb{F}_q^m$ such that $\mathcal{P}(\mathbf{s}) = \mathbf{t}$. This task is also known as the MQ Problem. It can be solved in polynomial time in the very under- or overdetermined case, i.e. $m \geq n(n+1)/2$ or $n \geq m(m+1)$, but is believed to be exponentially hard even for large scale quantum computers if $n \sim m$. By installing a secret trapdoor into the public map $\mathcal{P}$, one can render this task efficiently solvable and construct a signature scheme thereof. In UOV this trapdoor is a m-dimensional linear subspace $O \subset \mathbb{F}_q^n$ - the oil space - with the property $\mathcal{P}(\mathbf{o}) = 0$ for all $\mathbf{o} \in O$. The message μ, together with a random `salt`, is mapped to a target value $\mathbf{t}$ in the codomain $\mathbb{F}_q^m$ using a cryptographic hash function $\mathcal{H}$, i.e. $\mathbf{t} = \mathcal{H}(\mu||\texttt{salt})$. Computing a signature boils down to finding a preimage $\mathbf{s} \in \mathbb{F}_q^n$ with $P(\mathbf{s}) = \mathbf{t}$. To this end one can deploy the following method, if knowledge of the oil space is provided. First, one picks a vector $\mathbf{v}$ at random and then solves the equation

$$\mathcal{P}(\mathbf{v} + \mathbf{o}) = \mathcal{P}(\mathbf{v}) + \mathcal{P}(\mathbf{o}) + \mathcal{P}'(\mathbf{v}, \mathbf{o}) = \mathbf{t} \tag{1}$$

for $\mathbf{o} \in O$. The map $\mathcal{P}' : \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q^m$ defined by the equation above, is called the differential of $\mathcal{P}$ and is bilinear and symmetric [6]. Viewing the oil vector $\mathbf{o}$ as a linear combination of its m basis vectors given in O ensures $\mathcal{P}(\mathbf{o}) = \mathbf{0}_m$ and shows clearly that

$$\mathcal{P}'(\mathbf{v}, \mathbf{o}) = \mathbf{t} - \mathcal{P}(\mathbf{v}) \tag{2}$$

is a system of m linear equations in m variables. If there exists a solution, it can be computed efficiently, if not, one samples a new $\mathbf{v}$ and tries again. Together, the vinegar and oil vector yield a preimage $\mathbf{s} = \mathbf{v} + \mathbf{o}$ to the target $\mathbf{t}$, which constitutes the core of the signature.

Key Generation. Within key generation we need to generate the m-dimensional oil space O and a multivariate quadratic map $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ that vanishes on O. The map $\mathcal{P}$ is a sequence of m quadratic polynomials $p_1(\mathbf{x}), \ldots, p_m(\mathbf{x})$ in n variables. The linear and constant part of the polynomials is omitted, and the coefficients of the quadratic monomials can be stored in an upper triangular matrix $\mathbf{P}_i$, such that evaluating the polynomial p_i at a value $\mathbf{a}$ can be realized by computing $p_i(\mathbf{a}) = \mathbf{a}^\top \mathbf{P}_i \mathbf{a}$.

The oil space is chosen to be the space spanned by the rows of a matrix $\bar{\mathbf{O}} = (\mathbf{O}^\top \ \mathbf{I}_m)$, where $\mathbf{O} \in \mathbb{F}_q^{(n-m)\times m}$ is sampled uniformly at random. Thus, to guarantee that the quadratic polynomials p_i vanish on O, i.e. $\mathbf{o}^\top \mathbf{P}_i \mathbf{o} = 0$ for all $\mathbf{o} \in O$, one can set the term

$$(\mathbf{O}^\top \ \mathbf{I}_m) \begin{pmatrix} \mathbf{P}_i^{(1)} & \mathbf{P}_i^{(2)} \\ 0 & \mathbf{P}_i^{(3)} \end{pmatrix} \begin{pmatrix} \mathbf{O} \\ \mathbf{I}_m \end{pmatrix} = \mathbf{O}^\top \mathbf{P}_i^{(1)} \mathbf{O} + \mathbf{O}^\top \mathbf{P}_i^{(2)} + \mathbf{P}_i^{(3)}$$

to zero. This is achieved by sampling $\mathbf{P}_i^{(1)} \in \mathbb{F}_q^{(n-m)\times(n-m)}$ (upper triangular) and $\mathbf{P}_i^{(2)} \in \mathbb{F}_q^{(n-m)\times m}$ uniformly at random and setting $\mathbf{P}_i^{(3)}$ accordingly to

$$\mathbf{P}_i^{(3)} = \mathtt{Upper}(-\mathbf{O}^\top \mathbf{P}_i^{(1)} \mathbf{O} - \mathbf{O}^\top \mathbf{P}_i^{(2)}).$$

The described procedure is exactly the key generation algorithm of UOV, which is depicted in Fig. 1. Hereby, $\mathbf{O}$ and $\mathbf{P}_i^{(1)}, \mathbf{P}_i^{(2)}$ can be expanded from a private or public seed, respectively.

UOV.CompactKeyGen() // $csk = (\mathtt{seed}_{\mathtt{sk}}, \mathtt{seed}_{\mathtt{pk}})$

1 : $\mathtt{seed}_{\mathtt{sk}} \leftarrow \{0,1\}^{\mathtt{sk\_seed\_}len}$
2 : $\mathtt{seed}_{\mathtt{pk}} \leftarrow \{0,1\}^{\mathtt{pk\_seed\_}len}$
3 : $\mathbf{O} \leftarrow \mathtt{Expand}_{\mathtt{sk}}(\mathtt{seed}_{\mathtt{sk}})$
4 : $\{\mathbf{P}_i^{(1)}, \mathbf{P}_i^{(2)}\}_{i\in[m]} \leftarrow \mathtt{Expand}_P(\mathtt{seed}_{\mathtt{pk}})$
5 : **for** $j = i, \ldots, m$ **do**
6 : $\quad \mathbf{P}_i^{(3)} \leftarrow \mathtt{Upper}(-\mathbf{O}^\top \mathbf{P}_i^{(1)} \mathbf{O} - \mathbf{O}^\top \mathbf{P}_i^{(2)})$
7 : $cpk \leftarrow (\mathtt{seed}_{\mathtt{pk}}, \{\mathbf{P}_i^{(3)}\}_{i\in[m]})$
8 : $csk \leftarrow (\mathtt{seed}_{\mathtt{pk}}, \mathtt{seed}_{\mathtt{sk}})$
9 : **return** (cpk, csk)

Fig. 1. UOV key generation algorithm

Secret Key Expansion. As indicated above, it is necessary to solve Eq. 2 with respect to $\mathbf{o}$ during signing. The i-th component of $\mathcal{P}'(\mathbf{v}, \mathbf{o})$ is computed by $\mathbf{v}^\top(\mathbf{P}_i + \mathbf{P}_i^\top)\mathbf{o}$ and $\mathbf{o}$ is written as a linear combination of its basis vectors in

$\bar{\mathbf{O}}$, where the coefficients $\mathbf{x}$ are the variables we need to solve for. Thus, we can prepare the term $(\mathbf{P}_i + \mathbf{P}_i^\top)\mathbf{o}$ by setting $\mathbf{S}_i = (\mathbf{P}_i^{(1)} + \mathbf{P}_i^{(1)\top})\mathbf{O} + \mathbf{P}_i^{(2)}$ and store it in the expanded secret key *esk*, since it is independent from the message μ. The coefficients in $\mathbf{P}_i^{(1)}$ are necessary to compute $\mathcal{P}(\mathbf{v})$, so they are also added to *esk*. The pseudo code of the secret key expansion is presented in Fig. 2.

UOV.ExpandSK(csk) // $csk = (\mathtt{seed}_{\mathtt{sk}}, \mathtt{seed}_{\mathtt{pk}})$

1 : $\mathbf{O} \leftarrow \mathtt{Expand}_{\mathtt{sk}}(\mathtt{seed}_{\mathtt{sk}})$
2 : $\{\mathbf{P}_i^{(1)}, \mathbf{P}_i^{(2)}\}_{i\in[m]} \leftarrow \mathtt{Expand}_P(\mathtt{seed}_{\mathtt{pk}})$
3 : **for** $j = i, \ldots, m$ **do**
4 : $\quad \mathbf{S}_i \leftarrow (\mathbf{P}_i^{(1)} + \mathbf{P}_i^{(1)\top})\mathbf{O} + \mathbf{P}_i^{(2)}$
5 : $esk \leftarrow (\mathtt{seed}_{\mathtt{sk}}, \mathbf{O}, \{\mathbf{P}_i^{(1)}, \mathbf{S}_i\}_{i\in[m]})$
6 : **return** esk

Fig. 2. Algorithm that expands csk to esk in UOV

Signing. The signing algorithm is shown in Fig. 3. After generating the `salt` and deriving the target value $\mathbf{t}$, the vinegar vector $\mathbf{v}$ is sampled. Note, that the last m entries of $\mathbf{v}$ are set to zero, which enables a more efficient implementation. With $\mathbf{v}$ at hand, one computes the remaining part of $\mathcal{P}'(\mathbf{v}, \mathbf{o})$ in Line 7, which represents the linear part of the system given in Line 10. In order to evaluate $\mathbf{y} = \mathcal{P}(\mathbf{v})$ in Line 9, only the submatrix $\mathbf{P}_i^{(1)}$ is necessary, since the last entries of $\mathbf{v}$ were chosen to be zero. Solving the derived linear system reveals the coefficients $\mathbf{x}$ of the oil vector $\mathbf{o} = \bar{\mathbf{O}}\mathbf{x} = [\mathbf{O}\mathbf{x}, \mathbf{x}]$. Finally, the sum of the vinegar and oil vector yield the signature $\mathbf{s} = [\mathbf{v}, \mathbf{0}_m] + \bar{\mathbf{O}}\mathbf{x}$.

Public Key Expansion and Verification. To verify that the signer really found a preimage $\mathbf{s}$ under the public map $\mathcal{P}$ of the target vector $\mathbf{t}$, one needs to expand the public coefficients in $\mathbf{P}_i^{(1)}$ and $\mathbf{P}_i^{(2)}$ from the seed and check if $\mathcal{P}(\mathbf{s}) = \mathbf{t}$ really holds. We omit the pseudo code of these two functionalities, since we do not focus on them in the main part of this work.

Variants. We have to differentiate between the variants `uov-classic`, `uov-pkc` and `uov-pkc+skc`. In `uov-pkc+skc` the secret key is stored in a compact way *csk*, such that the function *UOV.ExpandSK* is called before signing with *UOV.Sign* and has to be protected as well. In `uov-classic` and `uov-pkc` *UOV.ExpandSK* is part of the key gen, since the secret key is stored in an expanded manner.

One considerable drawback of UOV are its large public keys, due to the amount of coefficients needed to define the public map $\mathcal{P}$. Even in the variants with compressed public keys `uov-pkc+skc` and `uov-pkc`, where a large fraction

$UOV.Sign(esk, \mu)$ // $esk = (\mathtt{seed}_{sk}, \mathbf{O}, \{\mathbf{P}_i^{(1)}, \mathbf{S}_i\}_{i\in[m]})$

1 : $\mathtt{salt} \leftarrow \{0,1\}^{\mathtt{salt\_}len}$
2 : $\mathbf{t} \leftarrow \mathtt{H}(\mu||\mathtt{salt})$ // $\mathbf{t} \in \mathbb{F}_q^m$
3 : **for** $ctr = 0, \ldots, 255$ **do**
4 : $\quad \mathbf{v} \leftarrow \mathtt{Expand}_v(\mu||\mathtt{salt}||\mathtt{seed}_{\mathtt{sk}}||ctr)$ // $\mathbf{v} \in \mathbb{F}_q^{n-m}$
5 : $\quad \mathbf{L} \leftarrow \mathbf{0}_{m\times m}$
6 : $\quad$ **for** $i = 1, \ldots, m$ **do**
7 : $\quad\quad$ Set i-th row of $\mathbf{L}$ to $\mathbf{v}^\top \mathbf{S}_i$
8 : $\quad$ **if** $\mathbf{L}$ is invertible **then**
9 : $\quad\quad \mathbf{y} \leftarrow [\mathbf{v}^\top \mathbf{P}_i^{(1)} \mathbf{v}]_{i\in[m]}$ // $\mathbf{y} \in \mathbb{F}_q^m$
10 : $\quad\quad$ Solve $\mathbf{L}\mathbf{x} = \mathbf{t} - \mathbf{y}$ for $\mathbf{x}$
11 : $\quad\quad \mathbf{s} \leftarrow [\mathbf{v}, \mathbf{0}_m] + \bar{\mathbf{O}}\mathbf{x}$ // $\mathbf{s} \in \mathbb{F}_q^n$
12 : $\quad\quad \sigma \leftarrow (\mathbf{s}, \mathtt{salt})$
13 : $\quad\quad$ **return** σ
14 : **return** 0

Fig. 3. Algorithm that signs a message μ in UOV

of the coefficients is expanded from a short seed, the part $\mathbf{P}_i^{(3)}$ still needs to be stored explicitly and contains around $m^3/2$ coefficients in $\mathbb{F}_q$. Here, one factor m comes from the number of polynomials m in the public map $\mathcal{P}$, but the remaining factor $m^2/2$ depends on the dimension of the oil space $dim(O) = m$, which are chosen to be equal in UOV.

2.3 MAYO, QR-UOV and SNOVA

In [7], Beullens presented MAYO, a signature scheme that also employs the oil-and-vinegar principle but reduces the dimension of the oil space from m to o drastically, which in turn reduces the number of coefficients in $\mathbf{P}_i^{(3)}$ to $mo^2/2$ and eventually shrinks down the public key size. The problem is that one can not simply decrease the dimension of the oil space from m to o, with $o \ll m$ and continue as before, since the system $\mathcal{P}'(\mathbf{v}, \mathbf{o}) = \mathbf{t} - \mathcal{P}(\mathbf{v})$ in Eq. (2) becomes a system of m linear equations in o variables and bears no solution with high probability. Thus an adaption to the signing and verification procedure was necessary and the following solution was suggested: The public key map $\mathcal{P}$ is stretched into a larger whipped map $\mathcal{P}^* : \mathbb{F}_q^{kn} \to \mathbb{F}_q^m$, such that it accepts k input vectors $\mathbf{s} \in \mathbb{F}_q^n$. This is realized by setting

$$\mathcal{P}^*(\mathbf{s}_1, \ldots, \mathbf{s}_k) := \sum_{i=1}^{k} \mathbf{E}_{ii}\mathcal{P}(\mathbf{s}_i) + \sum_{1\le i<j\le k} \mathbf{E}_{ij}\mathcal{P}'(\mathbf{s}_i, \mathbf{s}_j), \tag{3}$$

where the matrices $\mathbf{E}_{ij} \in \mathbb{F}_q^{m\times m}$ are fixed system parameters with the property that all their non-trivial linear combinations have rank m. The map $\mathcal{P}^*$ vanishes

on the subspace $O^k = \{(\mathbf{o}_1, \ldots, \mathbf{o}_k) |$ with $\mathbf{o}_i \in O$ for all $i \in [k]\}$ of dimension ko. This gives back some degrees of freedom when looking for a solution in the new domain $\mathbb{F}_q^{kn}$, i.e. the system obtained from

$$\mathcal{P}^*(\mathbf{v}_1 + \mathbf{o}_1, \ldots, \mathbf{v}_k + \mathbf{o}_k) = \mathbf{t} \tag{4}$$

after randomly sampling and inserting $(\mathbf{v}_1, \ldots, \mathbf{v}_k) \in \mathbb{F}_q^{kn}$, is a system of m linear equations in ko variables. Consequently, if the parameters are chosen such that $ko > m$, it possible to sample signatures $\mathbf{s} = (\mathbf{s}_1, \ldots, \mathbf{s}_k) = (\mathbf{v}_1 + \mathbf{o}_1, \ldots, \mathbf{v}_k + \mathbf{o}_k)$ similar as before, despite the small dimension of the initial oil space O.

Another attempt in reducing the public key size of UOV is made by QR-UOV. It is very similar to UOV, except that elements of a quotient ring $\mathbb{F}_q[x]/(f)$ are employed, instead of just finite field elements $\mathbb{F}_q$. Let l be a positive integer and $f \in \mathbb{F}_q[x]$ of degree l. Then any element g of the quotient ring $\mathbb{F}_q[x]/(f)$ uniquely defines a $l \times l$ matrix Φ_g^f over $\mathbb{F}_q$ such that $(1, x, \ldots, x^{l-1}) \cdot \Phi_g^f = (g, xg, \ldots, x^{l-1}g)$. The mapping from g to its polynomial matrix Φ_g^f is an injective ring homomorphism from $\mathbb{F}_q[x]/(f)$ to $\mathbb{F}_q^{l \times l}$ and every element of $\mathcal{A}_f \coloneqq \{\Phi_g^f \in \mathbb{F}_q^{l \times l} | g, \in \mathbb{F}_q[x]/(f)\}$ can be represented by only l elements in $\mathbb{F}_q$. For the construction of QR-UOV, the central maps F_i and public maps P_i are set to be composed of such matrices. This additional structure allows that not every element of these maps are stored since l^2 entries of Φ_g^f can be represented by just l coefficients of g. The central maps F_i are secret, easily invertible maps, that allow the signer to generate signatures efficiently. Their structure is hidden via the transformation $P_i = S^\top F_i S$.

SNOVA also work with quotient rings, but apply even more structure to the central and public maps. Each component of the central map $F = [F_1, \ldots, F_m] : \mathcal{R}^n \to \mathcal{R}^m$ is defined by

$$F_i = \sum_{\alpha=1}^{l^2} A_\alpha \cdot \left(\sum_{(j,k) \in \Omega} X_j (Q_{\alpha 1} F_{i,jk} Q_{\alpha 1}) X_k \right) \cdot B_\alpha,$$

where the $F_{i,jk}$ are randomly chosen from $\mathcal{R}$, A_α and B_α are invertible elements randomly chosen from $\mathcal{R}$, and $Q_{\alpha 1}, Q_{\alpha 2}$ are invertible matrices randomly chosen from $\mathbb{F}_q[S]$. Here $\mathcal{R}$ is the matrix ring $\mathbb{F}_q^{(l \times l)}$ and $\mathbb{F}_q[S]$ is a commutative symmetric subring of $\mathcal{R}$. Like in UOV, the public matrices are then set to fulfill the equation $P_i = F_i \circ T$ and therefore, are of similar type than the central matrices.

For more details about MAYO, QR-UOV, and SNOVA we refer to the respective specifications given in [11,20,44].

2.4 One Vinegar or One Oil Vector Is (In Many Cases) Sufficient for Complete Key Recovery

With respect to signature schemes, it is always an interesting task to determine the amount of data that is necessary to forge signatures, especially from an adversarial point of view. When the secret key is just a seed, as in MAYO, it

might become infeasible to recover this seed with side-channel attacks. But it still is useful to obtain information about the values of the expanded seed. Conversely, in `uov-classic` the secret key is of enormous size and only a fraction of it suffices to forge signatures. However, for UOV-based signature schemes, there is a pretty and common answer to this problem: knowledge of the employed oil space O allows to efficiently generate signatures. Even more, there are algebraic methods, that allow to recover the entire space in polynomial time as soon as one or several (depending on the concrete dimensions) oil vectors $\mathbf{o} \in O$ are found.

Regarding MAYO, it was already clear from the description in [7, Section 4.1] that one oil vector is enough to efficiently recover O. The case for UOV was treated in [2] and [36], where again only a single oil vector is needed, since all existing UOV parameter sets fulfill the equation $n - m \leq 2m$.

In [36], Pebereau also elaborated on the very unbalanced case, i.e. when $n > 3m$. In this scenario, there might be several oil vectors needed for polynomial time key recovery, namely β of them, where β is the smallest integer such that $n - \beta m \leq 2m$ holds. This becomes interesting, since some parameter sets suggested by QR-UOV and SNOVA lie in the very unbalanced case with $n \geq 6m$ or even up to $n \geq 9m$. In these cases, the security of the scheme might already be compromised when one or two oil vectors are leaked, but attackers are only able to forge signatures in polynomial time, when they get their hands on β of them.

However, if it is possible to reveal one oil vector by means of a physical attack, repeating it on several signing procedures will probably leak more of them. Thus, the attack strategy remains the same, even in the very unbalanced case. Most of the time obtaining a vinegar vector $\mathbf{v}$ is equally strong, since the corresponding oil vector $\mathbf{o}$ can be recovered by subtracting it from the signature $\mathbf{s} = \mathbf{v} + \mathbf{o}$. Consequently, every time one of them is used in a computation, it has to be protected against physical attacks.

3 Fault Attacks on UOV-Based Signatures

In this section, we study the vulnerability of UOV-based signature schemes towards fault attacks. We investigate existing fault attacks on UOV-based signature schemes and provide a more complete catalog of vulnerabilities by also presenting new attacks. Our analysis results in a list of functions that need to be protected in order to achieve an implementation that is more resistant to fault attacks.

We first analyze the vulnerability of UOV, as this scheme builds the base for all remaining signature schemes analyzed in this work. Subsequently, we explain how these attacks translate to the MAYO signature scheme and its current Cortex M4 implementation. Finally, we consider the case of the remaining UOV-based signatures, QR-UOV and SNOVA. We expect these rather young schemes to be subject to various algorithmical changes and code modifications in the future, thus the findings of this work can be seen as a starting point towards their physical security and as a general estimation whether the respective scheme might be vulnerable to this kind of attack.

To provide a better overview of the existing, transferred, and new vulnerabilities, we summarize the findings of this section in Table 1.

Table 1. Overview of existing and new fault attacks on UOV, MAYO, QR-UOV, and SNOVA. Regarding the feasibility of the attacks, we refer to the specifications submitted to the NIST call for additional signatures in mid-2023. When there are differences between the three UOV-variants `uov-classic`, `uov-pkc` and `uov-pkc+skc`, deterministic and randomized MAYO or the variants `SNOVA-esk` and `SNOVA-ssk`, we list them individually in the given order. With ✓ we state that an attack is possible, while ✗ means the opposite. By ⋆ we denote that an attack is generally possible, but the technical execution is more difficult than in the initially presented attack.

Attack description	Source	Initially for	Feasible in current version	Target
Fix vinegar vector	[3,4,24] [25,31,41]	Rainbow UOV MAYO	UOV: ✓ MAYO: ✓ QR-UOV: ✓ SNOVA: ✓	`UOV.Sign Line 4` [11] `Alg.8 Line 16,18` [20] `Alg.2 Line 10` [44] `Alg.11 Line 8`
Rowhammer on oil space O	[32]	LUOV	UOV: ✓ | ✓ | ⋆ MAYO: ⋆ QR-UOV: ⋆ SNOVA: ✓ | ⋆	uncompressed secret key in memory
Bit flip on stored secret matrices	[21,24,31]	Rainbow UOV	UOV: ✓ | ✓ | ⋆ MAYO: ✓ | ✗ QR-UOV: ⋆ SNOVA: ✗	uncompressed secret key in memory
Prevent addition of oil and vinegar	[39]	MAYO	UOV: ✓ MAYO: ✓ QR-UOV: ✗ SNOVA: ✗	`UOV.Sign Line 11` [11] `Alg.8 Line 44` [20] `Alg.2 Line 18` [44] `Alg.11 Line 14`
Disturb linear system setup	This work	UOV	UOV: ✗ MAYO: ✓ | ✗ QR-UOV: ✗ SNOVA: ✗	`UOV.Sign Line 7` [11] `Alg.8 Line 22-33` [20] `Alg.2 Line 11,12` [44] `Alg.11 Line 9,10`

3.1 Existing Attacks

In the following, we discuss fault attacks against UOV-based schemes that are present in the literature. If the attack was initially not developed for UOV itself, we try to give a detailed analysis of its applicability to UOV.

Fault Injection to Fix the Vinegar Vector. There has been a series of works [3,4,25,31,41] studying the effectiveness of an instruction skip in Line 4 of Fig. 3. These works set the number of necessary faulted signatures to $m = n - v$, which lies in the range from several dozens to hundreds, depending on the security level. Using the findings in [2,36] and [30] it is now clear that one

faulted signature is enough for efficient key-recovery. This can be obtained by the following observations.

If the instruction skip is applied successfully, it leads to the reusage of the vinegar variables that are still stored in $\mathbf{v}$ from a previous signing process. Denote the unfaulted previous signature by $\mathbf{s}$ and the faulted signature by $\tilde{\mathbf{s}}$. The oil vectors $\mathbf{o}$ and $\tilde{\mathbf{o}}$, which are computed during the respective signature generation and added to the vinegar vector $\mathbf{v}$, are different, but both are elements of the oilspace O. Thus, we have

$$\tilde{\mathbf{s}} - \mathbf{s} = (\mathbf{v} + \tilde{\mathbf{o}}) - (\mathbf{v} - \mathbf{o}) = \tilde{\mathbf{o}} - \mathbf{o} \in O,$$

since O is a linear subspace of $\mathbb{F}_q^n$. With a secret oil vector at hand, full key recovery can be achieved in a matter of seconds.

In [25], the authors present two attacks - an absorption skipping and absorption abort attack - directly on the SHAKE256 function that is used to derive the vinegar vectors. These lead to an predictable output of the sampling and therefore reveal the vinegar vectors $\mathbf{v}_i$. They do not make any assumptions about the memory initialization of the device. Since they present this attack specifically on MAYO, we will resume to this in Sect. 3.3.

RowHammer to Alter a Value in O. In [32], the authors present a Rowhammer attack to cause bit flips in the secret transformation $\mathbf{T}$ in LUOV and show how this can be exploited to recover individual bits of $\mathbf{T}$. Repeated execution of the fault attack leads to partial knowledge of the secret $\mathbf{T}$, one bit for every faulted signature. Once enough key bits of $\mathbf{T}$ are recovered, the attacker can apply algebraic analysis techniques to increase the efficiency of the attack and limit the number of faulted signatures that need to be obtained for a full key recovery.

Briefly summarized, the QuantumHammer attack works as follows. The secret data is stored in the DRAMs in memory cells. There is a certain threshold of the voltage level that determines whether a capacitor represents a binary one or zero. If one manages to activate neighboring rows rapidly, this can cause variations in the voltage level of the victim cells due to induction. When a certain threshold is passed, this results in a bit-flip from 0 to 1 or vice versa. Since the attacker does not know at which entry of the secret transformation $\mathbf{T}$ the bit flip occurred, he needs to apply a bit-tracing algorithm to locate the faulted spot and learn the initial value of the flipped bit. The bit flip in $\mathbf{T}$ causes an error in the last part of the signing algorithm in LUOV, where the (vinegar part of the) signature is finally computed by

$$\begin{pmatrix} s_1 \\ \vdots \\ s_v \end{pmatrix} = \begin{pmatrix} t_{11} & \cdots & t_{1m} \\ \vdots & \ddots & \vdots \\ t_{v1} & \cdots & t_{vm} \end{pmatrix} \times \begin{pmatrix} o_1 \\ \vdots \\ o_m \end{pmatrix} + \begin{pmatrix} v_1 \\ \vdots \\ v_v \end{pmatrix}. \tag{5}$$

Thus, a bit flip of the entry t_{ij} leads to a faulted signature entry $\tilde{s}_i$, where the erroneous entry $\tilde{s}_i$ differs from the correct one s_i by o_j. The bit-tracing algorithm now tries to correct the faulted signature by successively adding the values o_k

for $k \in \{1, \ldots, m\}$ to the entries s_l for $l \in \{1, \ldots, v\}$. Once the signature is corrected, the position (i, j) of the induced bit flip is successfully located. This works particularly well, since the entries t_{ij} in LUOV are binary, while the values o_l are elements of a larger field, depending on the parameter choice, e.g., of $\mathbb{F}_{2^8}$. Therefore, the chances that different bit flips in T, say at position t_{ij} and t_{ik}, cause the same error in s are rather small, since then $o_j = o_k$ needs to hold. For more details, please see [32, Section 3.3].

This attack can be transferred to UOV, since the targeted operation in Eq. (5) similarly appears in the signing algorithm of UOV, see Line 11 of Fig. 3. Here, the oilspace O takes the role of the linear transformation T and they behave equivalently. Since the second block of $\bar{\mathbf{O}} = (\mathbf{O}, \mathbf{I}_m)$ is the identity, the values in $\mathbf{x} \in \mathbb{F}_q^m$ are visible to any attacker as the last m entries of the signature $\mathbf{s}$, analogically to the vector $\mathbf{o}$ in LUOV.

However, when it comes to the bit-tracing algorithm we need to be more careful, since in UOV the entries $t_{i,j}$ are not binary anymore, but elements of $\mathbb{F}_{2^8}$ as well. Let $s_i = \sum_{l=1}^{m} t_{il} \cdot o_l + v_i$ be the entry of the unfaulted signature vector. Introducing a single bit flip to t_{ij} results in an faulted entry $\tilde{s}_i = \sum_{l=1}^{m} t_{il} \cdot o_l + v_i + f_{ij} \cdot o_j$, with $f_{ij} \in \mathbb{F}_{2^8}$ having hamming weight 1. Now, if we have $f_{ij} \cdot o_j = f_{ik} \cdot o_k$, for two different indices $j \neq k$, this implies a bit flip corresponding to f_{ij} results in the same faulted signature entry $\tilde{s}_i$ as a bit flip corresponding to f_{ik} would. The bit-tracing algorithm is still able to correct the faulted signature, but would not be able to decide if the deviation results from f_{ij} or f_{ik} and thus, could not uniquely determine the position of the introduced bit flip.

Bit Flip in Matrices of `esk`. See [21,24,31] for this attack. It is shown that changing a single coefficient of $\mathbf{F}_i$ leads to reduction of the UOV instance to a smaller one. Recent UOV implementations, including the NIST submission [12], are not working with the central maps $\mathbf{F}_i$ anymore, but include the public matrices $\mathbf{P}_i^{(1)}$ and auxiliary matrices $\mathbf{S}_i = (\mathbf{P}_i^{(1)} + \mathbf{P}_i^{(1)\top})\mathbf{O} + \mathbf{P}_i^{(2)}$ in their secret key. However, this is merely a change in notation, since the former blocks $\mathbf{F}_i^{(1)}$ and $\mathbf{F}_i^{(2)}$ of $\mathbf{F}_i$ were defined just like that. The analysis [21,24,31] can be applied to the new setting accordingly and a single altered coefficient in $\mathbf{P}_i^{(1)}$ or $\mathbf{S}_i$, leads to faulted signature $\mathbf{s}'$, such that $\mathcal{P}(\mathbf{s}')$ and $\mathbf{t}$ indeed deviate in exactly one entry. This difference then allows conclusions about the used oil space O.

Fault Injection to Skip the Addition of the Vinegar and Oil Parts. This fault attack targets the addition in Line 11 of Fig. 3, where the vinegar and oil part are added to receive the signature $\mathbf{s} = \mathbf{v} + \mathbf{o}$. It highly depends on the chosen implementation, but if an attacker is able to exclude the vinegar part by an instruction skip, this is threatening. If there is a way to avoid the contribution of $\mathbf{v}$, this directly reveals $\mathbf{s} = \mathbf{o}$ in the signature, which enables key recovery. Currently, this is implemented via a `memcpy` call that copies $\mathbf{v}$ to $\mathbf{s}$, so this line needs to be protected.

Remark 1. Sampling the vinegar vector $\mathbf{v}$ is randomized by including the `salt` as input to the expand function in Line 4 of Fig. 3. If we would deal with a

deterministic approach instead, where the signing procedure generates the same vinegar vector $\mathbf{v}$ and outputs identical signatures when a message is signed twice, an adversary could exploit the faulted signature $\mathbf{s}' = \mathbf{v}$ as follows: Subtracting the un-faulted signature $\mathbf{s}$ of the same message $\mathbf{s} - \mathbf{s}' = \mathbf{v} + \mathbf{O}\mathbf{x} - \mathbf{v} = \mathbf{O}\mathbf{x} = \mathbf{o}$ reveals an oil vector.

A similar strategy that also only works in the deterministic setting, is to disturb the computation of the oil vector during signing. If one is able to enforce the computation of a different solution $\mathbf{x}'$ to the linear system, then this again reveals an oil vector. Subtracting the two signatures $\mathbf{s}' - \mathbf{s} = \mathbf{v} + \mathbf{O}\mathbf{x}' - \mathbf{v} - \mathbf{O}\mathbf{x} = \mathbf{O}(\mathbf{x} - \mathbf{x}')$ would cancel out the identical vinegar values and reveal an non-zero oil vector, since $\mathbf{x}' \neq \mathbf{x}$. A similar strategy is applied by the following fault attack.

3.2 New Attack

In addition to the attacks gathered in the section above, we identified the following spots, where exploits by an adversary are conceivable and countermeasures should be applied.

Fault Injection to Disturb Linear System Setup (Skip the New Assignment of L or Y). Disturbing the linear system $\mathbf{L}\mathbf{x} = \mathbf{t} - \mathbf{y}$ leads to a different solution $\mathbf{x}'$, which might lead to key recovery in the deterministic setting, as explained in Remark 1. Perturbing the values in $\mathbf{L}$ or $\mathbf{y}$ can be achieved by skipping Line 5 of Fig. 3 and some or all of the loop in Line 6–7 or introducing faults to the computation of $\mathbf{y}$ in Line 9, respectively. Hereby, it is important that not a single row of $\mathbf{L}$ remains all zero due to the fault attack. This would imply that the system is not solvable, which results in a second iteration of the signing loop, where $\mathbf{v}$ and $\mathbf{L}$ are refreshed. However, this strategy is only successful when the signature is deterministic and thus, current implementations are not vulnerable.

3.3 Transferability to MAYO

The functionalities of UOV are a subset of those needed to implement MAYO. Therefore, it seems natural to conclude that the listed fault attacks for UOV easily transfer to MAYO. Although this is more or less true, there are some minutiae to bear in mind. 1) The applicability of attacks that target the secret key in memory depends on the format (compressed vs. uncompressed) of the stored keys. MAYO always uses compressed keys, which makes MAYO resistant towards such attacks, although classic UOV is vulnerable. 2) Some attacks depend on the randomization choice of the scheme which might deviate to UOV. More details on these two aspects can be found in Sects. 5.1 and 5.2. 3) While UOV works with only one pair of oil and vinegar vectors, there are k of them in MAYO. Nevertheless, a single known oil vector is enough to perform key recovery in polynomial time. Thus, certain attacks might become technically easier in MAYO, since there are k targets available, while only one of them needs to be recovered.

Fault Injection to Fix One or More Vinegar Vectors. This fault attack can be easily transferred to MAYO. Current implementations clear the vinegar vectors at the end of the signing procedure as a security measure to avoid reusing attacks. However, in MAYO one all-zero vinegar vector will not lead to an unsolvable system in Line 37 of MAYO.Sign. This is due to the fact that multiple matrices $\mathbf{M}_i[j,:] = \mathbf{v}_i^\top \mathbf{L}_j$ contribute to the linear part of the system $\mathbf{A}$. Consequently, having some $\mathbf{v}_i$ set to zero will not necessarily result in a non-invertible system and another iteration of the signing loop. Thus, by inserting an instruction skip or loop abort in Line 16 or Line 18 such that one or more (but not all) vinegar vectors remain zero makes the corresponding oil vector $\mathbf{o}_i$ visible in the signature. Furthermore, the authors of [25] present three attacks on this subroutine, in total. Two of them target the absorption phase of the `shake256` internally and the other one forces the unknown input to be zero. In all cases the attacker can predict the outcome of the sampling process which reveals one or more vinegar vectors. Rightfully predicting randomly sampled intermediate values during the signing phase, by directly attacking the `shake256` function, is disastrous in many cryptographic primitives, as is also shown, e.g., in [27]. Thus, we only consider the latter attack as scheme-specific to MAYO, which should be treated with dedicated countermeasures. Instead of zeroing buffers with sensitive information at the end of signing, overwriting these buffers with random data is suggested in [25]. Countermeasures of the aforementioned attacks are discussed in more detail in Sect. 5.4.

RowHammer to Alter a Value in O. In MAYO the secret key only contains a seed $\texttt{seed}_{sk}$. The MAYO.API.sign algorithm (Algorithm 10 in [11]) employs the two functionalities MAYO.ExpandSK (Algorithm 6 in [11]) and MAYO.Sign (Algorithm 8 in [11]) to derive the expanded secret key `esk` from the seed and sign the message using `esk`. The secret oil space $\mathbf{O}$, which is the target of the original fault attack, is therefore not a part of the compressed secret key `csk` Thus, it is not stored in memory permanently and less accessible for bit-flip attempts. Furthermore, the addressed variable is zeroed at the end of the signing procedure as a security measure. This is not depicted in the pseudo code, but realized in the submitted implementations [11]. Thus we do not deem the attack given in [32] to be technically feasible in this scenario.

However, in case an adversary could manage to insert a bit-flip during signing, while $\mathbf{O}$ is stored as part of `esk` in the memory, a similar description as in Sect. 3.1 would apply, since the oil space takes an equivalent role as in UOV.

Bit Flip in Matrices of `esk`. Let $\mathbf{s}'$ be the faulted signature, that is generated when a single bit-flip is applied to either of the matrices $\{\mathbf{P}_i^{(1)}, \mathbf{S}_i\}$ in the expanded secret key, The crucial part of this fault attack against UOV is that $\mathcal{P}(\mathbf{s})$ and $\mathbf{t}$ only deviate in one entry. In contrast, the whipped public map $\mathcal{P}^*(\mathbf{s}_1, ..., \mathbf{s}_k) = \sum_{i=1}^{k} \mathbf{E}_{ii}\mathcal{P}(\mathbf{s}_i) + \sum_{1\leq i<j\leq k} \mathbf{E}_{ij}\mathcal{P}'(\mathbf{s}_i, \mathbf{s}_j)$ in MAYO is of different structure. The $\binom{k}{2}$-many emulsifier maps $\mathbf{E}_{ij}$ and the accumulation of all the transformed terms ensures that $\mathcal{P}^*(\mathbf{s}')$ and $\mathbf{t}$ deviate in most of the entries, even if only a single bit-flip was applied. We confirmed this statement with simula-

tions, where we manually change a single bit in one of the $\mathbf{P}_i^{(1)}$ or $\mathbf{S}_i$. Thus, the introduced fault attack can not be transferred to MAYO, at least not in a straightforward way.

However, for the deterministic variant the situation is different. The bit flips lead to altered solution vectors $\mathbf{x}' = (\mathbf{x}'_1, \ldots, \mathbf{x}'_k)$. This most likely results in an invalid signature $\mathbf{s}' = (\mathbf{s}'_1, \ldots, \mathbf{s}'_k) = (\mathbf{v}_1 + \mathbf{O}\mathbf{x}'_1, \ldots, \mathbf{v}_k + \mathbf{O}\mathbf{x}'_k)$. But if we compare that to the correct signature $\mathbf{s} = (\mathbf{s}_1, \ldots, \mathbf{s}_k) = (\mathbf{v}_1 + \mathbf{O}\mathbf{x}_1, \ldots, \mathbf{v}_k + \mathbf{O}\mathbf{x}_k)$ of the same message, we see that the vinegar part is identical in the deterministic scenario. Consequently, the term $\mathbf{s}_i - \mathbf{s}'_i = \mathbf{O}\mathbf{x}_i + \mathbf{O}\mathbf{x}'_i$ would reveal an oil vector, since $\mathbf{O}\mathbf{x}'_i \in O$.

Fault Injection to Skip the Addition of the Vinegar and Oil Parts. In general, this fault attack also applies to MAYO. In the MAYO signing procedure [11, Alg.2, Line 45] the sum of k pairs of vinegar and oil vectors is computed. In the current implementation this is achieved by adding both components into another variable `s` via `mat_add(vi, Ox, s + i * param_n, ..., 1);`. Regarding the fault attack, this approach is more secure than first copying `vi` to `s`, and then adding `Ox` to it, as it is done in UOV. However, before the actual addition happens, the vinegar part gets reassigned via `vi = Vdec + i * (param_n - param_o);`. If this instruction can be skipped and `vi` remains empty or assigned with a certain constant value, then an oil vector can be recovered with the approach given in Remark 1.

Fault Injection to Disturb Linear System Setup (Skip the New Assignment of A or Y). There are plenty of options to introduce faults during the linear system set up in [11, Alg.2] between Line 22 and 33, which might change the solution vectors $\mathbf{x'}_i$ and therefore also the resulting oil vectors $\mathbf{o'}_i$. In the non-randomized setting this could be exploited, similar to the description given in Remark 1.

3.4 Transferability to QR-UOV and SNOVA

In this section we discuss if and how the fault attacks in Table 1 can be transferred to QR-UOV and SNOVA. Both schemes employ the quotient ring structure, but this does not have a large impact on the mentioned attacks.

Fault Injection to Fix the Vinegar Vector. This attack works analogously for QR-UOV and SNOVA. As described in Sect. 3.1, the crucial part here is that the signature is composed of an oil and a vinegar vector. Then, the repeated usage of two identical vinegar vectors makes them canceling each other out, when those signatures are subtracted from each other. In QR-UOV and SNOVA this last step of mixing the oil and vinegar terms is represented by applying the secret linear transformation $\mathbf{s} = S^{-1}(y_1, \ldots, y_v, y_{v+1}, \ldots, y_n)$ in [20, Alg.2, Line 18] or $\mathbf{sig} = [T](X_0, \ldots, X_{v-1}, \tilde{X}_0, \ldots, \tilde{X}_{o-1})$ in [44, Alg.11, Line 14], where

the first v entries represent the vinegar and the last o entries represent the oil variables. Due to the block matrix structure of S and T, with an identity and zero block in the first column, the vinegar variables contribute unaltered to the signature.

RowHammer to Alter a Value in S or T. In general, this attack works as initially described for LUOV or adapted to UOV. However, QR-UOV and the variant SNOVA-ssk use compressed secret keys. Consequently, the linear transformations S and T, which are only part of the expanded secret keys, are not permanently stored in memory at a specific location. Thus, from a technical point of view, the execution of the fault attack becomes way more difficult. The time slot where the fault can be induced successfully is reduced to signing time and furthermore, the resulting bit-flip is not permanent, since the secret key will be expanded again in the next signing procedure. However, regarding SNOVA-esk the attack would work equivalently to the original one on LUOV.

Bit flip in Matrices of `esk`. The transfer of this fault attack is again non-trivial. Considering the notation of QR-UOV and SNOVA, there are now the matrices $F_{i,1}, F_{i,1}$ and $F_i^{11}, F_i^{12}, F_i^{21}$ under attack. It is not the quotient-ring structure that determines if a bit-flip in one of these matrices yields useful information for the attacker, but the structure of the public map. In the verification of QR-UOV, the signature is evaluated with the bare public map $\mathcal{P}$, similar to UOV. Therefore, we again are in the case that $\mathcal{P}(\mathbf{sig}')$ and $\mathbf{t}$ only differ in a single entry, which is exploitable.

SNOVA, in contrast, has the whipping structure of MAYO, as pointed out by Beullens in [9]. In Corollary 2 he states that the SNOVA public map can be written as $\mathcal{P}(\mathbf{U}) = \sum_{j=0}^{l-1} \sum_{k=0}^{l-1} \mathbf{E}_{j,k} \mathcal{B}(\mathbf{u}_j, \mathbf{u}_k)$, where the matrices $\mathbf{E}_{j,k}$ have a block diagonal structure with m identical blocks of size $\mathbb{F}_q^{(l^2 \times l^2)}$ on the diagonal. This goes well with the result of our simulations, where a bit-flip introduced to one of the matrices F_i^{11}, F_i^{12}, or F_i^{21}, caused the vectors $\mathcal{P}(\mathbf{sig})$ and $\mathbf{t}$ to deviate in l^2 entries. Thus, the mentioned fault attack can not be applied to SNOVA, at least not without profound modification.

Fault Injection to Skip the Addition of the Vinegar and Oil Parts. QR-UOV and SNOVA are both randomized schemes. As a result, the only way to mount this attack is to stop the vinegar variables - and only them - from contributing to the signature (see Remark 1).

We analyzed the submitted reference implementation from QR-UOV and came to the conclusion that this is not possible with a first-order fault attack. The signature is computed via `sig->s[i] = Fql_sub(vineger[i], t);`, where the vinegar part `vineger` is not altered or reassigned beforehand and the oil part is encoded in `t`. Skipping this instruction would detain both parts from appearing in `s`.

In SNOVA the case is a little different. They first copy the vinegar entry to the signature `gf16m_clone(signature...[index], X...[index]);` and afterwards add the oil entry to it. However, since this is done entry-wise, aborting

the loop or similar strategies would not be successful, as they also prevent the remaining oil entries from contributing.

Thus, both schemes do not seem to be vulnerable regarding that attack. Remarkably, this is not due to the quotient ring structure, but to their current implementation details.

Fault Injection to Disturb Linear System Setup. As concluded in Sect. 3.2, this attack only works in the deterministic setting. Since both QR-UOV and SNOVA do not expand the vinegar variables from a fixed seed, but generate them randomly, the given attack is not feasible.

4 Side-Channel Attacks on UOV-Based Signatures

In this section, we investigate the security of UOV-based signature schemes in terms of side-channel attacks. Similar to the previous section, we first recall and transfer existing attacks to UOV. With the goal of being as exhaustive as possible, we then consider further potential vulnerabilities. The resulting attacks are subsequently adapted to MAYO, QR-UOV, and SNOVA. Table 2 presents an overview of this section.

4.1 Existing Attacks

The following attacks against UOV or a familiar scheme are present in the literature. Both existing attacks target the *UOV.Sign* routine shown in Fig. 3 and focus on subroutines where secret data is multiplied with public values.

Power Analysis of the Evaluation of the Vinegar Vector. The target of this side-channel attack is the computation $\mathbf{y} = [\mathbf{v}^\top \mathbf{P}_i^{(1)} \mathbf{v}]_{i \in [m]}$ given in Line 9 of Fig. 3. The vinegar vector $\mathbf{v}$ is multiplied from both sides to m matrices $\mathbf{P}_i^{(1)}$ containing public values. This marks an evident entrance door for side channel attacks via power analysis. In [2] the authors showed how to exploit this vulnerability with a profiling attack, that gets along with only a single attack trace. In the profiling phase, the entries v_i of $\mathbf{v}$ are set by hand to certain known values. Then, the considered function is called and power traces are gathered - labeled with the respective value in v_i as reference. During the attack phase, the power trace of the execution of $\mathbf{y} = [\mathbf{v}^\top \mathbf{P}_i^{(1)} \mathbf{v}]_{i \in [m]}$ with the unknown (and secret) vector $\mathbf{v}$ is recorded and compared to the reference traces. The attack trace is likely to have the highest correlation to the reference trace where the identical value v_i is used. In this manner, the entries of $\mathbf{v}$ are revealed, which in turn exposes a secret oil vector $\mathbf{o}$ and enables complete secret key recovery.

Table 2. Overview of existing and new side-channel attacks on UOV, MAYO, QR-UOV, and SNOVA. Regarding the feasibility of the attacks, we refer to the specifications submitted to the NIST call for additional signatures in mid-2023. When there is a difference between deterministic and randomized MAYO, we list them individually in the given order. With ✓ we state that an attack is possible. By ⋆ we denote that an attack is generally possible, but the technical execution is more difficult than in the initially presented attack.

Description: Power analysis ...	Source	Initially for	Feasible in	Target
of vinegar evaluation	[2]	UOV	UOV: ✓	`UOV.Sign Line 9`
			MAYO: ✓	[11] `Alg.8 Line 29`
			QR-UOV: ✓	[20] `Alg.2 Line 12`
			SNOVA: ✓	[44] `Alg.8 Line 3,4`
of secret matrix multiplication	[26,34]	Rainbow	UOV: ⋆	`UOV.Sign Line 11`
		UOV	MAYO: ✓ | ⋆	[11] `Alg.8 Line 44`
		MAYO[a]	QR-UOV: ⋆	[20] `Alg.2 Line 18`
			SNOVA: ⋆	[44] `Alg.11 Line 14`
of linear system setup	This work	UOV	UOV: ✓	`UOV.Sign Line 7`
			MAYO: ✓	[11] `Alg.8 Line 27,29`
			QR-UOV: ✓	[20] `Alg.2 Line 11`
			SNOVA: ✓	[44] `Alg.9 Line 3,4,14,26`
of secret key expansion	This work [26]	UOV	UOV: ✓	`UOV.ExpandSK Line 4`
		MAYO[a]	MAYO: ✓	[11] `Alg.6 Line 17`
			QR-UOV: ✓	[20] `Alg.2 Line 7`
			SNOVA: ✓	[44] `Alg.6 Line 6,7`
during key generation	This work	UOV	UOV: ✓	`UOV.CompactKeyGen Line 6`
			MAYO: ✓	[11] `Alg.5 Line 16`
			QR-UOV: ✓	[20] `Alg.1 Line 5`
			SNOVA: ✓	[44] `Alg.5 Line 5`

[a] Simultaneously to our work, Jendral and Dubrova [26] demonstrated a deep learning assisted power analysis of the corresponding functions in MAYO. Note that their attack is the only side-channel attack considered in this work that was explicitly developed against MAYO. For the others, we analyzed if the attacks against UOV can be translated to MAYO.

Power Analysis of the Linear Subspace Matrix Multiplication. The authors of [34] present a differential power analysis (DPA) on the multiplication of the linear transformation $\mathbf{T}$ with the intermediate vector $\mathbf{x}$, that is the solution to the derived linear system. In our notation here, the transformation $\mathbf{T}$ is replaced by the basis $\bar{\mathbf{O}}$ of the linear subspace O. Therefore, this attack can be seen as a power analysis of the matrix vector multiplication in Line 11 of Fig. 3. The attack takes advantage of the fact that some entries of the vector $\mathbf{x}$ are part of the signature, since they are not altered by the identity block of $\bar{\mathbf{O}}$, resp. T. In more detail, we have $\bar{\mathbf{O}}\mathbf{x} = [\mathbf{O}\mathbf{x}, \mathbf{x}]$ and $\mathbf{s} = [\mathbf{v}, \mathbf{0}_m] + [\mathbf{O}\mathbf{x}, \mathbf{x}]$. Thus, during the computation of $\mathbf{O}\mathbf{x}$, the secret entries in $\mathbf{O}$ are multiplied with known values and are consequently vulnerable to DPA.

The authors in [34] require a few dozen of repeated computations of $\mathbf{Ox}$ to recover the entries in $\mathbf{O}$ by using correlation coefficients. At the time they performed this attack on Rainbow and UOV, these schemes used a deterministic approach. This allowed the authors to make the valid assumption that $\mathbf{x}$ will not change when the same message is signed repeatedly. In the current randomized implementation, the solution vector $\mathbf{x}$ will change with every signing procedure, since every time a new vinegar vector $\mathbf{v}$ is sampled, leading to a completely different linear system $\mathbf{Lx} = \mathbf{t} - \mathbf{y}$.

Hence, the attack will not work in the presented form and needs to be adapted to the new setting. However, we still believe the considered function $\mathbf{Ox}$ needs to be protected, since sensitive data is multiplied with public values, which could be exploited with more evolved analysis methods, like profiling or machine learning techniques[5].

4.2 New Attacks

Power analysis attacks are possible on various other spots of the UOV functionalities. In *UOV.ExpandSK* and *UOV.CompactKeyGen* there is a bulk of matrix multiplications that involve the secret matrix $\mathbf{O}$ and public values stored in $\mathbf{P}_i^{(1)}$ and $\mathbf{P}_i^{(2)}$, which is clearly vulnerable. Regarding the algorithm *UOV.Sign*, we additionally identified the following operation where caution is required.

Power Analysis of the Computation of L. In Line 7 of Fig. 3 the linear part $\mathbf{L}$ of the system of equations in Line 10 is computed. Hereby, the i-th row of $\mathbf{L}$ is given by $\mathbf{v}^\top \mathbf{S}_i$. Both components involved are unknown to an attacker, which makes it harder to mount a successful power analysis attack than in the considered scenarios above, where one component is public. However, the matrices $\mathbf{S}_i$ are part of the expanded secret key, and, consequently, remain constant over various signing procedures with the same secret key. If Hamming weight information about one of the factors and their product is leaked, we have seen that blind side-channel attacks [16,38] can exploit this and reveal information about the considered factors $\mathbf{v}$ and $\mathbf{S}_i$.

Power Analysis of the Computation of S_i During Secret Key Expansion. The multiplication in Line 4 of Fig. 2 could be analyzed similarly to the side-channel attack against Line 9 of Fig. 3. This is also critical and needs to be countered. In the variant `uov-pkc+skc` compressed secret keys are used and the mentioned procedure is part of the signing process. Regarding the other two variants `uov` and `uov-pkc`, where the $\mathbf{S}_i$ are already part of the key, this functionality is attributed to key generation.

Power Analysis of the Computation of $P_i^{(3)}$ During Key Generation. The same holds for the matrix multiplications during key generation, depicted in Line 6 of Fig. 1. If an adversary is able to retrieve side-channel information here, these operations also need to be protected, following the same reasoning.

[5] The recent attack [26] confirms this conjecture.

4.3 Transferability to MAYO, QR-UOV, and SNOVA

The task of theoretically transferring the discussed side-channel attacks to MAYO, QR-UOV, and SNOVA is considerably less complicated than it was for the fault attacks. On one hand this is due to the fact that certain implementation choices, like the utilization of compressed keys, have a smaller impact on the effectiveness of side-channel attacks. On the other hand, also the scheme-specific properties are less critical here, since all four schemes contain the typical UOV-like work flow: Generate the vinegar vector(s), compute the constant and linear part of the system of equations, solve the linear system via Gauss, multiply the solution with the oil space to receive corresponding oil vector(s), and finally add the vinegar and oil part together. Thus, the vulnerable subroutines listed in Table 2 need to be executed some way or the other, and whether the scheme uses elements of the field $\mathbb{F}_{2^4}$ or $\mathbb{F}_{2^8}$ or of a quotient ring $\mathbb{F}_q[x]/(f)$, like QR-UOV and SNOVA, is not decisive for the theoretical applicability of the attack, since they also boil down to multiplications over a huge amount of field elements.

The practical execution of the attack, in contrast, will depend heavily on the chosen implementation. In [10], the MAYO team announced that they will change their specification from a bit- to a nibble-sliced representation for their keys etc. Among other things, they present an efficient implementation of MAYO on the Arm Cortex-M4, where the costly matrix multiplications are based on the Method of the Four Russians. This method deviates from previous approaches and to the best of our knowledge, there are no reported side-channel attacks against such implementations in the literature. To perform such a side-channel analysis could pose some interesting challenges and therefore, provides a charming open research questions.

For the remaining schemes QR-UOV and SNOVA, there are currently no Arm Cortex-M4 implementations available, so a concrete analysis of their side-channel security cannot yet be performed. Nevertheless, we hope Table 2 with the respective code lines can provide a good orientation about the vulnerable functions, which need to be treated with caution.

5 Implementation Guidelines

In this section, we present implementation guidelines and dedicated countermeasures to protect implementations of UOV-based signature schemes against the physical attacks presented in this work.

During our research, we identified several theoretical attack vectors - both existing and new ones - that do not lead to physical attacks in practice since the current UOV and MAYO implementations are not vulnerable against them. However, we realized that the implementation decisions that prevent these attacks do not seem to be motivated by preventing physical attacks, primarily. As an example, the utilization of compressed keys first and foremost serves the purpose of reducing key sizes, but at the same time it prevents fault attacks that alter the secret key in memory (cf. Sects. 3.3 and 5.2). Hence, we do not consider it correct to term these parts of the implementations *countermeasures*,

in contrast to concrete modifications of implementations with the aim of making the implementations more resistant towards physical attacks. Still, we consider it important to list also these implementation decisions in this section to emphasize their importance for physical attack security, which is why we term this section *implementation guidelines* instead of *countermeasures*.

5.1 Randomized Signatures

From a physical security point of view, it is desirable to utilize a randomized signature generation process. For the considered UOV implementation this is already the case. In Line 1 of Fig. 3 the `salt` s generated randomly. This `salt` among others) contributes as input to the `Hash` and `Expand` functions that are used to derive the target vector $\mathbf{t} \in \mathbb{F}_q^m$ and the vinegar vector $\mathbf{v} \in \mathbb{F}_q^{n-m}$. Thus, if the same message is signed twice, the generated signatures (and most of the intermediate values) are different. In contrast, the considered MAYO implementation offers both options - random and deterministic - that determine how the `salt` s derived in Line 10 of the signing algorithm. If the signature computation is deterministic, this is beneficial for an attacker. Subtracting signatures of identical messages leads to vinegar parts canceling each other out, possibly revealing non-zero oil vectors, if one of the two latter fault attacks in Table 1 is applied correctly. Thus, we recommend the usage of the randomized version to prevent both of these attacks. Furthermore, this helps to mitigate side-channel analysis with the goal of obtaining the sampled vinegar vectors. If the vinegar vectors vary between different signing processes, it will be much harder to apply differential power analysis methods. Nevertheless, we suggest masking as an additional countermeasure, see Sect. 5.5.

5.2 Compressed Keys

In addition to the obvious advantage of reduced key sizes, the use of compressed keys is also beneficial with respect to physical security. If the secret key only consists of a seed, there are less options to introduce exploitable faults while it is stored in memory. Recently, [21] and [32] showed that bit flips introduced to an uncompressed secret key can lead to serious leakage. Even when the precise spot of occurrence is unknown at first, there are methods to localize the bit flips and use them to achieve full key recovery. Moreover, [32] practically executed the attack on LUOV, emphasizing its relevance for UOV-based schemes.

Using compressed keys prevents both fault attacks described in this work that target the secret key in memory, namely the Rowhammer attack on the secret matrix $\mathbf{O}$ and the one introducing bit flips on the secret matrices.

5.3 Counter RowHammer Fault Attack

However, there are scenarios where key compression techniques are undesirable, e.g., in order to enable a faster signing process. Here, we introduce a method

to prevent the RowHammer fault attack on the secret subspace O for such scenarios. To be precise, it is not the secret subspace O which is stored in memory, but a certain basis of it. Right now this basis is represented in standard form, such that the identity part $\mathbf{I}_{m\times m}$ can be omitted and only the remaining $\mathbf{O} \in M_{m\times(n-m)}(\mathbb{F}_q)$ is stored. This method is memory efficient, but the standard form for a given oil space is unique, hence fixed. This enables the bit tracing algorithm used in the RowHammer attack.

Instead one could compute m random - though linearly independent - vectors of O and store this modified basis instead. During signing, we would load the modified basis from memory, compute its standard form and continue signing like usual. Afterwards we again transform the basis to a random one by building random linear combinations. This way, the explicit form of the secret key, i.e., the basis of the subspace O, would change with every signature generation, while the secret information remains the same.

The resulting overhead is obvious. On one hand, the size of the matrix to be stored increases from $M_{m\times(n-m)}(\mathbb{F}_q)$ to $M_{m\times n}(\mathbb{F}_q)$. At first glance, this seems like a considerable drawback, but the fraction of the expanded secret key that is consumed by $\mathbf{O}$ or its enlarged version is rather small compared to the matrices $\mathbf{S}_i$ that are also part of the secret key. Thus, the expanded key size would only increase from 238 to 240 KB in `uov-Ip`.

On the other hand, the effort to compute the standard form at the beginning of the signature generation and the randomized basis at the end, will increase the signing time slightly.

5.4 Modify Vinegar Variable After Usage

The fault attack that leads to re-using or zeroing (most of) the vinegar variables belongs to the most prominent ones in literature, see, e.g., [3,24,31,41]. It leads to valid signatures, since the actual signing process is unaltered by the fault. Only the vinegar variables are forced to values that are either known by the attacker, or have been used before. The part of signing that computes the actual solution to the equation $\mathcal{P}(\mathbf{v}+\mathbf{o}) = \mathbf{t}$ is executed correctly and therefore finds a correct signature. Thus, unlike many fault attacks, it can not be detected by a validity check.

Instead, one actively needs to ensure that the sampled vinegar variables are sound and vary across consecutive signing procedures. To this end, we suggest the following modification. After the sampled vinegar variables are used in Line 7 and 9 of Fig. 3, we add a vector with random values to it, i.e., `v += r`. Since the vinegar vector `v` is added to the signature s via `s += v` towards the end of the signing process, the component `r` needs to be removed from the derived signature at the end by appending the instruction `s -= r`.

Obviously this countermeasure could be circumvented with two additional instruction skips, which would lead to a third-order fault attack altogether. An attacker who is able to introduce three independent faults, however, could probably attack a signature algorithm in a simpler way and is therefore not considered

a relevant scenario in this work. Irrespective of this, depending on how the additional random value $\mathbf{r}$ is chosen, this countermeasure can even be circumvented more easily: When $\mathbf{r}$ is designed to be a new variable with randomly sampled values, the assignment of the random value to the variable $\mathbf{r}$ might be skipped, resulting in $\mathbf{r} = 0$. In this case, the countermeasure would be completely circumvented by a single instruction skip, leading to a second-order fault attack altogether, which can be considered to be realistic [14]. To avoid such second-order fault attack, we instead suggest to not initialize $\mathbf{r}$ with randomly sampled values, but to use already existing intermediate values of the signing procedure and directly add them to $\mathbf{v}$. For instance, we can make use of the unknown entries of the vectors $\mathbf{r} := \mathbf{v}^\top \cdot \mathbf{S}_i$ for any $i \in \{1, \ldots, m\}$, that are used to compose the linear system which is solved during signing.

This countermeasure can be seen as an approach to randomize the data stored in $\mathbf{v}$ after its usage, which is also suggested by [25]. Since $\mathbf{v}$ is added to the signature $\mathbf{s}$ subsequently, this furthermore employs the idea of infective computing [22]. The component $\mathbf{r}$ is a secret error, which needs to be removed finally. If an injected fault skips the addition of this error, the output of the algorithm will be incorrect and can not be exploited by an attacker.

5.5 Masking Against Power Analysis

The listed side-channel attacks in Sect. 4 all follow a similar concept, namely the power analysis of certain matrix vector multiplications, that ultimately boil down to field multiplications in $\mathbb{F}_q$. The field is rather small, e.g., $q \in \{2^4, 2^8\}$ in UOV or $q = 2^4$ in MAYO, and 32-bit processors, like the ARM Cortex-M family, treat multiple field elements at once. Even though this hampers Hamming weight analysis of the secret or vulnerable values, it has been shown that DPAs [34] or profiling attacks [2] are possible.

The most common countermeasure to prevent power analysis attacks is masking. Currently, to the best of our knowledge there are no masked implementations of UOV-based signature schemes available. In this work, we bridge this gap by providing a first masked version of UOV and MAYO. The goal is to protect the vulnerable intermediate values, mainly the oil and vinegar vectors, whenever they are used, and thereby protect against the existing and newly developed attacks in Sect. 4.1 and Sect. 4.2. Since the majority of the utilized functions in signing (and key generation) is linear, they are straightforward to mask. In the following, we provide an overview of the affected lines in the pseudo code and the measures we implemented to mitigate their vulnerability. Therefore, we refer to the pseudocode of UOV, especially the signing algorithm in Fig. 3.

- The original implementation uses `shake256` [1] to generate the vinegar vector $\mathbf{v} \in \mathbb{F}_q^n$. In fact it only samples $n - m$ entries of $\mathbf{v}$, as the last m entries are set to zero. Instead, we propose to use a masked version `masked_shake256` [5] to sample two (additive) shares of these entries, that will be used for computations later on and are combined at the end of the signing procedure.

- *Line 7, compute $\boldsymbol{v}^\top \boldsymbol{S}_i$, the linear part of the system of linear equations:* In this step we compute the coefficients of the linear part of the system of equations that needs to be solved during signing. Hereby, the vinegar vector $\mathbf{v}$ is multiplied with numerous matrices $\mathbf{S}_i$, which are part of the (expanded) secret key and do not change in subsequent signature generations. Each resulting vector represents a row in the matrix $\mathbf{L}$ that constitutes the linear system. The matrices $\mathbf{S}_i$ are defined by $\mathbf{S}_i \leftarrow (\mathbf{P}_i^{(1)} + \mathbf{P}_i^{(1)\top})\mathbf{O} + \mathbf{P}_i^{(2)}$ (see Line 4 of Fig. 2) and contain information about the secret oil space O. Therefore, we split them randomly into two additive shares. Since the function $\mathbf{v}^\top \cdot \mathbf{S}_i$ is linear in both components and the vinegar vector already arrives in two shares, we need to compute it four times, one for each combination of the respective two components.
- *Line 9, compute $\boldsymbol{v}^\top \boldsymbol{P}_i^{(1)} \boldsymbol{v}$, which contributes to the constant part of the system of linear equations:* In [2] the authors perform a profiled side-channel attack against this operation. Since the values in $\mathbf{P}_i^{(1)}$ are public, they can take those matrices as given in the public key and collect profiling traces of this operation for various known values of $\mathbf{v}$. After the profiling phase has finished, only a single attack trace of this operation with the used vinegar vector v is needed to recover its actual value. Thus, masking the vector $\mathbf{v}$ is not an option, since an attacker could just recover the value of both shares with this kind of single trace attack. Instead, we suggest to mask the values given in $\mathbf{P}_i^{(1)}$. To collect meaningful profiling traces it is crucial that an attacker knows the exact value of $\mathbf{P}_i^{(1)}$. By masking the entries of these matrices, we prohibit this strategy and render the collected profiling traces useless. Consequently, we compute $\mathbf{y}_0 = \mathbf{v}^\top \mathbf{P}_{i,0}^{(1)} \mathbf{v}$ and $\mathbf{y}_1 = \mathbf{v}^\top \mathbf{P}_{i,1}^{(1)} \mathbf{v}$ for the two shares $\mathbf{P}_{i,0}^{(1)}$ and $\mathbf{P}_{i,1}^{(1)}$ and continue with the additive shares $\mathbf{y}_0$ and $\mathbf{y}_1$, which contribute to the constant part of the system of linear equations.
- *Line 10, solve $\boldsymbol{L}\boldsymbol{x} = \boldsymbol{t} - \boldsymbol{y}$ for $\boldsymbol{x}$:* At this point we already arrive with two shares $\mathbf{y}_0$ and $\mathbf{y}_1$ of $\mathbf{y}$. Since we do not want an attacker to get track of the value $\mathbf{y}$, we suggest to continue with these two shares and compute two solutions $\mathbf{x}_0$ and $\mathbf{x}_1$ of the linear systems $\mathbf{L}\mathbf{x}_0 = \mathbf{t} - \mathbf{y}_0$ and $\mathbf{L}\mathbf{x}_1 = -\mathbf{y}_1$. Their sum $\mathbf{x} = \mathbf{x}_0 + \mathbf{x}_1$ gives us the coefficients of the oil vector, since $\mathbf{L}\mathbf{x} = \mathbf{L}(\mathbf{x}_0 + \mathbf{x}_1) = \mathbf{L}\mathbf{x}_0 + \mathbf{L}\mathbf{x}_1 = \mathbf{t} - \mathbf{y}_0 - \mathbf{y}_1 = \mathbf{t} - \mathbf{y}$ just like in the original implementation.
- *Line 11, add together vinegar and oil vector:* First, we compute two shares $\mathbf{o}_0$ and $\mathbf{o}_1$ of the oil vector. To this end, we split the oil space randomly into two additive shares $\mathbf{O}_0$ and $\mathbf{O}_1$. Now, we can compute $\mathbf{o}_0 = \mathbf{O}_0\mathbf{x}$ and $\mathbf{o}_1 = \mathbf{O}_1\mathbf{x}$. We do not need to work with the shares of $\mathbf{x}$ anymore, since $\mathbf{x}$ itself is leaked as part of the signature to the public anyway. Finally, to receive the signature $\mathbf{s}$, we add up all the shares $\mathbf{s} = \mathbf{v}_0 + \mathbf{v}_1 + \mathbf{o}_0 + \mathbf{o}_1$.

We implemented these countermeasures as described above and measured their overhead. The practical results are presented in Sect. 6. As one could expect, masking the functions in Line 7 and 9 is responsible for the majority of the total overhead induced by masking UOV. These functions are quite expensive

themselves and the vast amount of randombytes that is required to generate the shares of the involved matrices also contributes significantly to the increased number of clock cycles, as detailed in Sect. 6.

6 Practical Results

This section firstly presents the performance evaluation of our protected implementations for UOV and MAYO. As there is currently no Cortex-M4 implementation of QR-UOV and SNOVA publicly available, the practical part of this work focus on UOV and MAYO. In this section, we further provide experimental results in terms of side-channel resistance. For all evaluation purposes, we use existing, unprotected implementations for NIST security level I of both schemes as a basis. More precisely, our implementations are based on the respective optimized UOV (`ov-Ip-pkc/m4f`) and MAYO (`mayo1/m4f`) Cortex-M4 implementation available within the pqm4 [28] library.

To increase compatibility, all changes were applied within the respective signing function itself, i.e. the signature of the function remains unchanged. Note that our findings (cf. Sect. 5) and implemented measures can be easily applied to other parameter sets.

6.1 Performance Results

In this section we present some performance figures of our first-order masked implementation of UOV and MAYO. For benchmarking, we target the ST NUCLEO-L4R5ZI board featuring an Arm Cortex-M4F core with 640 KB of RAM and 2 MB of flash memory. All randomness required for masking is generated using the internal hardware random number generator available on that board. We used the `arm-none-eabi-gcc` compiler (version 13.3.1) with the compiler flags `-O3 -std=gnu99 -mthumb -mcpu=cortex-m4 -mfloat-abi=hard -mfpu=fpv4-sp-d16` for compilation.

For the targeted parameter set (`ov-Ip`) of UOV, the combined size of the expanded secret key and the expanded public key is 516 KB. Due to the 640 KB of RAM, both expanded keys fit into the RAM. However, in order to obtain the required space for masking within the 640 KB of RAM on the ST NUCLEO-L4R5ZI board, we applied the approach [13,15] of writing the keys to flash memory. Table 3 presents the memory requirements of our protected implementations compared to the existing versions. In the case of UOV, it shows 1) the increase of stack usage during the key generation due to having to cache the keys in RAM before writing them to the flash memory, 2) the additional stack usage when signing due to our implemented masking measures, and 3) the increase of code size required for masking. Whereas in the case of MAYO, the additional memory requirement is significantly lower, as the size of the RAM is sufficient for masking without writing the keys to the flash memory. This is due to having to cache the keys in RAM before writing them to flash.

Table 3. Memory requirements for each implementation. Code, data and BSS size listed are in bytes, stack usage in 2^{10} byte (i.e., KiB).

Scheme	Impl.	Library size			Stack usage		
		Code	Data	BSS	keygen	sign	verify
`ov-Ip-pkc`	m4f	80 006	0	0	15.2	5.1	2.5
	m4f-flash	80 062	0	0	401.6	5.1	2.5
	masked-m4f-flash	213 076	0	0	401.6	264.4	2.5
MAYO_1	m4f	16 513	8	0	72.7	110.8	430.3
	masked-m4f	17 630	8	0	72.7	217.2	430.3

Table 4 compares the performance of protected and unprotected versions of UOV. Thereby, we differentiate between certain subroutines (cf. Sect. 5) to clarify the cost of each measure. In addition to the masked implementation, we implemented blinding as an alternative protecting method for two suitable and most costly subroutines based on the following approach. The functions $\mathbf{v}^\top \mathbf{S}_i$ and $\mathbf{v}^\top \mathbf{P}_i \mathbf{v}$ are linear with respect to the used matrices, so blinding works in a straightforward way. We multiply them with random values $r_i \in \mathbb{F}_q \backslash \{0\}$ beforehand and nullify its effect by multiplying the result with r_i^{-1}. Note that we use different random values r_i for each of the matrices $\mathbf{S}_i$ and $\mathbf{P}_i$ for $i \in \{1, \ldots, m\}$.

This approach is less powerful than masking every single entry of these matrices, but it still ensures that the values in $\mathbf{S}_i$ do not remain identical over various signing procedures and the values in $\mathbf{P}_i$ are not open to public anymore. Table 4 shows that 1) the masked version is about 5× slower than the unprotected implementation and 2) blinding is in total almost 2× slower than masking.

Table 4. Cortex-M4F cycle counts for our protected implementations in comparison to the optimized unprotected implementation of `ov-Ip-pkc`. Note that the implementation with blinding only differs from the masked implementation in two subroutines.

Pseudo code	Subroutine	UOV unprotected	[this work] masking	[this work] with blinding
Line 4	Sample vinegar vectors SHAKE256	13 455	132 363	132 363
Line 7	Linear part of system $L = v^\top \cdot S_i$	1 083 775	6 816 989	12 069 951
Line 9	Constant part of system $y = v^\top \cdot P_i \cdot v$	903 390	3 721 735	8 359 110
Line 10	Solve linear system Solve $Lx = t - y$ for x	435 349	872 866	872 866
Line 11	Add oil and vinegar $s = v + Ox$	26 633	109 454	109 454
CM of Sec. 5.4	Modify vinegar after usage $\mathrm{v} = \mathrm{v} + \mathrm{r}$	-	768	768
Total cycle counts for signing		2 478 708	11 840 264	21 916 475

Table 5 compares the performance of protected and unprotected versions of MAYO. Similar to UOV, we present the performance results of each implemented subroutine. The present figures show that the overhead of masking is in total smaller than 2×. Although we followed the same approach for both schemes, the slowdown for UOV is significantly larger compared to MAYO.

Table 5. Cortex-M4F cycle counts for various subroutines within the expanding and signing procedure in comparison to the unprotected implementation of MAYO_1.

Pseudo code	Subroutine	MAYO unprotected	[this work] masking
[11] `Alg.6 Line 17`	Secret key expansion $L_i = (P_i^{(1)} + P_i^{(1)T})O + P_i^{(2)}$	2 165 338	5 343 609
[11] `Alg.8 Line 16`	Sample vinegar vectors & randomizer SHAKE256	40 775	394 917
[11] `Alg.8 Line 27`	Linear part of system $M_i[j,:] = v_i^T L_j$	524 900	1 782 457
[11] `Alg.8 Line 30`	Constant part of system $u = v_i^T P_a^{(1)} v_i$	1 969 234	3 782 901
[11] `Alg.8 Line 38`	Solve linear system Solve $Ax = y$ for x	928 381	1 858 051
[11] `Alg.8 Line 45`	Add vinegar and oil terms $s_i = (v_i + Ox_i) \mid\mid x_i$	105 209	188 488
Total cycle counts for signing		9 122 185	16 783 809

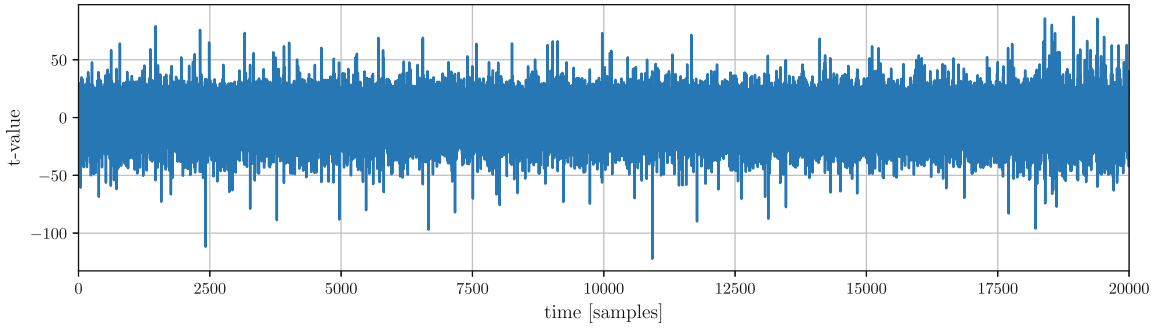
6.2 Side-Channel Evaluation

In this section, we present evaluation results for potential side-channel leakages of an unprotected compared to our masked implementation of UOV. All experiments regarding leakage evaluation were carried out using the ChipWhisperer tool chain [33,43] in Python (version 3.9.5) and performed on a ChipWhisperer-Lite board with an STM32F405 target board featuring an Arm Cortex-M4 core with 192 KB of RAM and 1 MB of flash memory.

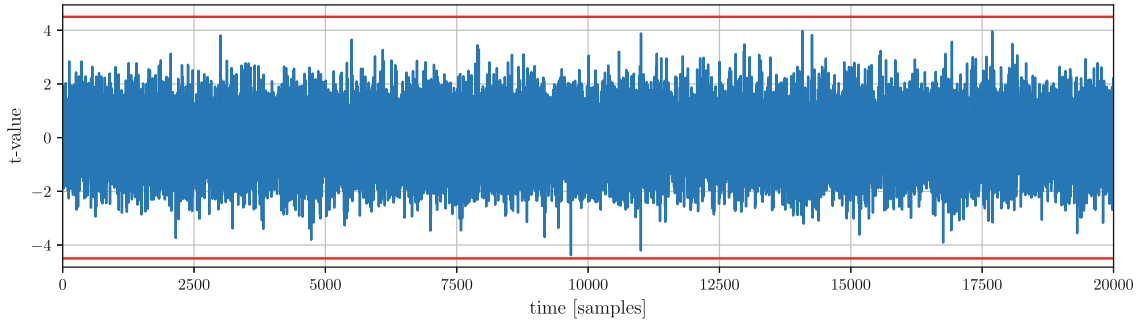
Since all vulnerable subroutines (cf. Table 4), with the exception of SHAKE, boil down to multiplications, we focus our efforts on the most costly function `gfmat_prod` which multiplies a vector $\mathbf{v}$ with matrices $\mathbf{S}_i$. Therefore, our implementation for side-channel evaluation only provides the `gfmat_prod` function and all required subroutines for generating traces. For leakage evaluation, we applied the commonly used Welch's t-test methodology [40]. More precisely, we used the fixed vs. random (FvR) approach. Thereby, we multiply a random vinegar vector with fixed matrices or random matrices, resp. In this case, these matrices represent part of the (extended) secret key.

As shown in Fig. 4a, the unmasked implementation is highly leaking by presenting very high t-values in the range of about $(100, -100)$, confirming the

threat induced by the leakages. In contrast, the t-values for the masked implementation depicted in Fig. 4b are all in the required range of $(-4.5, 4.5)$.



(a) Unmasked



(b) Masked

Fig. 4. Evaluation of the t-test after 10,000 traces for 20 000 samples traced during the computation of the `gfmat_prod` function. The range of the t-values is significantly lower for the masked (b) than for the unmasked version (a). The red lines in (b) indicate the threshold for side-channel leakage. (Color figure online)

7 Conclusion

In this work we conducted an extensive literature review of all existing physical attacks on UOV-based signature schemes and identified further attack vectors. Since all analyzed schemes share a large amount of operations that are contributed to the oil-and-vinegar principle, the theoretical idea behind the attacks transfers really well across the schemes, both for side-channel and fault attacks. Even the utilization of the quotient ring structure in QR-UOV and SNOVA had no impact on the transferability. The technical realization, however, depends highly on the given implementation.

We conclude that certain implementation choices, namely the utilization of compressed keys and employing a randomized signing process, has a positive impact on the resistance against fault attacks. The remaining fault attacks can be covered by dedicated countermeasures, with only a small overhead. Note, that we only covered first-order fault attacks in this work.

However, we see a greater risk with regard to side-channel attacks. In every scheme we analyzed, sensitive values are multiplied with huge amounts of public data, which represents a major gateway for power analysis methods. This observation is confirmed by our TVLA conducted on unprotected multiplication routines of UOV. To this end, we present a first-order masked version of both UOV and MAYO on the basis of their existing optimized Cortex-M4 implementations available within the pqm4 library. The results are supposed to serve as a first assessment of the overhead one can expect when applying masking countermeasures to UOV-based schemes. We observed that the produced overhead is smaller for MAYO than for UOV and identified two reasons for that. First, the amount of random bytes that are necessary to split the matrices into two shares is considerably smaller in MAYO, due to smaller parameters. Second, there are some subroutines in MAYO, i.e., the multiplication with public emulsifier maps and the accumulation of these products, where we concluded masking is not required. Thus, the share of sensitive operations is a little higher in UOV than in MAYO.

References

1. FIPS PUB 202: SHA-3 standard: Permutation-based hash and extendable-output functions. Federal Information Processing Standards Publication 202. National Institute of Standards and Technology, U.S. Department of Commerce (2015)
2. Aulbach, T., Campos, F., Krämer, J., Samardjiska, S., Stöttinger, M.: Separating oil and vinegar with a single trace side-channel assisted Kipnis-Shamir attack on UOV. IACR Trans. Cryptogr. Hardw. Embed. Syst. (2023). https://doi.org/10.46586/tches.v2023.i3.221-245
3. Aulbach, T., Kovats, T., Krämer, J., Marzougui, S.: Recovering rainbow's secret key with a first-order fault attack. In: Progress in Cryptology - AFRICACRYPT 2022: 13th International Conference on Cryptology in Africa. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-17433-9_15
4. Aulbach, T., Marzougui, S., Seifert, J.-P., Ulitzsch, V.Q.: Mayo or may-not: exploring implementation security of the post-quantum signature scheme MAYO against physical attacks. In: Workshop on Fault Detection and Tolerance in Cryptography, FDTC 2024. IEEE (2024). https://doi.org/10.1109/FDTC64268.2024.00012
5. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Building power analysis resistant implementations of Keccak. In: Second SHA-3 Candidate Conference (2010). https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=fe3d80a12e34d67ce14d438935302c6ef371901c
6. Beullens, W.: Improved cryptanalysis of UOV and rainbow. In: Canteaut, A., Standaert, F.-X. (eds.) EUROCRYPT 2021. LNCS, vol. 12696, pp. 348–373. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77870-5_13
7. Beullens, W.: MAYO: practical post-quantum signatures from oil-and-vinegar maps. In: AlTawy, R., Hülsing, A. (eds.) SAC 2021. LNCS, vol. 13203, pp. 355–376. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-99277-4_17
8. Beullens, W.: Breaking rainbow takes a weekend on a laptop. In: Advances in Cryptology - CRYPTO 2022. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-15979-4_16

9. Beullens, W.: Improved cryptanalysis of SNOVA. IACR Cryptol. ePrint Arch. (2024). https://eprint.iacr.org/2024/1297
10. Beullens, W., Campos, F., Celi, S., Hess, B., Kannwischer, M.J.: Nibbling MAYO: optimized implementations for AVX2 and cortex-m4. IACR Trans. Cryptogr. Hardw. Embed. Syst. (2024). https://doi.org/10.46586/tches.v2024.i2.252-275
11. Beullens, W., Campos, F., Celi, S., Hess, B., Kannwischer, M.J.: MAYO. Technical report, National Institute of Standards and Technology (2023). https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures
12. Beullens, W., et al.: UOV. Technical report, National Institute of Standards and Technology (2023). https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures
13. Beullens, W., et al.: Oil and vinegar: modern parameters and implementations. IACR Trans. Cryptogr. Hardw. Embed. Syst. (2023). https://doi.org/10.46586/tches.v2023.i3.321-365
14. Blömer, J., Da Silva, R.G., Günther, P., Krämer, J., Seifert, J.P.: A practical second-order fault attack against a real-world pairing implementation. In: 2014 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2014. IEEE Computer Society (2014). https://doi.org/10.1109/FDTC.2014.22
15. Chen, M.-S., Chou, T.: Classic McEliece on the ARM cortex-m4. IACR Trans. Cryptogr. Hardw. Embed. Syst. (2021). https://doi.org/10.46586/tches.v2021.i3.125-148
16. Clavier, C., Reynaud, L.: Improved blind side-channel analysis by exploitation of joint distributions of leakages. In: Fischer, W., Homma, N. (eds.) CHES 2017. LNCS, vol. 10529, pp. 24–44. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66787-4_2
17. Ding, J., et al.: Rainbow. Technical report, National Institute of Standards and Technology (2020). https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions
18. Ding, J., Deaton, J., Vishakha, Yang, B.-Y.: The nested subset differential attack - a practical direct attack against LUOV which forges a signature within 210 minutes. In: Advances in Cryptology - EUROCRYPT 2021. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77870-5_12
19. Ding, J., et al.: TUOV. Technical report, National Institute of Standards and Technology (2023). https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures
20. Furue, H., et al.: QR-UOV. Technical report, National Institute of Standards and Technology (2023). https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures
21. Furue, H., Kiyomura, Y., Nagasawa, T., Takagi, T.: A new fault attack on UOV multivariate signature scheme. In: Post-Quantum Cryptography - 13th International Workshop, PQCrypto 2022. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-17234-2_7
22. Gierlichs, B., Schmidt, J.-M., Tunstall, M.: Infective computation and dummy rounds: fault protection for block ciphers without check-before-output. In: Hevia, A., Neven, G. (eds.) LATINCRYPT 2012. LNCS, vol. 7533, pp. 305–321. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33481-8_17
23. Goubin, L., et al.: PROV. Technical report, National Institute of Standards and Technology (2023). https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures

24. Hashimoto, Y., Takagi, T., Sakurai, K.: General fault attacks on multivariate public key cryptosystems. In: Yang, B.-Y. (ed.) PQCrypto 2011. LNCS, vol. 7071, pp. 1–18. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25405-5_1
25. Jendral, S., Dubrova, E.: MAYO key recovery by fixing vinegar seeds. IACR Cryptol. ePrint Arch. (2024). https://eprint.iacr.org/2024/1550
26. Jendral, S., Dubrova, E.: Single-trace side-channel attacks on MAYO exploiting leaky modular multiplication. IACR Cryptol. ePrint Arch. (2024). https://eprint.iacr.org/2024/1850
27. Jendral, S., Mattsson, J.P., Dubrova, E.: A single-trace fault injection attack on hedged module lattice digital signature algorithm (ML-DSA). In: Workshop on Fault Detection and Tolerance in Cryptography, FDTC 2024. IEEE (2024). https://doi.org/10.1109/FDTC64268.2024.00013
28. Kannwischer, M.J., Rijneveld, J., Schwabe, P., Stoffelen, K.: pqm4: testing and benchmarking NIST PQC on ARM cortex-m4. IACR Cryptol. ePrint Arch. (2019). https://eprint.iacr.org/2019/844
29. Kipnis, A., Patarin, J., Goubin, L.: Unbalanced oil and vinegar signature schemes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 206–222. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_15
30. Koo, N., Shim, K.-A.: Security analysis of reusing vinegar values in UOV signature scheme. IEEE Access (2024). https://doi.org/10.1109/ACCESS.2024.3409778
31. Krämer, J., Loiero, M.: Fault attacks on UOV and rainbow. In: Polian, I., Stöttinger, M. (eds.) COSADE 2019. LNCS, vol. 11421, pp. 193–214. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-16350-1_11
32. Mus, K., Islam, S., Sunar, B.: Quantumhammer: a practical hybrid attack on the LUOV signature scheme. In: CCS 2020: 2020 ACM SIGSAC Conference on Computer and Communication Security. ACM (2020). https://doi.org/10.1145/3372297.3417272
33. O'Flynn, C., Chen, Z.D.: ChipWhisperer: an open-source platform for hardware embedded security research. In: Prouff, E. (ed.) COSADE 2014. LNCS, vol. 8622, pp. 243–260. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10175-0_17
34. Park, A., Shim, K.-A., Koo, N., Han, D.-G.: Side-channel attacks on post-quantum signature schemes based on multivariate quadratic equations - rainbow and UOV. IACR Trans. Cryptogr. Hardw. Embed. Syst. (2018). https://doi.org/10.13154/tches.v2018.i3.500-523
35. Patarin, J., et al.: VOX. Technical report, National Institute of Standards and Technology (2023). https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures
36. Pébereau, P.: One vector to rule them all: Key recovery from one vector in UOV schemes. In: Post-Quantum Cryptography - 15th International Workshop, PQCrypto 2024. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-62746-0_5
37. Petzoldt, A., Thomae, E., Bulygin, S., Wolf, C.: Small public keys and fast verification for $\mathcal{M}$ultivariate $\mathcal{Q}$uadratic public key systems. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 475–490. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23951-9_31
38. Ravi, P., Jap, D., Bhasin, S., Chattopadhyay, A.: Invited paper: machine learning based blind side-channel attacks on PQC-based KEMs - a case study of kyber KEM. In: IEEE/ACM International Conference on Computer Aided Design, ICCAD 2023. IEEE (2023). https://doi.org/10.1109/ICCAD57390.2023.10323721

39. Sayari, O., Marzougui, S., Aulbach, T., Krämer, J., Seifert, J.-P.: HAMAYO: a fault-tolerant reconfigurable hardware implementation of the MAYO signature scheme. In: Constructive Side-Channel Analysis and Secure Design - 15th International Workshop, COSADE 2024. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-57543-3_13
40. Schneider, T., Moradi, A.: Leakage assessment methodology. In: Güneysu, T., Handschuh, H. (eds.) CHES 2015. LNCS, vol. 9293, pp. 495–513. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48324-4_25
41. Shim, K.-A., Koo, N.: Algebraic fault analysis of UOV and rainbow with the leakage of random vinegar values. IEEE Trans. Inf. Forensics Secur. (2020). https://doi.org/10.1109/TIFS.2020.2969555
42. Tao, C., Petzoldt, A., Ding, J.: Efficient key recovery for all HFE signature variants. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021. LNCS, vol. 12825, pp. 70–93. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-84242-0_4
43. NewAE Technology. Repository of ChipWhisperer tool chain - commit a9527b5 (2023). https://github.com/newaetech/chipwhisperer
44. Wang, L.-C., et al.: SNOVA. Technical report, National Institute of Standards and Technology (2023). https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures

HaMAYO: A Fault-Tolerant Reconfigurable Hardware Implementation of the MAYO Signature Scheme

Oussama Sayari[1( )], Soundes Marzougui[1,2( )], Thomas Aulbach[3( )], Juliane Krämer[3], and Jean-Pierre Seifert[1,4]

[1] Technical University of Berlin, Berlin, Germany
oussama_sayari@yahoo.fr, jean-pierre.seifert@tu-berlin.de
[2] STMicroelectronics, Diegem, Belgium
soundes.marzougui@st.com
[3] University of Regensburg, Regensburg, Germany
{thomas.aulbach,juliane.kraemer}@ur.de
[4] Fraunhofer Institute SIT, Darmstadt, Germany

Abstract. MAYO is a topical modification of the established multivariate signature scheme UOV. Signer and Verifier locally enlarge the public key map, such that the dimension of the oil space and therefore, the parameter sizes in general, can be reduced. This significantly reduces the public key size while maintaining the appealing properties of UOV, like short signatures and fast verification. Therefore, MAYO is considered as an attractive candidate in the NIST call for additional digital signatures and might be an adequate solution for real-world deployment in resource-constrained devices.

When emerging to hardware implementation of multivariate schemes and specifically MAYO, different challenges are faced, namely resource utilization, which scales up with higher parameter sets. To accommodate this, we introduce a configurable hardware implementation designed for integration across various FPGA architectures. Our approach features adaptable configurations aligned with NIST-defined security levels and incorporates resources optimization modules. Our implementation is specifically tested on the Zynq ZedBoard with the Zynq-7020 SoC, with performance evaluations and comparisons made against previous hardware implementations of multivariate schemes.

Furthermore, we conducted a security analysis of the MAYO implementation highlighting potential physical attacks and implemented lightweight countermeasures.

Keywords: MAYO · Multivariate Cryptography · Post-Quantum Cryptography · Digital Signature · Hardware Implementation · Physical Security

1 Introduction

As quantum computing continues to advance, it is anticipated that quantum attacks can break many of the computational problems that classical

R. Wacquez and N. Homma (Eds.): COSADE 2024, LNCS 14595, pp. 240–259, 2024.
https://doi.org/10.1007/978-3-031-57543-3_13

cryptography relies on, such as factorization and discrete logarithms used in RSA and ECDSA, respectively. To address this, researchers have proposed new mathematical assumptions and computational problems that are difficult to solve with quantum computers, resulting in the field of post-quantum cryptography. These new assumptions are grouped into different families, such as lattice-based, code-based, hash-based, and multivariate cryptography.

Multivariate schemes mainly rely on the difficulty of solving large systems of multivariate quadratic equations, known as the MQ Problem. As such, the signature scheme Rainbow [DS05] was a finalist in the third round of the NIST post-quantum cryptography (PQC) Standardization Process. Rainbow is a two-layered version of the UOV signature scheme [KPG99]. Hence, multivariate signature schemes based on the oil and vinegar principle received a lot of attention. They offer very short signatures and efficient verification, since the signature is mainly the solution to a system of multivariate quadratic equations, and verifying boils down to evaluating the polynomials at the presumed solution. Still, during the third round, Beullens developed an algebraic attack on Rainbow [Beu22a], targeting the layer structure that differentiates Rainbow from UOV. This led to the elimination of Rainbow from the ongoing process since it lost all its alleged advantages over the base scheme UOV.

Since mainly lattice-based signatures remained in the competition, NIST called for the submission of additional post-quantum digital signature schemes to enhance the given variety of signatures by prioritizing those that are not reliant on structured lattices, have short signatures and fast verification. The majority of the multivariate schemes submitted to this process are based on the oil and vinegar principle.

MAYO, introduced in [Beu22b], is one of them. It uses the same trapdoor - a secret oil space that is annihilated by the public key map - but is developed such that the signer and the verifier locally enlarge the public key matrices. Therefore, the dimension of the oil space can be reduced. That also allows to reduce other parameters like the number of variables in the quadratic equations since certain algebraic attacks get harder with a smaller oil space [KS06]. In total, this leads to significantly smaller public keys in MAYO, while keeping good performance numbers and signature sizes. For instance, with parameters targeting the first security level of the NIST process, the public key size of MAYO is 1,168 bytes, the secret key is 24 bytes, and the signature size is 321 bytes [BCC+23]. These results make the MAYO signature scheme even more compact than state-of-the-art lattice-based signature schemes such as Falcon and Dilithium [PQD23].

Contribution. In this paper, we present an open source pure hardware implementation of the multivariate signature scheme MAYO. Our main target was a trade-off between SRAM/BRAM Consumption and FPGA Slides. In a second part, we investigated the physical security of MAYO implementation against side-channel analysis and fault-injection attacks. We, moreover, suggest lightweight countermeasures and implement them.

The contribution is summarized as follows:

- We manually settle a pure hardware implementation of MAYO. Our implementation is reconfigurable and can be easily integrated with different FPGA architectures and for different security levels.
- Certain functionalities used within key generation and signing are optimized, with a focus on low memory consumption.
- We present a new approach for the Gaussian solver and compare it to the well-known GSMITH approach of Rupp et al. in [REBG11].
- We considered threats emerging from possible fault injection and side channel analysis attacks, and cover them by employing low cost countermeasures.

The source code is available upon request.

Deployed Parameter Set. When we started with the hardware implementation, there was only one proof of concept implementation available on https://github.com/WardBeullens/MAYO and it used the parameter set $(n = 62, m = 60, o = 6, k = 10, q = 31)$ (see also [Beu22b, Section 8]). Thus, we also deployed these parameters in our work. In the meantime, the parameters were updated and as a main difference, MAYO also works over a field with even characteristic now, i.e., $q = 16$. This allows for higher efficiency and further implementation tricks, since now one field element occupies 4 bits instead of 5, and consequently, 2 field elements can be stored in one byte. The other parameters were also updated, but with minor impact. Thus, our work is one of the very few implementations of a multivariate schemes that utilizes a finite field with odd characteristic.

Related Work. At the time of writing this paper, there is a scarcity of complete hardware designs for post-quantum cryptographic schemes [ZZW+21, XL21, FG18, HZ18]. However, given that the NIST PQC reached the fourth round and started the call for additional digital signature schemes, it is expected that more dedicated hardware designs will emerge. These designs would be instrumental in showcasing the strength and inherent properties of specific protocols [NIS23a].

Multivariate schemes necessitate the development of comprehensive and extensive implementation designs to address the challenging gaps due to the schemes' large key sizes [DS05, KPG99]. These key sizes often pose challenges for devices with limited resources, as they may struggle to accommodate the storage requirements of these schemes. Moreover, multivariate schemes commonly involve memory and time-consuming blocks, with the Gaussian solver being a well-known performance bottleneck [REBG11]. Despite the above-mentioned challenges, there have been a few published hardware implementations that have reported results for multivariate schemes [TYD+11, HZ18, FG18].

In [FG18], Ferozpuri and Gaj present a high-speed FPGA implementation of Rainbow. Their hardware implementation uses a parameterized system solver where the execution time is proportional to the system dimension, i.e., it can solve an n-by-n system in n clock cycles. Moreover, their work reduces the number of required multipliers by almost half, speeds up execution as compared to

the previous state-of-the-art work, and implements Rainbow for higher security levels.

In [TYD+11], Tang et al. present another high-speed hardware implementation of Rainbow. The authors targeted similar functionalities for optimization as in [FG18], i.e., the Gaussian solver and the multipliers. They developed a new parallel hardware design for the Gaussian elimination and designed a novel multiplier to speed up the multiplication of three elements over a finite field. With Rainbow being broken [Beu22a], all its previously published software and hardware implementations needs to be revised and transferred to secure schemes for practical use. To address this issue, MAYO is seen as a viable alternative, showcasing improved performance results.

Simultaneous Work. During the preparation of this paper, hardware implementations of UOV [BCH+23] and MAYO [HSMR23] were published in 2023. The latter already features the updated parameter set of MAYO ($n = 66, m = 64, o = 8, k = 9, q = 16$), where m is chosen to be a multiple of 32 and q is a power of 2 to facilitate further implementation optimizations.

2 Preliminaries

The MAYO signature scheme [Beu22b] is a special modification of the UOV signature scheme [KPG99] and belongs to the field of multivariate cryptography. Herein, the main object is the multivariate quadratic map $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ with m components and n variables. In more detail, it is a sequence $p_1(\mathbf{x}), \ldots, p_m(\mathbf{x})$ of m quadratic polynomials in n variables $\mathbf{x} = (x_1, \ldots, x_n)$, with coefficients in a finite field $\mathbb{F}_q$. Very abbreviated, multivariate cryptography is based on the hardness of finding a preimage $\mathbf{s} \in \mathbb{F}_q^n$ of a target vector $\mathbf{t} \in \mathbb{F}_q^m$ under a given multivariate quadratic map $\mathcal{P}$, i.e., solving a multivariate system of quadratic equations. This task is often referred to as the MQ problem. One way that allows the signer to compute a signature $\mathbf{s}$ is to install a secret trapdoor into the public map $\mathcal{P}$.

2.1 The Trapdoor in UOV

In UOV, the trapdoor information is a basis of a secret linear subspace $\mathcal{O} \subset \mathbb{F}_q^n$ of dimension $\dim(\mathcal{O}) = m$, the so-called oil space [Beu21]. The multivariate quadratic map $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ is then chosen in a way that it vanishes on this oil space, i.e., $\mathcal{P}(\mathbf{o}) = \mathbf{0}_m$ for all $\mathbf{o} \in \mathcal{O}$. For the multivariate quadratic polynomials $p_i(\mathbf{x})$, which constitute the map $\mathcal{P}$ via $\mathcal{P}(\mathbf{x}) = p_1(\mathbf{x}), \ldots, p_m(\mathbf{x})$, one can define their *polar form* or *differential* as

$$p_i'(\mathbf{x}, \mathbf{y}) \coloneqq p_i(\mathbf{x} + \mathbf{y}) - p_i(\mathbf{x}) - p_i(\mathbf{y}) + p_i(\mathbf{0}).$$

Since we commonly work with homogeneous polynomials, the term $p_i(\mathbf{0})$ will be omitted in the following. Similarly, we can define the polar form of $\mathcal{P}$ as

$$\mathcal{P}'(\mathbf{x}, \mathbf{y}) = p_1'(\mathbf{x}, \mathbf{y}), \ldots, p_m'(\mathbf{x}, \mathbf{y}).$$

As shown in [Beu21, Theorem 1], the map $\mathcal{P}' : \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q^m$ is a symmetric and bilinear map. Furthermore, if one has knowledge of the secret oil space, it can be used to efficiently find preimages $\mathbf{x} \in \mathbb{F}_q^n$ of a given target $\mathbf{t} \in \mathbb{F}_q^m$ such that $\mathcal{P}(\mathbf{x}) = \mathbf{t}$. To do so, one can randomly pick a vinegar vector $\mathbf{v} \in \mathbb{F}_q^n$ and solve the system $P(\mathbf{v} + \mathbf{o}) = \mathbf{t}$ for $\mathbf{o} \in \mathcal{O}$. This is possible since in

$$\mathbf{t} = \mathcal{P}(\mathbf{v} + \mathbf{o}) = \mathcal{P}(\mathbf{v}) + \mathcal{P}(\mathbf{o}) + \mathcal{P}'(\mathbf{v}, \mathbf{o}) \tag{1}$$

the term $\mathcal{P}(\mathbf{v})$ is constant and $\mathcal{P}(\mathbf{o})$ vanishes, so whenever the linear map $\mathcal{P}'(\mathbf{v}, \cdot)$ is non-singular, the system has a unique solution $\mathbf{o} \in \mathcal{O}$, which can be computed efficiently. This happens with probability roughly $\frac{q-1}{q}$. If this is not the case, one can simply pick a new value for $\mathbf{v}$ and try again. Without a description of the oil space $\mathcal{O}$, the term $\mathcal{P}(\mathbf{o})$ implies that Eq. 1 constitutes a system of quadratic equations, which remains hard to solve.

Building a signature scheme directly from this setting has one big disadvantage. The oil space needs to be as large as the image space of the multivariate quadratic map $\mathcal{P}$, i.e., $\dim \mathcal{O} = m$. To counter the Kipnis-Shamir attack [KS06], the parameter n needs to be sufficiently larger than m, with $n \approx 2,5m$ being used in all currently considered implementations. The parameter m itself needs to be of a certain size as well, to provide security against direct attacks or the intersection attack [Beu21]. This leads to key pairs of enormous size, which is considered the main drawback of multivariate signatures. Recently, Beullens developed the signature scheme MAYO to tackle this problem.

2.2 Description of MAYO

The essential modification is the downsizing of the dimension of the oil space to $\dim \mathcal{O} = o < m$. Actually, this oil space is now too small to sample signatures, since the system $\mathcal{P}(\mathbf{v} + \mathbf{o}) = \mathbf{t}$ given in Eq. 1 consists consequently of m linear equations in o variables and is unlikely to have any solutions. Thus, the approach taken in [Beu22b] is to stretch the public key map into a larger whipped map $\mathcal{P}^* : \mathbb{F}_q^{kn} \to \mathbb{F}_q^m$, such that it accepts k input vectors $\mathbf{x} \in \mathbb{F}_q^n$. This is realized by defining

$$\mathcal{P}^*(\mathbf{x}_1, ..., \mathbf{x}_k) := \sum_{i=1}^{k} \mathbf{E}_{ii}\mathcal{P}(\mathbf{x}_i) + \sum_{1 \le i < j \le k} \mathbf{E}_{ij}(\mathcal{P}'(\mathbf{x}_i, \mathbf{x}_j)), \tag{2}$$

where the matrices $\mathbf{E}_{ij} \in \mathbb{F}_q^{m \times m}$ are fixed system parameters with the property that all their non-trivial linear combinations have rank m.

It is easy to see that $\mathcal{P}^*$ vanishes on the subspace $\mathcal{O}^k = \{(\mathbf{o}_1, \ldots, \mathbf{o}_k) |$ with $\mathbf{o}_i \in \mathcal{O}$ for all $i \in [k]\}$ of dimension ko. By choosing the parameters such that $ko \ge m$, the k copies of the oil space are large enough to construct preimages of a target vector $\mathbf{t} \in \mathbb{F}_q^m$ under the whipped map $\mathcal{P}^*$. In more detail, the signer randomly samples $(\mathbf{v}_1, \ldots, \mathbf{v}_k) \in \mathbb{F}_q^{kn}$, and then solves

$$\mathcal{P}^*(\mathbf{v}_1 + \mathbf{o}_1, ..., \mathbf{v}_k + \mathbf{o}_k) = \mathbf{t} \tag{3}$$

for $(\mathbf{o}_1, ...\mathbf{o}_k) \in \mathcal{O}^k$. Observe from Eq. 2 that this system remains linear in the presence of the linear emulsifier maps $\mathbf{E}_{ij} \in \mathbb{F}_q^{m \times m}$. Thus, the signer can efficiently compute a preimage $\{\mathbf{s}_i = \mathbf{v}_i + \mathbf{o}_i\}_{i \in [k]}$ of $\mathbf{t}$. Similar to UOV, the verifier just needs to check if the given $\{\mathbf{s}_i\}_{i \in [k]}$ satisfy Eq. 3.

Remark 1. Please note that both, the signer and the verifier, only locally whip up the public key map $\mathcal{P}$ to $\mathcal{P}^*$, so this modification comes with no additional cost in terms of key sizes. However, it entails additional computations during signing and verification. Furthermore, it increases signature size, since now a k-tuple of vectors in $\mathbb{F}_q^n$ constitute the signature. These negative effects are cushioned by the ability to reduce parameter sizes while maintaining the security level.

2.3 The Implemented MAYO Functionalities

The above descriptions remain rather high-level and abstract. Here we show more details about the main functionalities that need to be implemented, e.g., evaluations of (parts of) the public key map $\mathcal{P}$ via vector-matrix multiplications and finding solutions to the generated linear system via Gaussian elimination. Due to the page limit we do not present all the algorithms we implemented here, but refer to the MAYO specification [BCC+23, Section 2], specifically to the algorithms *MAYO.CompactKeyGen()*, *MAYO.ExpandSK(csk)* and *MAYO.Sign(esk,M)*. The latter will also play a major role in our security discussion in Sect. 4, so it is presented in Algorithm 1 below. The first few lines are used to sort the bit string of the expanded secret key to the respective matrices (line 1–5) and to derive a target vector $\mathbf{t} \in \mathbb{F}_q^m$ and salt (line 7–11). The main part of the signing process can be described by generating random variables (line 15–19), inserting the vinegar variables $\mathbf{v}_i$ into $\mathcal{P}$ to set up a linear system (line 21–35), solving the system (line 37–40) and adding the solution to the vinegar variables (line 42–45).

3 Hardware Design

In this section, we present the hardware design of our implementation. Our primary goal is to provide a reconfigurable hardware code that can be easily integrated with different FPGA architectures and for different security levels.

Although MAYO has keys of reduced size compared to other multivariate alternatives, it still necessities a large amount of internal memory to execute the key-generation and signing phase [Beu22b] in the order of several dozen KB. This is partially attributed to the fact that the keys are stored as seeds. During the signing the seed is expanded into large matrices, e.g., for the parameter set $(n, m, o, k, q) = (66, 64, 8, 9, 16)$, the public key of 1168B is expanded into 70KB.

For implementation and testing of our hardware design, we opted for the target board Zynq ZedBoard with the Zynq-7020 SoC [Xil23], which has 85K Logic Cells and 4.9MB Block RAM serving as an upper bound for the memory consumption.

Algorithm 1. MAYO.Sign(esk,M) [BCC+23]

Input: Expanded secret key $\text{esk} \in \mathcal{B}^{\text{esk bytes}}$, Message $M \in \mathcal{B}^*$
Output: Signature $\text{sig} \in \mathcal{B}^{\text{sig bytes}}$

// Decode esk
$\text{seed}_{sk} \leftarrow \text{esk}[0 : \texttt{sk\_seed\_bytes}]$
$\mathbf{O} \leftarrow \texttt{Decode}_{\mathbf{O}}(\text{esk}[\texttt{sk\_seed\_bytes} : \texttt{sk\_seed\_bytes} + \mathbf{O}\texttt{\_bytes}])$
$\{\mathbf{P}_i^{(1)}\}_{i\in[m]} \leftarrow \texttt{Decode}_{P^{(1)}}(\text{esk}[\texttt{sk\_seed\_bytes} + \mathbf{O}\texttt{\_bytes}] : \texttt{sk\_seed\_bytes} + \mathbf{O}\texttt{\_bytes}] + \texttt{P1\_bytes}])$
$\{\mathbf{L}_i\}_{i\in[m]} \leftarrow \texttt{Decode}_{L}(\text{esk}[\texttt{sk\_seed\_bytes} + \mathbf{O}\texttt{\_bytes}] + \texttt{P1\_bytes} : \texttt{esk\_bytes}])$

// Hash message and derive salt and $\mathbf{t}$
$M_{digest} \leftarrow \texttt{SHAKE256}(M, \texttt{digest\_bytes})$
$\text{R} \leftarrow \mathbf{0}_{R_{bytes}}$
$\text{salt} \leftarrow \texttt{SHAKE256}(M_{digest} \,\|\, R \,\|\, \text{seed}_{sk}, \texttt{salt\_bytes})$
$\mathbf{t} \leftarrow \texttt{Decode}_{vec}(m, \texttt{SHAKE256}(M_{digest} \,\|\, \text{salt}, \lceil (m \log(q))/8 \rceil))$

// Attempt to find a preimage for $\mathbf{t}$
for ctr from 0 to 255 **do**
    # Derive $\mathbf{v}_i$ and r
    $V \leftarrow \texttt{SHAKE256}(M_{digest} \,\|\, \text{salt} \,\|\, \text{seed}_{sk} \,\|\, \text{ctr}, k \cdot v_{bytes} + \lceil ko \log(q)/8 \rceil)$
    for i from 0 to $k-1$ **do**
        $\mathbf{v}_i \leftarrow \texttt{Decode}_{vec}(n-o, V[i \cdot v_{bytes} : (i+1) \cdot v_{bytes}])$
    $\mathbf{r} \leftarrow \texttt{Decode}_{vec}(ko, V[k \cdot v_{bytes} : k \cdot v_{bytes} + \lceil ko \log(q)/8 \rceil])$

    // Build linear system $Ax = y$.
    $\mathbf{A} \leftarrow \mathbf{0}_{m \times ko} \in \mathbb{F}_q^{m \times ko}$
    $\mathbf{y} \leftarrow \mathbf{t}$, $\ell \leftarrow 0$
    for i from 0 to $k-1$ **do**
        $\mathbf{M}_i \leftarrow \mathbf{0}_{m \times o} \in \mathbb{F}_q^{m \times o}$
        for j from 0 to $m-1$ **do**
            $\mathbf{M}_i[j, :] \leftarrow \mathbf{v}_i^\intercal \mathbf{L}_j$
        for j from $k-1$ to i **do**
            $\mathbf{u} \leftarrow \{\mathbf{v}_i^\intercal \mathbf{P}_a^{(1)} \mathbf{v}_i\}_{a\in[m]}$ **if** $i = j$
            $\mathbf{u} \leftarrow \{\mathbf{v}_i^\intercal \mathbf{P}_a^{(1)} \mathbf{v}_j + \mathbf{v}_j^\intercal \mathbf{P}_a \mathbf{v}_i\}_{a\in[m]}$ **if** $i \neq j$
            $\mathbf{y} \leftarrow \mathbf{y} - \mathbf{E}^\ell \mathbf{u}$
            $\mathbf{A}[:, i \cdot o : (i+1) \cdot o] \leftarrow \mathbf{A}[:, i \cdot o : (i+1) \cdot o] + \mathbf{E}^\ell \mathbf{M}_j$
            if $i \neq j$ **then**
                $\mathbf{A}[:, j \cdot o : (j+1) \cdot o] \leftarrow \mathbf{A}[:, j \cdot o : (j+1) \cdot o] + \mathbf{E}^\ell \mathbf{M}_i$
            $\ell \leftarrow \ell + 1$

    // Try to solve the system
    $\mathbf{x} \leftarrow \texttt{SampleSolution}(\mathbf{A}, \mathbf{y}, \mathbf{r})$
    if $\mathbf{x} \neq \perp$ **then**
        break

// Finish and output the signature
$\mathbf{s} \leftarrow \mathbf{0}_{kn}$
for i from 0 to $k-1$ **do**
    $\mathbf{s}[i \cdot n : (i+1) \cdot n] \leftarrow (\mathbf{v}_i + \mathbf{O}\mathbf{x}[i \cdot o : (i+1) \cdot o] \,\|\, \mathbf{x}[i \cdot o : (i+1) \cdot o])$
return $\text{sig} = \texttt{Decode}_{vec}(s) \,\|\, \text{salt}$

The majority of the system architecture of our hardware design is described in VHDL, while a few modules are implemented using Verilog.

It is essential for the architecture to be encapsulated as an Intellectual Property (IP), to ensure design reuse. We developed Keygen and Sign IPs intended for use on an end-user device in diverse applications such as the authentication of bank transactions. It remains paramount that these two IPs guarantee compliance with the device's memory constraints, especially regarding time and memory utilization. In contrast, we expect that the verification process takes place within an environment boasting ample resources such as a dedicated server, where security measures are not as critical as those required for IPs operating directly on confidential data, i.e., Keygen and Sign.

It is possible to utilize one of the IPs on the target chip. Both cores are independent and capable of coexisting on the Programmable Logic operating at respectable frequencies.

The CPU-Peripheral communications between the built IPs are handled through AXI4-FULL, AXI-Lite, and interrupts. The provided firmware takes care of the AXI transactions, thanks to the Zynq hybrid architecture. Incidentally, the design focuses on maintaining high transfer bit-rates by extensively leveraging the CPU's 32-bit architecture. Frequencies and reset signals are also controlled by the hardcore and are propagated throughout the design.

Based on the proposed MAYO pseudo-code in [BCC+23], the scheme incorporates multiple helper functions that are implemented as sub-modules and arithmetic units within the hardware IPs. This approach fulfills another significant design requirement by minimizing unused module and minimizing the utilization of Flip-Flops (FFs) and Lookup Tables (LUTs). By avoiding code duplication in hardware and organizing the design into smaller, specialized modules, each capable of performing a single functionality, the overall efficiency and modularity of the design are improved.

Considering the scheme's parameter set, the memory is divided into *three* True Dual Port BRAMs, statically partitioned into 2×256KB BRAMs to store big matrices and large vectors like the $\mathcal{P}$ system and $\mathcal{O}^k$ subspaces, and 1×4KB BRAM designated for small scratch buffers and sensitive information such as the seed, signature, and secret key. Among these BRAMs, only one of the big BRAMs is exposed to CPU through the AXI bus. Detailed memory management and utilization is deliberated later in Sect. 3.4. As shown in Fig. 1, most modules are connected to the BRAMs accordingly.

3.1 Hash Function

Our design employs the Keccak core [BDH+22] to generate seeds and expand the message as a first step of the signing process. For the first security level, SHAKE128 was used as an extendable-output function (XOF) based on the FIPS 202 standard [NIS23c]. We note that for higher security levels, it is necessary to adjust the parameters within the Keccak core accordingly. Nonetheless, the fundamental design of the hash sub-module remains applicable and does not require significant changes.

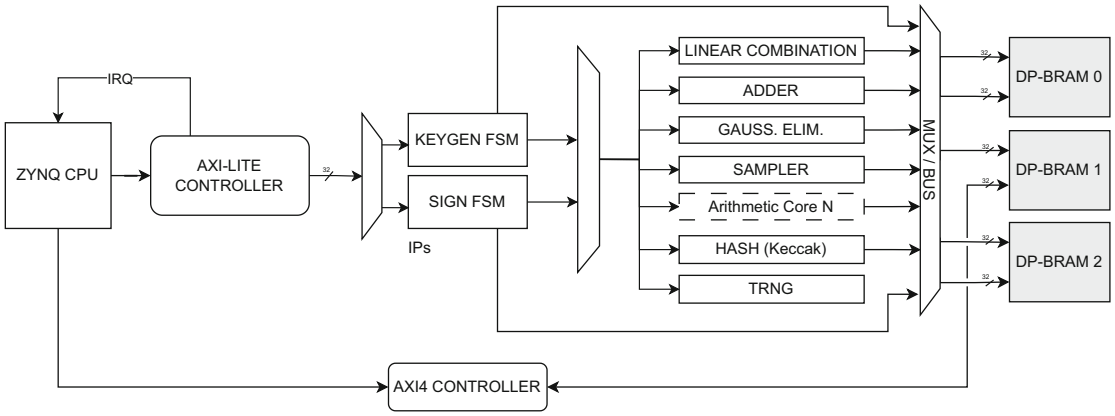


Fig. 1. Block Diagram of the MAYO Core

The Keccak implementation in [BDH+22] streams data utilizing a different format compared to the proposed MAYO hardware 32-bit format. To address this discrepancy, we developed a wrapper around the core. The reasoning behind this is that MAYO algorithm requires a hash of approximately 120KB for the key generation. The hash is eventually stored in the inner 32-bit-wide block memory.

The proposed architecture stores the input seed and output message in separate descriptor-like registers. These intermediate registers are simultaneously accessed by the hash core and BRAM. The core itself takes care of BRAM communication and indexing, simplifying the architecture's state change and its modularity.

3.2 Random Number Generator

The random number generator leverages AES-128 in CTR Cipher mode, with the flexibility to seamlessly switch to AES-256 if necessary. Tinkering with key parameters like seed and counter interval (PRNG-Based) is effortlessly accomplished within the core. To optimize FPGA Slice utilization, the core's decryption functionalities have been deprecated, given the inherent independence of CTR-mode from such operations.

3.3 Vector-Matrix Multiplication

Referring to Sect. 2.2, it is evident that matrix-vector multiplication proceeded by a $\mathbb{F}_q$ space reduction, is a frequently utilized operation throughout the algorithm. Hence, its optimization will improve the performance of our design.

Compared to the initial MAYO Software C implementation[1], the vector-matrix multiplication iterates through a matrix stored in a row-wise manner, as seen in the left side of Fig. 2, multiplying (using MULT operation) the content with a given series of coefficients and accumulating the results. Once this nested

[1] Note here that we refer to the first implementation of MAYO scheme by Ward Beullens in [Beu22b].

row/column loop concludes, another loop starts reducing the accumulated result through MOD operation. For instance, on an ARM Cortex-M3 with ARMv7-M instruction set, a single MULT operation with 8-bit operands takes around 2 to 3 clock cycles [ARM]. The reduction is done using the MOD operation that is usually translated to MULT and UDIV as Cortex-M3 lacks native modulo calculation. Consequently, the vector-matrix multiplication function could consume up to 6500 clock cycles, excluding the memory load and store operations.

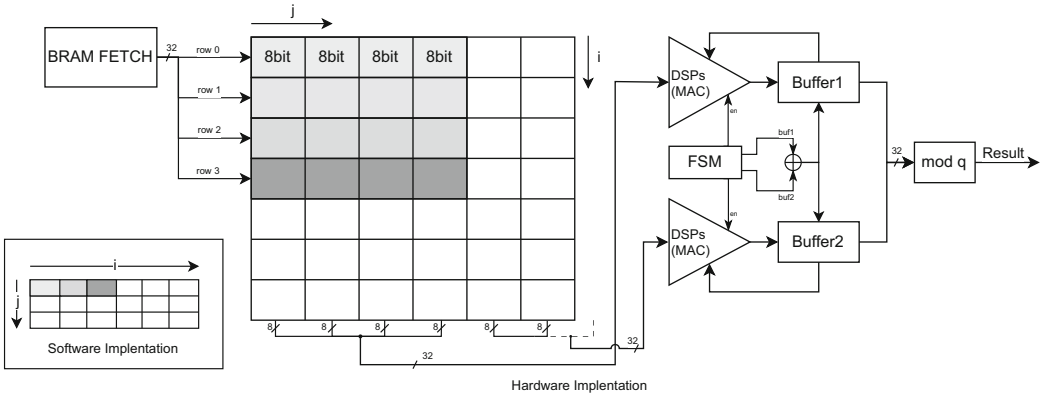


Fig. 2. Matrix-Vector multiplication architecture; on the left side the vector-matrix multiplication iterates through a matrix stored in a row-wise manner as in the software implementation. In hardware design, we reversed the indexing order, and input four bytes to each DSP which executes 4 multiplications simultaneously.

In this paper, we process the multiplications differently. Firstly, our design offers four values on each memory read operation thanks to its 32-bit wide bus and executes 4 MULT operations from one row simultaneously. Secondly, we reversed the indexing of the input matrix, as shown in Fig. 2.

As matrices are stored row-wise, each memory access returns $four$ sequential cells from *one* row. Note that the matrix is stored in BRAMs and not in an FF-layered structure.

Furthermore, the input of both Digital Signal Processors (DSPs) is composed of 4 bytes. This architecture helps increase the throughput and enables the parallelization of both MULT and MOD operations.

Once the accumulated data of a block of four columns begins the final MOD operation, the subsequent block is fetched and starts with MULT operation. The first row of the Matrix $\mathbf{M}$ and the first coefficient of the Vector $\mathbf{V}$ are fetched from the BRAMs. The read port then keeps feeding the system with blocks from each consequent row noted as $\mathbf{M}[rowIndex, columnBlock]$, until the accumulated result is ready to be stored through a different write-only port (WriteRES).

3.4 Memory Organization

The hardware implementation of MAYO mainly relies on BRAMs to store its vectors and matrices. To ensure that both cores, namely the KeyGen and Sign,

have sufficient stack-like memory, 82% (4.03 Mb) of the available on-chip BRAM is allocated for the implementation. Thereby we provide enough headroom for potential parameter modification of MAYO that might increase memory usage, e.g., when changing the security level from 1 to 5, the expanded secret key size increases from 70KB to 557KB [BCC+23].

The design aligns itself with the 32-bit ARM multi-core processor architecture and uses a 32-bit data bus width. This approach simplifies data processing within each sub-module. In the case of MAYO, the values are usually stored in a 5 bits-wide reduced space. For the NIST security level 1, the scheme operates on values that are eventually reduced to $\mathbb{F}_q$, meaning that the results must be less or equal to $q = 31$. To store such numbers in the BRAM, $5 = \lceil \log_2(31) \rceil$ bits are mandatory. As a result, the design allocates 8 bits of memory (i.e., unsigned char) for each numerical unit. We, then, exploit the 32-bit architecture in various pipeline techniques by processing simultaneously four 8-bit values.

It is important to note that our implementation adapts the parameter set $(n = 62, m = 60, o = 6, k = 10, q = 31)$ and resulted in a public key and signatures have a size of 803B and 420B respectively. However for the NIST first level the parameters are $(66, 64, 8, 9, 16)$ and result in public key and signature size of 926B and 387B.

There exist different variants of the MAYO first security level where the public key size is increased at the expense of smaller signature. Precisely, these variants increase the n which is the number of variables in the multivariate quadratic polynomials in the public key at the expense of decreasing k which is the whipping parameter. This results in bigger public key size and smaller signature size as the whipping parameters are directly connected to the calculation of the signature.

In addition, the q does not have significant impact on the sizes of the public key and the signature itself but more on the stack-like memory during the key generation and the singing processes. On the other hand, if $q = 16$, one byte can be used to pack two elements as all elements are in $\mathbf{F}_{16}$. However, this is not the case for our implementation.

It is important to note, that not *all* the allocated memory is utilized for the first security level. In fact, only roughly 70% (2.8 Mb) of the allocated BRAM of the Zynq device is filled with data. The rest is left empty, but deemed necessary due to ARM's 32-bit memory alignment rules. The content of the BRAM cells is pre-allocated and statically organized since the sizes of most elements are pre-defined. In other words, all vectors and matrices' addresses are provided in a VHDL file to create a mapping. This file is then included in all sub-modules for better consistency. To eliminate dependency on vendor-specific SDKs, a set of Python scripts takes charge of memory template generation. These scripts meticulously analyze the VHDL file, dynamically determining the required depth of BRAMs. This approach not only fosters platform independence but also enhances adaptability by allowing seamless adjustments to memory configurations based on the specifics of the VHDL code. The result is a more

agile and versatile solution for memory management within the FPGA design, especially for various parameter sets.

The memory is partitioned into *three* dual port BRAMs, offering enhanced performance and flexibility. This configuration allows, for instance, efficient reading from one port while dedicating the other port for writing. Some sub-modules, such as vector-matrix multiplication, or vector addition, tend to utilize three ports for dual read and final write operations, therefore allowing better opportunities for parallelism within the sub-module.

Small buffers and vectors that are not meant to be accessed exclusively by programmable hardware are found in the smaller BRAM. The big BRAMs are indeed also shared with the CPU through AXI bus to stream input information such as the message and secret key to the MAYO core itself. Furthermore, since the key generation and the signing are not designed to operate synchronously but rather consecutively, multiple arrays and vector spaces overlap if one of their lifetimes expires. This approach helps the system avoid unnecessary increases in memory fingerprints.

3.5 Gaussian Elimination

Solving a System of Linear Equations (SLE) is evidently one of the primordial computations for the MAYO algorithm to generate a valid message signature as explained in Sect. 2.2. Several publications deal with hardware implementation of Gaussian elimination for various cryptographic applications, primarily focusing on $\mathbb{F}_2$. Among them, GSMITH [REBG11] has been widely recognized for efficiently handling $\mathbb{F}_{2^k}$ equations. Unfortunately, GSMITH's architecture only conforms with small and medium-sized matrices, whereas MAYO's SLE $m \times m$ shaped matrix is larger. This quadratic shape depends on the NIST security level. Not only would the proposed GSMITH architecture utilize costly resources, but also hinder the overall architecture's performance and increase the needed Look-up Tables (LUTs) when targeting $\mathbb{F}_{31}$. GSMITH describes, in fact, a systolic network composed of various types of tiny processors capable of specific Gaussian steps and propagating its values. Yet, since the source code was not open-sourced, we had to redesign GSMITH. The final architecture, however, fails to meet our resource requirements, depleting the Zynq's FFs and LUTs, due to the internal registers required in each GSMITH processor and its interconnection with the proposed BRAM. When considering the other needed arithmetic cores, we concluded it was unfeasible to fit GSMITH for the first security level.

To overcome this issue, we developed a state machine that fetches values directly from BRAM as the matrix is stored externally rather than within the core's FFs. Additionally, it was mandatory to allocate sufficient memory to accumulate every cell in the matrix. In other words, during the first step of the Gaussian elimination, multiplying rows with scalars may surpass the existing 8-bit limit. Hence, the targeted matrix is initially unpacked into 16-bit wide values with added padding, meaning that every row in the BRAM now contains two instead of four values.

Moreover, to speed up the mod-inverse, which calculates the needed value to transform the pivot element into 1 throughout the first scale step, prefilled Read-Only Memory (ROM) with end results of this operation is utilized instead of performing the actual calculations on run-time. These optimizations contribute to the overall effectiveness of the MAYO core in solving an SLE. Although GSMITH might offer superior performance, this core certainly consumes less memory resources. Our architecture is theoretically compatible with other configuration sets, with a marginal difference in resource utilization. For instance, n, m, o control the SLE size which should affect BRAM consumption, while q modifies the LUT consumption, cell width, and the unpacking operation. The solver should support up to $\mathbb{F}_{2^8}$ and for smaller q values, unpacking the matrix might become unnecessary, as the result could still fit inside the original 8-bit vector.

3.6 Optimizations and Firmware

Besides resource utilization, the goal of our design is to achieve a reasonable time area trade-off. Therefore, we designed the sub-modules in a way that they share the same access to one of the BRAM ports. Nevertheless, the usage of each port, whether for reading, writing, or both differs. The core responsible for the vectors addition, for example, features multiple modes depending on the location of the input vectors in different BRAMs. It efficiently utilizes all available ports to leverage data throughput and synchronize the addition process accordingly.

Another notable design optimization lies in the polynomial reduction sub-module where multiple arrays of scratch buffers are used to minimize memory interactions. Hence, the core is provided only with new values which are stored as final results.

Various functionalities of MAYO are divided into separate modules, each described individually. That said, each module still has access to header-like files that declare the security level parameters, the memory space allocations, utility functions required to fetch offsets or even ROM secret keys specifically intended for non-debugging purposes. Numerous bit vectors are built upon these constants. The code's style guide itself heavily discourages simple number inclusion, but instead, it is expected to utilize these pre-defined macro-like lines to improve code readability and ensure that the overall architecture can fit different configuration sets, i.e., different security levels.

In addition to the hardware implementation, the utilization of the MAYO core necessitates the development of accompanying firmware. This firmware serves as the interface between the hardware core and the software MAYO application, setting AXI/AXI-Lite transactions up. The existing C Bit-fields feature Control and Status Registers that can enable debug mode, interrupts, and supply the ARM CPU with the end of executions information besides the interrupt signal.

4 Mitigation of Physical Attack Vectors in MAYO

In Sect. 2.2, we stated that the secret key is solely given by the secret linear oil space O. Thus, an attacker is able to forge signatures, as soon as she recovered O. Even more, the description of the reconciliation attack in [Beu22b, Section 4.1] shows that it is enough to know a single vector $o_1 \in O$, to recover the remaining space O in polynomial time, since the first vector o_1 implies m linear equations via $\mathcal{P}'(o_1, o_2)$ on the entries of o_2. Consequently, we need to solve m quadratic equations in $n-m-o$ variables. Since in MAYO $n < m+o$ holds, the remaining basis vectors of O can be obtained just by solving linear equations.

Moreover, the randomly generated vinegar variables can also be used to recover the secret key. Recall, that a given MAYO signature has the form $s = (s_1, \ldots, s_k) = (v_1 + o_1, \ldots, v_k + o_k)$, so the knowledge of one of the v_i's together with the corresponding s_i leads the attacker to a vector of the oil space and thus, to the full secret key.

In the following, we show different scenarios where the attacker uses fault injection or side-channel attacks to reveal either a vinegar or an oil vector.

4.1 Fault Injection

The attacks suggested in the following are first-order fault injection attacks and assume an attacker to be able to skip one specific instruction during the signing process. The resulting faulted signature is used to recover the secret key.

Skip Sampling of Vinegar Values (Re-using). The main idea here is to insert an instruction skip during the sampling of the vinegar variables. In Algorithm 1, this corresponds to a jump over line 18, for one (or more) of the $i \in 1, \ldots, k$. This fault injection attack forces the same vinegar variable $v_i \in \mathbb{F}_q^n$ to be used for two consecutive signatures of different messages m and m'. We subtract the obtained correct (not faulted) signature s and the faulted signature s' and receive $s - s' = (s_1 - s'_1, \ldots, s_k - s'_k)$. Observe that for the entry i, where $v_i = v'_i$ holds, we have

$$s_i - s'_i = v_i + o_i - v'_i - o'_i = o_i - o'_i.$$

Since O forms a subspace, we know $o_i - o'_i \in O$ and thus, we found a vector in the secret subspace.

It has already been shown that UOV [KPG99] and Rainbow [DS05] are vulnerable to this kind of attack [AKKM22], so this can be seen as an extension of the approach to MAYO, which also works with vinegar and oil variables. Note, that the attack leads to valid signatures, and therefore, cannot be mitigated by a signature check.

Implemented Countermeasure. To mitigate this attack we shuffle the vinegar variables $v_i \in \mathbb{F}_q^n$ at the end of the signing algorithm. This is more secure than zeroing the respective variables since $v_i = 0$ could also lead to the leakage of

oil variables in the next signing procedure. Thus, it is advisable to permute the entries of the used variables instead, rendering them unknown to an attacker and ensuring $v_i \neq v'_i$.

Skip Addition of Oil Values. An attack vector that follows a similar reasoning, is to skip the addition of the oil variable o_i at the end of the signing process (see line 45 in Algorithm 1) for one (or more) $i \in 1, \ldots, k$. If the fault is injected correctly, this modifies the resulting signature to $s' = (v_1 + o_1, \ldots, v_i, \ldots, v_k + o_k)$. First, we see that s' is not a valid signature anymore, since $\mathcal{P}(s') \neq t$ with very high probability. Let s be the valid signature corresponding to the same message, then we can compute

$$s_i - s'_i = v_i + o_i - v'_i = o_i \in O.$$

Note that the signing is deterministic and the randomness that is used to generate the vinegar variable depends solely on the given message, which we have chosen to be identical. Therefore, $v_i = v'_i$. Again, we found a vector of the secret oilspace $o_i \in O$ and recover the remaining space with the reconciliation attack in negligible time.

Implemented Countermeasure. To avoid this attack we need to guarantee, that the vinegar and oil variables are really added, and neither of them are part of the signature by skipping their addition or the assignment of their values. Since the faulted signature is not valid anymore, one option is to verify the generated signature. However, this comes with a considerable performance overhead. Therefore, we rather chose to implement a check, that monitors if the entries of the computed signature s_i are different from the earlier generated vinegar variables v_i.

4.2 Side Channel Analysis

In this section, we focus on the leakage of the vector-matrix multiplication function. This function is called multiple times during key generation, secret key expansion and signing. It multiplies a secret vector by a known matrix (part of the public key), as shown in line 29 and 30 of Algorithm 1, as well as in line 16 of the algorithm MAYO.CompactKeyGen() and in line 17 of MAYO:ExpandSK(csk), for which we refer to [BCC+23, Section 2.1.5]. In MAYO, or more general, in UOV-based signature schemes, this is repeated for a considerable amount of public key matrices $P_i^{(1)}$.

An attacker is able to measure the power traces of the multiplication $(v_i)_j \cdot (P_a^{(1)})_{j,\cdot}$ for several $a \in [m]$, perform a profiling or a correlation attack, and predict the value $(v_i)_j$ which is supposed to remain unknown. This attack strategy was demonstrated in [ACK+23] ,where the authors attack an implementation of UOV, that incorporates similar operations as the one mentioned above. Again, the recovered values of v_i lead to efficient key recovery.

Implemented Countermeasure. In order to execute the SCA successfully, the attacker needs to know both, the value of the cofactor in $P_a^{(1)}$ and at which point in time the target $(v_i)_j$ is multiplied with this value. Thus, our approach to mitigate this attack, is to rearrange the order in which the multiplications are executed. In previous implementations optimized for efficiency a vinegar variable $(v_i)_j$ is picked and multiplied consecutively to the corresponding entry in all $P_a^{(1)}$ for $a \in \{1, \ldots, m\}$. This way, there is a certain interval in the power trace, that contains m multiplications of the sensitive value $(v_i)_j$ with public values. We treat the $P_a^{(1)}$ individually, and thus, the entry $(v_i)_j$ is only multiplied with $(P_a^{(1)})_{j.}$ before we move on to the next multiplication $(v_i)_{j+1} \cdot (P_a^{(1)})_{j+1,\cdot}$. Consequently, on a 32-bit architecture, where at least 4 field elements are treated at once (even 8 if we move to the updated parameters $q = 16$), this massively increases the failure probability of a correlation attack, since the power trace is now related to 4 different secret field elements $((v_i)_j, (v_i)_{j+1}, (v_i)_{j+2}, (v_i)_{j+3})$ at once, and not only to the same secret element $(v_i)_j$ as previously. However, more advanced analysis methods that employ machine learning for the selection of point of interest might still pose a threat to this approach. This could require a vast amount of profiling traces and we leave a concrete analysis thereof as future work.

5 Results, Comparison, and Discussion

In Table 1, we show the resource consumption of the whole design and sub-modules for the first security level defined by the NIST PQC standardization process [NIS23b]. The parameters defining MAYO are q (the size of the finite field), n (the number of variables in the multivariate quadratic polynomials in the public key), m (the number of multivariate quadratic polynomials in the public key), o (the dimension of the oil space), and k (the whipping parameter, satisfying $ko \geq m$). For our results, these parameters are set to $q = 31$, $n = 62$, $m = 60$, $o = 6$, and $k = 10$.

Our design stands out as the most optimized among the current implementations of multivariate schemes concerning resource utilization. The proposed design effectively utilizes roughly 31% of the total logic resources available on the Zynq board, specifically accounting for 13K Flip-Flops (FFs) and 21K Look-Up Tables (LUTs). These resources are distributed among different sub-modules.

The dominance of the Keccak core is evident as it commands the majority of FPGA slices, enveloping nearly third of the entire design. This dominance arises from its expansive internal buffer and its' interwoven XOR network, crucial for generating the output hash. Additionally, the RNG Core, integrating AES-128, significantly contributes to resource consumption. Remarkably, the combined impact of these cores results in approximately 40% (9K LUTs) of the design's overall slice usage, underscoring the notion that the MAYO core in isolation represents a minimalist design.

Our Gaussian elimination proves an improvement in the memory utilization as compared to the previous work [REBG11]. In [REBG11], the FPGA imple-

mentation on Xilinx Spartan-3 XC3S1500 (300 MHz) consumes 7,384 and 2,574 LUTs and FFs, respectively, for a number of equations equal to 50. In our implementation on a Zynq Z-7020 (100MHz), for a number of equations equal to 60, the consumption in LUTs and FFs is 1,822 and 413, respectively.

Table 1. Resource utilization of our hardware design on Zynq 7020 at a frequency of 100 MHz.

	Resource Utilization		
Submodules	LUTs	FFs	DSP
Keccak (Hash)	6759	4453	0
RNG	2354	3208	0
Vector-Matrix multiplication	1035	528	8
Oil Space Sampling	176	289	0
Gaussian Elimination	1822	413	3
Vector Addition	485	300	0
Vector Negation	176	93	0
Vinegar Sampling	245/686*	277/614*	0
BRAMs Port management	448	0	0
FSM Signing	2871	1057	0
Combined Architectures	21000	13005	11

* Secure implementation

Table 2. Comparison of our results with related work

Implementation	Platform	LUT	FF	DSP	BR
Our	Z-7020 @ 100MHz	21,000	13,005	11	129
[HSMR23]	KC705 @ 100MHz	91,266	42,113	2	45
[HSMR23]	AU280 @ 225MHz	89,014	42,066	2	45
[BCH+23]	Artix-7 @ 90.8MHz	32,422	23,262	2	48

Implementation	Platform	Key Generation cylcles	Signing cycles
Our	Z-7020 @ 100MHz	996K	2,867K
[HSMR23]	KC705 @ 100MHz	12K	42K
[BCH+23]	Artix-7 @ 90.8MHz	11,072K	843K

We present in Table 1 the resource utilization of our implementation. While the implementation by [HSMR23] is highly optimized for efficiency, our implementation shows better performance in the direction of LUT and FF usage, as showed in Table 2. We use 4.3× less LUTs and 3.2× less FFs while our BRAM utilization is 2.8× more, we believe that this is due to the parameter set we follow specifically the choice of q = 31. This also has a significant impact on the execution time, since we could not rely on optimized modules, but had to build some of them from scratch like the method for solving SLEs (see Sect. 3.5 for more details). Tuning our implementation to the new parameter set so that each two elements can be packed in one byte for example will result in reducing considerably the BRs utilization and execution time.

When compared to the implementation from [BCH+23] corresponding to a hardware implementation of the variant of ov-Ip with $n = 112$, $m = 44$, and $\mathbf{F_{256}}$, our implementation shows less consumption of LUTs. This is mainly due to the fact that their implementation LUTs utilization increases with higher q [BCH+23]. For example, for $\mathbf{F_{256}}$, the LUT is 8-in-8-out and requires 40 LUTs in the synthesis, while for $\mathbf{F_{16}}$, it requires 2 LUTs [BCH+23]. Our results show reduced LUTs and FFs which lead to faster logic operations, potentially resulting in improved clock speeds and reduced latency, especially for the key generation. The integration of 8 DSPs for vector-matrix multiplication shows potential for heightened parallel processing capabilities within the system architecture.

Our primary goal revolved around achieving an efficient usage of memory utilization and taking a first step towards physical security. Furthermore, our

parameters choice proved the adaptability of the MAYO scheme for deployment in resource-constrained devices *even* in case the field is extended to $q = 31$ instead of 16. In fact, our implementation offers a commendable trade-off, showcasing an adept combination of efficient resource utilization and operational speed. Furthermore, the implementation of the proposed countermeasures had hardly any impact on resource utilization as shown in Table 1. The increase in clock cycles that originates from the countermeasures lies in the order of hundreds and can be disregarded when considering the overall costs.

6 Conclusion

The implementation of multivariate signature schemes has faced challenges due to their large key sizes, impeding them from deployment on resource-constrained embedded devices. In response, the MAYO scheme was developed as a new modification of the mature UOV signature scheme. MAYO has successfully addressed the issue of large key sizes and can now be seen as one of the prominent candidates of NIST's call for additional digital signatures in regard of performance, key, and signature size. In this paper, we introduced a reconfigurable hardware implementation of MAYO, optimized to reduce the memory consumption during the key generation and the signing processes. Our implementation serves as evidence of MAYO's practicality for real-world deployment especially when deployed in resource-constraints devices. In fact, our design highlights the necessity of time area trade off. Moreover, we discussed a set of new security challenges brought by the deployment of MAYO in embedded systems, particularly in terms of defending against fault injection and side-channel attacks and suggest lightweight countermeasures.

Acknowledgments. The authors acknowledge the financial support by the Federal Ministry of Education and Research of Germany in the programme of the project Full Lifecycle Post-Quantum PKI - FLOQI (ID 16KIS1074). Furthermore, this work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - project number 505500359. Moreover, we would like to thank Amir Moradi for his valuable input which greatly improved the paper.

References

[ACK+23] Aulbach, T., Campos, F., Krämer, J., Samardjiska, S., Stöttinger, M.: Separating oil and vinegar with a single trace: side-channel assisted Kipnis-Shamir attack on UOV. IACR Trans. Cryptogr. Hardw. Embed. Syst. 221–245 (2023)

[AKKM22] Aulbach, T., Kovats, T., Krämer, J., Marzougui, S.: Recovering rainbow's secret key with a first-order fault attack. In: Batina, L., Daemen, J. (eds.) AFRICACRYPT 2022. LNCS, vol. 13503, pp. 348–368. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-17433-9_15

[ARM] ARM. Armv7-m architecture reference manual. https://developer.arm.com/documentation/ddi0403/d/Application-Level-Architecture/The-ARMv7-M-Instruction-Set

[BCC+23] Beullens, W., Campos, F., Celi, S., Hess, B., Kannwischer, M.: MAYO-algorithm specifications. MAYO team (2023). https://pqmayo.org/assets/specs/mayo.pdf

[BCH+23] Beullens, W., et al.: Modern Parameters and Implementations. Cryptology ePrint Archive (2023)

[BDH+22] Bertoni, G., Daemen, J., Hoffert, S., Peeters, M., Van Assche, G., Van Keer, R.: Keccak open-source hardware implementation (2022). https://keccak.team/index.html

[Beu21] Beullens, W.: Improved cryptanalysis of UOV and Rainbow. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021. LNCS, vol. 12696, pp. 348–373. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77870-5_13

[Beu22a] Beullens, W.: Breaking rainbow takes a weekend on a laptop. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022. LNCS, vol. 13508, pp. 464–479. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-15979-4_16

[Beu22b] Beullens, W.: MAYO: practical post-quantum signatures from oil-and-vinegar maps. In: Altawy, R., Hülsing, A. (eds.) SAC 2021. LNCS, vol. 13203, pp. 355–376. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-99277-4_17

[DS05] Ding, J., Schmidt, D.: Rainbow, a new multivariable polynomial signature scheme. In: Ioannidis, J., Keromytis, A., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 164–175. Springer, Heidelberg (2005). https://doi.org/10.1007/11496137_12

[FG18] Ferozpuri, A., Gaj, K.: High-speed FPGA implementation of the NIST round 1 rainbow signature scheme. In: 2018 International Conference on ReConFigurable Computing and FPGAs (ReConFig), pp. 1–8 (2018)

[HSMR23] Hirner, F., Streibl, M., Mert, A.C., Roy, S.S.: A hardware implementation of mayo signature scheme. IACR Cryptology ePrint Archive 2023:1267 (2023)

[HZ18] Yi, H., Nie, Z.: High-speed hardware architecture for implementations of multivariate signature generations on FPGAs. EURASIP J. Wirel. Commun. Netw. 1687–1499 (2018)

[KPG99] Kipnis, A., Patarin, J., Goubin, L.: Unbalanced oil and vinegar signature schemes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 206–222. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_15

[KS06] Kipnis, A., Shamir, A.: Cryptanalysis of the oil and vinegar signature scheme. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 257–266. Springer, Heidelberg (2006). https://doi.org/10.1007/BFb0055733

[NIS23a] NIST. NIST post-quantum cryptography standardization (2023). https://csrc.nist.gov/Projects/post-quantum-cryptography/workshops-and-timeline

[NIS23b] NIST. NIST post-quantum cryptography standardization: evaluation criteria (2023). https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/evaluation-criteria/security-(evaluation-criteria)

[NIS23c] NIST. SHA-3 standard: permutation-based hash and extendable-output functions (2023). https://csrc.nist.gov/publications/detail/fips/202/final

[PQD23] PQDB post-quantum data base (2023). https://www.pqdb.info/

[REBG11] Rupp, A., Eisenbarth, T., Bogdanov, A., Grieb, O.: Hardware SLE solvers: efficient building blocks for cryptographic and cryptanalytic applications. Integration **44**(4), 290–304 (2011)

[TYD+11] Tang, S., Yi, H., Ding, J., Chen, H., Chen, G.: High-speed hardware implementation of rainbow signature on FPGAs. In: Yang, B.Y. (ed.) PQCrypto 2011. LNCS, vol. 7071, pp. 228–243. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25405-5_15

[Xil23] AMD Xilinx. Zynq-7000 SoCs with Hardware and Software Programmability (2023). https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html

[XL21] Xing, Y., Li, S.: A compact hardware implementation of CCA-secure key exchange mechanism CRYSTALS-KYBER on FPGA. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2021**(2), 328–356 (2021)

[ZZW+21] Zhao, C., et al.: A compact and high-performance hardware architecture for CRYSTALS-Dilithium. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2022**(1), 270–295 (2021)

C

Analyzing Security Features

In this chapter, we include our contributions to the analysis of security features of UOV-based signature schemes.

Contents

In Appendix C.1, we include the PQCrypto'24 paper *Practical Key-Recovery Attack on MQ-Sign and More* [**A**DM+24].

Finally, the paper *Hash your Keys before Signing: BUFF Security of the Additional NIST PQC Signatures* in Appendix C.2 is a copy of the PQCrypto'24 paper [**A**ST24].

Practical Key-Recovery Attack on MQ-Sign and More

Thomas Aulbach[1( )], Simona Samardjiska[2], and Monika Trimoska[3]

[1] University of Regensburg, Regensburg, Germany
thomas.aulbach@ur.de
[2] Radboud Universiteit, Nijmegen, The Netherlands
simonas@cs.ru.nl
[3] Eindhoven University of Technology, Eindhoven, The Netherlands
m.trimoska@tue.nl

Abstract. In this paper we describe attacks on the UOV-based signature scheme called MQ-Sign. MQ-Sign was submitted by Shim, Kim, and An as a first-round candidate for standardization in the (South) Korean post-quantum cryptography competition (KpqC). The scheme makes use of sparseness of the secret central polynomials and equivalent key construction to reduce the size of the private key. The authors propose four variants exploiting different levels of sparsity, MQ-Sign-SS, MQ-Sign-RS, MQ-Sign-SR, and MQ-Sign-RR with the last one being the standard UOV signature scheme.

We show that apart from the MQ-Sign-RR variant, all the others are insecure. Namely, we present a polynomial-time key-recovery attack on the variants MQ-Sign-SS and MQ-Sign-RS and a forgery attack on the variant MQ-Sign-SR below the claimed security level. Our attack exploits exactly the techniques used for reduction of keys - the sparsity of the central polynomials in combination with the specific structure of the secret linear map $\mathbf{S}$.

We provide a verification script for the polynomial-time key-recovery attack, that recovers the secret key in less than seven seconds for security level V. Furthermore, we provide an implementation of the non-guessing part of the forgery attack, confirming our complexity estimates.

1 Introduction

In recent years we have witnessed a substantial effort from standardization bodies and the cryptographic community to design, develop and scrutinize candidates for post-quantum secure key-encapsulation mechanisms and digital signatures [7,12,14,20,26]. This effort is racing an equally fuelled one for developing a large scale error-tolerant universal quantum computer which, although still very much elusive, will likely be reality in a decade or so [19]. When this happens, all the classical cryptography we are happily using today will be immediately rendered insecure. Therefore, as the community widely agrees upon, we need to move as fast as possible with the standardization of post-quantum cryptosystems that we believe are secure even against quantum adversaries.

M.-J. Saarinen and D. Smith-Tone (Eds.): PQCrypto 2024, LNCS 14772, pp. 168–185, 2024.
https://doi.org/10.1007/978-3-031-62746-0_8

On the other hand, we need to be extremely careful in the assessment of the level of scrutiny put into these standardization processes. For example, a major disruption in NIST's standardization process, and certainly a shock for the crypto community, was the cryptanalysis [3,28] of the two multivariate quadratic ($\mathcal{MQ}$) signature schemes - GeMSS [6] and Rainbow [8] after they were chosen as finalists [20]. Both of these schemes were thought to be well understood, with solid security analysis, albeit both with ad-hoc designs and no security proof.

These developments resulted in NIST choosing two lattice-based signature schemes in the new standard [18,25] in addition to the heavy SPHINCS+ [13], and no adequate solution for use-cases in need of very small signatures. NIST reopened the call for post-quantum digital signature proposals, specifying the need for shorter signatures with fast verification. This spurred a huge number of new multivariate signatures, many of which are variants of UOV (Unbalanced Oil and Vinegar) [16]. UOV is one of the oldest, simplest and most studied ad-hoc multivariate signature schemes. It has very short signatures, but the public key is huge. Therefore, it was not particularly interesting for a very long time, especially since the alternative Rainbow seemed to be more efficient for the same security level (after the attack by Beullens [3] this advantage disappeared). After Rainbow was out of the game, the community returned to UOV in a new round of attempts to reduce the size of the public key while not compromising the security.

One of those efforts is the MQ-Sign [27] signature scheme submitted to the Korean Post-Quantum Cryptography Competition [26], and since recently selected to advance to the 2nd round. The MQ-Sign submission combines two known techniques from multivariate cryptography - equivalent keys [24] and sparse central polynomials [30]. The central map is a standard UOV map that can additionally exhibit sparseness in either the vinegar-vinegar part or the vinegar-oil part. The authors propose four different variants. Both the vinegar-vinegar and vinegar-oil parts being sparse corresponds to the MQ-Sign-SS variant, which yields the smallest private keys. In the variant MQ-Sign-RS, the vinegar-vinegar part is random and the vinegar-oil part is sparse. The two parts switch their structure in the MQ-Sign-SR variant. Finally, the variant MQ-Sign-RR, where both parts are random, corresponds to the standard UOV signature scheme.

1.1 Our Contribution

In this work, we study the security of the MQ-Sign signature scheme. We propose two attacks that cover all variants using sparseness, i.e. every except the last, MQ-Sign-RR variant.

First, we show how the property of using sparse polynomials can be exploited to develop a polynomial time key-recovery attack on the variants MQ-Sign-SS and MQ-Sign-RS. Our attack relies on two key properties – the sparseness property of the vinegar-oil quadratic part and the specific structure of the linear transformation $\mathbf{S}$, as per the *equivalent keys* key generation technique. We first recover the linear transformation $\mathcal{S}$, which allows to subsequently compute the

central map $\mathcal{F}$. Our attack is very efficient, and recovers the key in just seconds regardless of the security level.

Second, we introduce a forgery attack on the variant MQ-Sign-SR which is actually a direct attack using only the public key. Our attack exploits a bilinear substructure emerging as a result of the sparse secret polynomials. The attack is not practical, but still shows that MQ-Sign-SR falls short of the claimed security levels by about 30 bits.

We perform a complexity analysis of both attacks, showing that these three variants do not reach the originally estimated security levels. The claims in our complexity analysis are additionally backed up with experimental results. Most notably, we provide an implementation of the practical key-recovery attack that is executed in less than seven seconds for all security levels. We also provide an implementation of the non-guessing part of the forgery attack, confirming our complexity estimates. Both the implementation of attacks and the code used for confirming the complexity estimates are open source.

1.2 Timeline

Our key recovery attack on MQ-Sign-RS and MQ-Sign-SS with $\mathcal{S}$ in block matrix structure (using the equivalent keys optimization) was announced in March 2023. Shortly afterwards, Ikematsu, Jo, and Yasuda [15] generalized our approach and gave an efficient attack that also works with general $\mathcal{S}$. As a result of the two attacks, the authors of MQ-Sign removed the two variants MQ-Sign-RS and MQ-Sign-SS from their specifications in the ongoing KpqC competition. Note that in the current version of the specifications, both remaining variants still use the equivalent key optimization, and do not use a random linear transformation $\mathcal{S}$.

1.3 Organization of the Paper

In Sect. 2 we provide the necessary background on multivariate cryptography, in particular the UOV signature scheme and the optimization choices used in MQ-Sign. We introduce the announced attacks in Sect. 3 and 4. In more detail, we first show in Sect. 3 that the sparse vinegar-oil polynomials in MQ-Sign-RS and MQ-Sign-SS let us derive enough linear equations to compute the secret linear transformation $\mathcal{S}$ in a matter of seconds. Section 4 demonstrates a strategy to attack MQ-Sign-SR by first guessing a selection of variables and subsequently solving a part of the equations for the remaining ones. Even though the cost of the guessing part remains quite high, this shows that the remaining sparse variant slightly fails to provide the required security levels. We provide verification scripts of the stated attacks in Sect. 5 and discuss the impact on the MQ-Sign variants in Sect. 6. Finally, we debate about the still appealing question of using sparse polynomials in UOV and shift attention to the public equations instead.

2 Preliminaries

Throughout the text, $\mathbb{F}_q$ will denote the finite field of q elements, and $\mathrm{GL}_n(\mathbb{F}_q)$ and $\mathrm{AGL}_n(\mathbb{F}_q)$ will denote respectively the general linear group and the general affine group of degree n over $\mathbb{F}_q$. We will also use the notation $\mathbf{x} = (x_1, \ldots, x_n)^\intercal$ for the vector $(x_1, \ldots, x_n) \in \mathbb{F}_q^n$.

2.1 Multivariate Signatures

First, we recall the general principle of $\mathcal{MQ}$ public key cryptosystems. A typical $\mathcal{MQ}$ public key cryptosystem relies on the knowledge of a trapdoor for a particular system of polynomials over the field $\mathbb{F}_q$. The public key of the cryptosystem is usually given by a multivariate quadratic map $\mathcal{P} = (\mathcal{P}^{(1)}, \ldots, \mathcal{P}^{(m)}) : \mathbb{F}_q^n \to \mathbb{F}_q^m$, where

$$\mathcal{P}^{(k)}(x_1, \ldots, x_n) = \sum_{1 \le i \le j \le n} \gamma_{ij}^{(k)} x_i x_j + \sum_{i=1}^{n} \beta_i^{(k)} x_i + \alpha^{(k)}$$

for some coefficients $\gamma_{ij}^{(k)}, \beta_i^{(k)}, \alpha^{(k)} \in \mathbb{F}_q$. It is obtained by obfuscating a structured central map

$$\mathcal{F} : (x_1, \ldots, x_n) \in \mathbb{F}_q^n \to \left(\mathcal{F}^{(1)}(x_1, \ldots, x_n), \ldots, \mathcal{F}^{(m)}(x_1, \ldots, x_n)\right) \in \mathbb{F}_q^m,$$

using two bijective affine mappings $\mathcal{S}, \mathcal{T} \in \mathrm{AGL}_n(\mathbb{F}_q)$ that serve as a sort of mask to hide the structure of $\mathcal{F}$. The public key is defined as

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

The mappings $\mathcal{S}$ and $\mathcal{T}$ are part of the private key s. Besides them, the private key may also contain other secret parameters that allow creation, but also easy inversion of the transformation $\mathcal{F}$. Without loss of generality, we can assume that the private key is $s = (\mathcal{F}, \mathcal{S}, \mathcal{T})$.

Signature Generation. To generate a signature for a message d, the signer uses a hash function $\mathcal{H} : \{0,1\}^\star \to \mathbb{F}_q^m$ to compute the hash value $\mathbf{w} = \mathcal{H}(d) \in \mathbb{F}_q^m$ and computes recursively $\mathbf{x} = \mathcal{T}^{-1}(\mathbf{w}) \in \mathbb{F}_q^m$, $\mathbf{y} = \mathcal{F}^{-1}(\mathbf{x}) \in \mathbb{F}_q^n$, and $\mathbf{z} = \mathcal{S}^{-1}(\mathbf{y})$. The signature of the message d is $\mathbf{z} \in \mathbb{F}_q^n$. Here, $\mathcal{F}^{-1}(\mathbf{x})$ means finding one (of possibly many) preimages of $\mathbf{x}$ under the central map $\mathcal{F}$.

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

The standard signature generation and verification process of a multivariate signature scheme works as shown in Fig. 1.

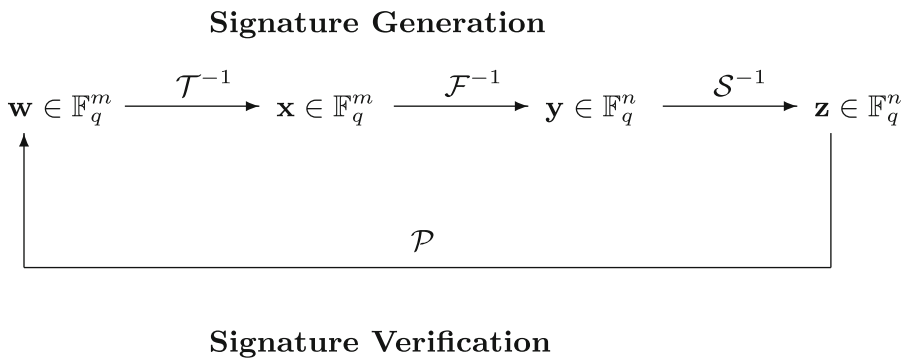


Fig. 1. General workflow of multivariate signature schemes.

2.2 Unbalanced Oil and Vinegar

The Unbalanced Oil and Vinegar signature scheme is one of the oldest multivariate signature schemes. It was proposed by Kipnis, Patarin, and Goubin at EUROCRYPT'99 [16] as a modification of the oil and vinegar scheme of Patarin [22] that was broken by Kipnis and Shamir in 1998 [17].

The characteristic of the oil and vinegar construction is in the special structure of the central map in which the variables are divided in two distinct sets, vinegar variables and oil variables. The vinegar variables are combined quadratically with all of the variables, while the oil variables are only combined quadratically with vinegar variables and not with other oil variables. Formally, the central map is defined as $\mathcal{F} : \mathbb{F}_q^n \to \mathbb{F}_q^m$, with central polynomials

$$\mathcal{F}^{(k)}(x_1, \ldots, x_n) = \sum_{i \in V, j \in V} \gamma_{ij}^{(k)} x_i x_j + \sum_{i \in V, j \in O} \gamma_{ij}^{(k)} x_i x_j + \sum_{i=1}^{n} \beta_i^{(k)} x_i + \alpha^{(k)} \quad (1)$$

where $n = v + m$, and $V = \{1, \ldots, v\}$ and $O = \{v + 1, \ldots, n\}$ denote the index sets of the vinegar and oil variables, respectively.

It can be shown that if an oil and vinegar central map is used in the standard $\mathcal{MQ}$ construction the affine mapping $\mathcal{T}$ does not add to the security of the scheme and is therefore not necessary. Hence the secret key consists of a linear transformation $\mathcal{S}$ and central map $\mathcal{F}$, while the public key is defined as $\mathcal{P} = \mathcal{F} \circ \mathcal{S}$. In order to sign a message, we need to find a preimage of $\mathcal{F}$. This can be done by simply fixing the vinegar variables to some random values. In this way, we obtain a system of m linear equations in m variables, which has a solution with probability around $1 - 1/q$. If the obtained system does not have a solution, we repeat the procedure with different values for the vinegar variables.

Key Generation. It was shown in [23] that for any instance of a UOV secret key $(\mathcal{F}, \mathcal{S})$, there exists an equivalent secret key $(\mathcal{F}, \mathbf{S})$ with

$$\mathbf{S} = \begin{pmatrix} \mathbf{I}_{v \times v} & \mathbf{S}_1 \\ \mathbf{0}_{m \times v} & \mathbf{I}_{m \times m} \end{pmatrix}. \quad (2)$$

Furthermore, the quadratic polynomials of the central map $\mathcal{F} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ can be represented using upper triangular matrices $\mathbf{F}^{(1)}, \ldots, \mathbf{F}^{(m)} \in \mathbb{F}_q^{n \times n}$ where each nonzero coefficient (i, j) in $\mathbf{F}^{(k)}$ corresponds to the nonzero coefficient of $x_i x_j$ in $\mathcal{F}^{(k)}$. Note that the $m \times m$ block on the bottom right of these matrices is empty, since the polynomials of the central map have no quadratic oil terms. Thus, these matrices contain an upper triangular block $\mathbf{F}_1^{(k)} \in \mathbb{F}_q^{v \times v}$ and a block $\mathbf{F}_2^{(k)} \in \mathbb{F}_q^{v \times m}$ on the top right. In other words, the matrices are of the form:

$$\mathbf{F}^{(k)} = \begin{pmatrix} \mathbf{F}_1^{(k)} & \mathbf{F}_2^{(k)} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}.$$

Thus, in order to obtain a key pair, it suffices to first randomly generate $(\mathbf{S}_1, \mathbf{F}^{(1)}, \ldots, \mathbf{F}^{(m)})$ and then compute $(\mathbf{P}^{(1)}, \ldots, \mathbf{P}^{(m)})$ by evaluating $\mathbf{P}^{(k)} = \mathbf{S}^\top \mathbf{F}^{(k)} \mathbf{S}$ and bringing the resulting matrices to upper triangular form.

2.3 MQ-Sign

MQ-Sign is a signature scheme based on UOV. The scheme uses inhomogenous polynomials and each polynomial of the central map can be written as

$$\mathcal{F}^{(k)} = \mathcal{F}_V^{(k)} + \mathcal{F}_{OV}^{(k)} + \mathcal{F}_{L,C}^{(k)}$$

where

$$\mathcal{F}_V^{(k)}(x_1, \ldots, x_n) = \sum_{i \in V, j \in V} \gamma_{ij}^{(k)} x_i x_j, \text{ and } \mathcal{F}_{OV}^{(k)}(x_1, \ldots, x_n) = \sum_{i \in V, j \in O} \gamma_{ij}^{(k)} x_i x_j.$$

These can alternatively be referred to as the vinegar-vinegar quadratic part and the vinegar-oil quadratic part. Finally, $\mathcal{F}_{L,C}^{(k)}$ refers to the linear and constant part of the polynomials. In the following, we ignore the linear and constant parts, since our attack does not use them.

The main design goal of MQ-Sign is to reduce the size of the secret key compared to traditional UOV. This is achieved using sparse polynomials for the quadratic part of the central map. If sparseness is introduced in the $\mathcal{F}_V^{(k)}$ part, then it is defined as

$$\mathcal{F}_{V,S}^{(k)}(x_1, \ldots, x_n) = \sum_{i=1}^{v} \alpha_i^k x_i x_{(i+k-1(\bmod v))+1} \tag{3}$$

If, on the other hand, sparseness is introduced in the $\mathcal{F}_{OV}^{(k)}$ part, then it is defined as

$$\mathcal{F}_{OV,S}^{(k)}(x_1, \ldots, x_n) = \sum_{i=1}^{v} \beta_i^k x_i x_{(i+k-2(\bmod m))+v+1}. \tag{4}$$

The MQ-Sign proposal provides a parameter selection for four variants of the scheme: MQ-Sign-SS, MQ-Sign-RS, MQ-Sign-SR and MQ-Sign-RR. The first

S/R in the suffix specifies whether $\mathcal{F}_V$ is defined with sparse ($\mathcal{F}_{V,S}$) or random polynomials ($\mathcal{F}_{V,R}$). The second S/R refers to the same property, but for $\mathcal{F}_{OV}$. Note that the variant MQ-Sign-RR corresponds to the standard UOV scheme defined with inhomogenous polynomials.

If both $\mathcal{F}_{V,S}$ and $\mathcal{F}_{OV,S}$ are used, the size of the secret key is reduced to $2vm$ field elements.

The authors provide an elaborate security analysis including all known relevant attacks on UOV. However, they do not consider the sparseness of (parts of) the secret polynomials in any of the attacks. Their assumption is that it is not exploitable within the known cryptanalytic techniques. Table 1 summarizes the parameters chosen by the authors for security levels I, III, and V.

Note that when $\mathcal{F}_{V,S}$ is used, the size of the public key can also be reduced, as, due to the equivalent keys structure of $\mathbf{S}$ as in (2), a part of the public key is equivalent to a part of the secret key and thus sparse. This is however not taken into consideration in the implementation of MQ-Sign or in the public key sizes reported in Table 1.

Table 1. The parameter selection for security category I, III and V for the variants SS, RS, SR and RR of MQ-Sign with key sizes in bytes.

Sec. level	Parameters (q, v, m)	sig	PK	SK (SS)	SK (RS)	SK (SR)	SK (RR)
I	$(2^8, 72, 46)$	134	328 441	15 561	133 137	164 601	282 177
III	$(2^8, 112, 72)$	200	1 238 761	37 729	485 281	610 273	1 057 825
V	$(2^8, 148, 96)$	260	2 892 961	66 421	1 110 709	1 416 181	2 460 469

3 An Efficient Key-Recovery Attack on Variants Using Sparse $\mathcal{F}_{OV}$

In the following, we consider $\mathcal{C}$ to be the class of polynomials defined by $\mathcal{F}_{V,R} + \mathcal{F}_{OV,S}$, denoting that only $\mathcal{F}_{OV}$ needs to be defined as in (4), i.e. with sparse polynomials. This corresponds to the MQ-Sign-SS and MQ-Sign-RS variants.

In this section we show that the usage of $\mathcal{F}_{OV,S}$ introduces weaknesses that enable a practical key-recovery attack that takes merely seconds to mount. In the attack, we essentially solve the Extended Isomorphism of Polynomials (EIP) problem as defined in [9] (see also [27]). We recall here its definition.

$\mathsf{EIP}(n, m, \mathcal{P}, \mathcal{C})$:
Input: An m-tuple of multivariate polynomials $\mathcal{P} = (\mathcal{P}^{(1)}, \mathcal{P}^{(2)}, \ldots, \mathcal{P}^{(m)}) \in \mathbb{F}_q[x_1, \ldots, x_n]^m$ and a special class of m-tuples of multivariate polynomials $\mathcal{C} \subseteq \mathbb{F}_q[x_1, \ldots, x_n]^m$.
Question: Find – if any – $\mathbf{S} \in \mathrm{GL}_n(q)$ and $\mathcal{F} = (\mathcal{F}^{(1)}, \mathcal{F}^{(2)}, \ldots, \mathcal{F}^{(m)}) \in \mathcal{C}$ such that $\mathcal{P} = \mathcal{F} \circ \mathbf{S}$.

Solving this problem is in general not easy. In fact, the security of ad-hoc multivariate schemes is based on the hardness on this problem. However, if $\mathcal{F}$ exhibits enough structure, then the problem can become easy to solve.

We next show that the sparse structure present in MQ-Sign-SS and MQ-Sign-RS is enough to solve the corresponding EIP problem very efficiently. In order to see this, note that the computation of the public key for UOV-like signature schemes can be written in matrix form as:

$$\begin{pmatrix} \mathbf{P}_1^{(k)} & \mathbf{P}_2^{(k)} \\ \mathbf{0} & \mathbf{P}_4^{(k)} \end{pmatrix} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{S}_1^\top & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{F}_1^{(k)} & \mathbf{F}_2^{(k)} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{S}_1 \\ \mathbf{0} & \mathbf{I} \end{pmatrix},$$

for all $k \in \{1, \ldots, m\}$. From this we deduce

$$\begin{pmatrix} \mathbf{P}_1^{(k)} & \mathbf{P}_2^{(k)} \\ \mathbf{0} & \mathbf{P}_4^{(k)} \end{pmatrix} = \begin{pmatrix} \mathbf{F}_1^{(k)} & (\mathbf{F}_1^{(k)} + \mathbf{F}_1^{(k)\top})\mathbf{S}_1 + \mathbf{F}_2^{(k)} \\ \mathbf{0} & \mathsf{Upper}(\mathbf{S}_1^\top \mathbf{F}_1^{(k)} \mathbf{S}_1 + \mathbf{S}_1^\top \mathbf{F}_2^{(k)}) \end{pmatrix}, \tag{5}$$

where $\mathsf{Upper}(\mathbf{M})$ denotes the unique upper triangular matrix $\tilde{\mathbf{M}}$ such that the difference $\tilde{\mathbf{M}} - \mathbf{M}$ is skew-symmetric. Equation (5) shows how different blocks of the public key are obtained from the blocks of the secret key, and having these relations allows us to mount an algebraic attack that will recover all of the entries of the secret key. We first model this correspondence between the public and the secret key as a system of equations where the variables are the entries of $\mathbf{S}_1$ and $\mathbf{F}_1^{(k)}$. From the two upper blocks we obtain the following two equations

$$\begin{aligned} \mathbf{P}_1^{(k)} &= \mathbf{F}_1^{(k)} \\ \mathbf{P}_2^{(k)} &= (\mathbf{F}_1^{(k)} + \mathbf{F}_1^{(k)\top})\mathbf{S}_1 + \mathbf{F}_2^{(k)}. \end{aligned}$$

From these, we infer that

$$\mathbf{P}_2^{(k)} = (\mathbf{P}_1^{(k)} + \mathbf{P}_1^{(k)\top})\mathbf{S}_1 + \mathbf{F}_2^{(k)}. \tag{6}$$

Ignoring the sparseness at first, from (6) we can derive a linear system of vm^2 equations in $v(m^2 + m)$ variables (vm that correspond to the entries of the unknown block of the linear transformation $\mathbf{S}$, and vm^2 from the entries of $\mathbf{F}_2^{(k)}$). Even though the system is linear, a solution cannot be extracted easily as it is highly underdetermined. But considering the sparseness in the MQ-Sign-SS and MQ-Sign-RS instances, the following key observation allows us to solve the system easily in practice.

The matrices $\mathbf{F}_2^{(k)}$ are part of the secret key, but we know that they are sparse. From the description of $\mathcal{F}_{OV}$ in (4) we can see that the value of $\mathbf{F}_2^{(k)}$ is known on $(vm - v)$ entries. Since $\mathbf{F}_2^{(k)}$ appears linearly in (6), we can extract constraints from the entries where the value of $\mathbf{F}_2^{(k)}$ is zero and obtain a system that is only in the $\mathbf{S}_1$ variables. Let $\widetilde{\mathbf{P}}_1^{(k)} = \mathbf{P}_1^{(k)} + \mathbf{P}_1^{(k)\top}$. We obtain the following system of

equations, where we denote by $\tilde{p}_{i,j}^{(k)}$ the entries of $\widetilde{\mathbf{P}}_1^{(k)}$, by $s_{i,j}$ the entries of $\mathbf{S}_1$, by $p_{i,j}^{(k)}$ the entries of $\mathbf{P}_2^{(k)}$, and by $f_{i,j}^{(k)}$ the entries of $\mathbf{F}_2^{(k)}$.[1]

$$\sum_{1 \leqslant p \leqslant v} \tilde{p}_{i,p}^{(k)} s_{p,j} - p_{i,j}^{(k)} = 0, \quad \forall (i,j,k) \text{ s.t. } f_{i,j}^{(k)} = 0. \tag{7}$$

This is a linear system in vm variables. The number of equations that we can obtain if we use all of the m quadratic maps from the public key is $mv(m-1)$. Hence, the system has vm linearly independent equations with overwhelming probability. As such, it can be solved efficiently through Gaussian Elimination. This is under the assumption that the system behaves as a random system and has no specific structure that results in non-trivial dependencies between the equations, which will be argued below as part of the complexity analysis. We conclude that, ignoring some of the equations from (6), specifically those where $f_{i,j}^{(k)}$ is not zero, allowed us to derive a linear system that is only in variables from $\mathbf{S}_1$. Once we recover the secret map $\mathbf{S}$, computing $\mathcal{F}$ is easy, as we just need to apply the inverse linear transformation on $\mathcal{P}$.

We further refine our modeling to obtain a more efficient attack, using the following strategy. Note from (7) that each equation in the system contains variables from only one column of $\mathbf{S}_1$. This observation allows us to optimize the attack by solving for one column at a time. This is more evident when we look at the matrix representation of our linear system. Let us define a matrix $\mathbf{A}'$ as

$$\begin{pmatrix} \widetilde{\mathbf{P}}_1^{(1)} \\ \widetilde{\mathbf{P}}_1^{(2)} \\ \vdots \\ \widetilde{\mathbf{P}}_1^{(m)} \end{pmatrix},$$

i.e. a block matrix obtained by concatenating vertically the quadratic maps $\widetilde{\mathbf{P}}_1^{(k)}$. Then, let $\mathbf{A}$ be a block matrix that has copies of $\mathbf{A}'$ on the main diagonal and zeros everywhere else

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}' & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{A}' & \cdots & \mathbf{0} \\ \vdots & & \ddots & \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{A}' \end{pmatrix}.$$

Now, let $\mathbf{x}^\top = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m)$ be a vector obtained by concatenating the columns of $\mathbf{S}_1$. Finally, let $\mathbf{b}^\top = \mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m$, be a vector that is obtained by concatenating the first column of each quadratic map $\mathbf{P}_2^{(k)}$, followed by the second column of each map, etc.

[1] Here, and in the following, the submatrix indices are ommited where there is no ambiguity.

We can then rewrite $\widetilde{\mathbf{P}}_1^{(k)}\mathbf{S}_1 = \mathbf{P}_2^{(k)}$, for all $k \in \{1, \ldots, m\}$, as $\mathbf{Ax} = \mathbf{b}$. Indeed, we have

$$\begin{pmatrix} \mathbf{A}' & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{A}' & \cdots & \mathbf{0} \\ \vdots & & \ddots & \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{A}' \end{pmatrix} \cdot \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_m \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_m \end{pmatrix},$$

where

$$\mathbf{A}' = \begin{pmatrix} \tilde{p}_{1,1}^{(1)} & \cdots & \tilde{p}_{1,v}^{(1)} \\ \tilde{p}_{2,1}^{(1)} & \cdots & \tilde{p}_{2,v}^{(1)} \\ & \vdots & \\ \tilde{p}_{v,1}^{(1)} & \cdots & \tilde{p}_{v,v}^{(1)} \\ & \vdots & \\ \tilde{p}_{1,1}^{(m)} & \cdots & \tilde{p}_{1,v}^{(m)} \\ \tilde{p}_{2,1}^{(m)} & \cdots & \tilde{p}_{2,v}^{(m)} \\ & \vdots & \\ \tilde{p}_{v,1}^{(m)} & \cdots & \tilde{p}_{v,v}^{(m)} \end{pmatrix}, \quad \mathbf{x}_i = \begin{pmatrix} s_{1,i} \\ s_{2,i} \\ \vdots \\ s_{v,i} \end{pmatrix}, \quad \text{and } \mathbf{b}_i = \begin{pmatrix} p_{1,i}^{(1)} \\ p_{2,i}^{(1)} \\ \vdots \\ p_{v,i}^{(1)} \\ \vdots \\ p_{1,i}^{(m)} \\ p_{2,i}^{(m)} \\ \vdots \\ p_{v,i}^{(m)} \end{pmatrix}.$$

Looking at where the zero entries lie in $\mathbf{A}$, we can now split the problem. We solve $\mathbf{A}'\mathbf{x}_i = \mathbf{b}_i$ for all $i \in \{1, \ldots, m\}$, and for every system that we solve, we reveal one column of $\mathbf{S}_1$.

3.1 Complexity Analysis

Using this strategy, instead of solving one linear system in vm variables, we solve m linear systems in v variables. Thus, our attack has only $\mathcal{O}(mv^\omega)$ time complexity, where ω is the linear algebra constant. A strong requirement for the success of the attack is that all of the linear subsystems that we need to solve are determined. Since we are combining solutions of subsystems to recover the entire solution, having even a small nonzero number of solutions to the subsystems would rapidly increase the complexity of the attack. However, in the following, we argue that we can rely on the assumption that all subsystems have exactly one solution.

Table 2. Theoretical complexity of our attack against the MQ-Sign-SS and MQ-Sign-RS variants.

Security level	Parameters (q, v, m)	Attack complexity
I	$(2^8, 72, 46)$	2^{24}
III	$(2^8, 112, 72)$	2^{27}
V	$(2^8, 148, 96)$	2^{29}

As per the analysis in the previous section, the ith subset of equations is obtained from

$$\begin{pmatrix} \tilde{p}^{(1)}_{1,1} & \cdots & \tilde{p}^{(1)}_{1,v} \\ \tilde{p}^{(1)}_{2,1} & \cdots & \tilde{p}^{(1)}_{2,v} \\ & \vdots & \\ \tilde{p}^{(1)}_{v,1} & \cdots & \tilde{p}^{(1)}_{v,v} \\ & \vdots & \\ \tilde{p}^{(m)}_{1,1} & \cdots & \tilde{p}^{(m)}_{1,v} \\ \tilde{p}^{(m)}_{2,1} & \cdots & \tilde{p}^{(m)}_{2,v} \\ & \vdots & \\ \tilde{p}^{(m)}_{v,1} & \cdots & \tilde{p}^{(m)}_{v,v} \end{pmatrix} \cdot \begin{pmatrix} s_{1,i} \\ s_{2,i} \\ \vdots \\ s_{v,i} \end{pmatrix} = \begin{pmatrix} p^{(1)}_{1,1} \\ p^{(1)}_{2,1} \\ \vdots \\ p^{(1)}_{v,1} \\ \vdots \\ p^{(m)}_{1,1} \\ p^{(m)}_{2,1} \\ \vdots \\ p^{(m)}_{v,1} \end{pmatrix}. \tag{8}$$

From this equality, we extract $v(m-1)$ equations. That is, one equation for each entry from $\mathbf{b}_i$, ignoring entries (i, j) where $f^{(k)}_{i,j}$ is not zero. We are interested in how many of these equations are linearly independent. From (8) we can see that each equation can be viewed as a linear combination of the $s_{\_,i}$ variables where the coefficients come from a row of $\widetilde{\mathbf{P}}^{(k)}_1$, plus a constant that corresponds to an entry of $\mathbf{P}^{(k)}_2$. Hence, the number of linearly independent equations is exactly determined by the rank of $\mathbf{A}'$. It is actually the rank of $\left(\mathbf{A}' \ \mathbf{b}_i\right)$, but we can ignore the constant in our case. Indeed, if the rank of $\mathbf{A}'$ is smaller than the rank of $\left(\mathbf{A}' \ \mathbf{b}_i\right)$, this would result in the system derived from (8) being inconsistent. This case cannot happen when we model a coherent instance of UOV key generation. Now, recall that public key in UOV-based schemes is generated randomly (or derived from a randomly generated central map) and thus it is comprised of matrices of full rank with high probability. Hence, a concatenation of several such matrices is also full rank, which is v in this case (the dimension of the column space being v) – equal to the number of variables. We have also performed experiments to verify this claim, and out of 500 runs of the attack on MQ-Sign-SS with level I parameters, not once did the attack fail for not having enough independent equations in any of the subsystems. Table 2 summarizes the effect of the attack on the different MQ-Sign parameters.

4 A Forgery Attack on Variants Using Sparse $\mathcal{F}_V$

In this section we show a forgery attack on the MQ-Sign-SR variant, where the polynomials of $\mathcal{F}_V$ are defined as in Eq. (3). A forgery attack on a multivariate signature scheme aims at finding a signature $\mathbf{z} \in \mathbb{F}_q^n$ for a given target value $\mathbf{t} \in \mathbb{F}_q^m$, such that $\mathcal{P}(\mathbf{z}) = \mathbf{t}$ is fulfilled. We show that in the case of MQ-Sign-SR, a forgery is directly possible using only the public key.

Recall from Sect. 3, that, when the linear transformation $\mathcal{S}$ is given as in Eq. (2), it holds that $\mathbf{P}_1^{(k)} = \mathbf{F}_1^{(k)}$. This means that the sparsity of the secret coefficient matrices gets transferred to the public system. In more detail, an attacker faces the task of finding $(\mathbf{z}_v, \mathbf{z}_o) \in \mathbb{F}_q^n$ such that

$$(\mathbf{z}_v, \mathbf{z}_o) \begin{pmatrix} \mathbf{P}_1^{(k)} & \mathbf{P}_2^{(k)} \\ \mathbf{0} & \mathbf{P}_4^{(k)} \end{pmatrix} \begin{pmatrix} \mathbf{z}_v \\ \mathbf{z}_o \end{pmatrix} = \mathbf{z}_v \mathbf{P}_1^{(k)} \mathbf{z}_v + \mathbf{z}_v \mathbf{P}_2^{(k)} \mathbf{z}_o + \mathbf{z}_o \mathbf{P}_4^{(k)} \mathbf{z}_o = t_k \tag{9}$$

holds for all $k \in \{1, \ldots, m\}$, where $\mathbf{P}_1^{(k)}$ are sparse as in Eq. (3). The parameters $n \approx 2.5m$ allow us to fix the m entries of $\mathbf{z}_o \in \mathbb{F}_q^m$ and thereby remove the non-sparse submatrices $\mathbf{P}_2^{(k)}$ and $\mathbf{P}_4^{(k)}$ from the quadratic part of this system of equations. This leads us to equations of the form

$$\mathbf{z}_v \mathbf{P}_1^{(k)} \mathbf{z}_v + \mathrm{lin}(\mathbf{z}_v) = \sum_{i=1}^{v} \alpha_i^k z_i z_{(i+k-1(\bmod v))+1} + \mathrm{lin}(\mathbf{z}_v) = t_k. \tag{10}$$

The term $\mathrm{lin}(\mathbf{z}_v)$ summarizes the linear and constant terms emerging from Eq. (9) after fixing the entries of $\mathbf{z}_o$. Note that the resulting system is a system of m equations in v variables, and since v is greater than m, we can fix another $(v - m)$ variables and still expect to have a solution.

At the core of this attack is the observation that, due to the sparsity in $\mathbf{P}_1^{(k)}$, the resulting system has subsets of equations that are bilinear in some subsets of variables. Specifically, upon closer examination of the indices in Eq. (10), one notices that for odd k, the quadratic monomials appearing in the polynomial equation each consist of a variable with an odd and an even index. This implies that these $\frac{m}{2}$ equations are bilinear in the sets of variables $\{z_1, z_3, \ldots, z_{m-1}\}$ and $\{z_2, z_4, \ldots, z_m\}$, where we denote by z_i the variables in vector $\mathbf{z}_v$. Hence, randomly guessing e.g., the $\frac{v}{2}$ odd-indexed variables gives us a $\frac{v-m}{2}$-dimensional linear solution space for the even-indexed variables in the $\frac{m}{2}$ bilinear equations. Let us denote by $\tilde{\mathbf{z}}_v$ the vector comprised of the vinegar variables that have not yet been assigned, i.e. the even-indexed vinegar variables. At this point, the overall system is of the following form

$$\sum_{i=0}^{\frac{v}{2}-1} \alpha_{2i+1}^k z_{((2i+1)+k-1(\bmod v))+1} + \alpha_{2i+2}^k z_{2i+2} + \mathrm{lin}(\tilde{\mathbf{z}}_v) = t_k, \qquad \text{if } k \text{ odd}$$

$$\sum_{i=1}^{\frac{v}{2}} \alpha_{2i}^k z_{2i} z_{(2i+k-1(\bmod v))+1} + \mathrm{lin}(\tilde{\mathbf{z}}_v) = t_k, \qquad \text{if } k \text{ even.}$$

The probability that there exists a solution to the complete system - including the remaining $\frac{m}{2}$ quadratic (non-bilinear) equations - with the previously guessed odd variables is around $q^{-(\frac{v}{2}-(v-m))}$, since we can only fix $v-m$ variables in a quadratic system with v variables in m equations and still expect to find a solution. An alternative view is that, to obtain the $\frac{v-m}{2}$-dimensional linear solution space, we can fix $(v-m)$ variables and enumerate the rest with the usual cost of enumeration. This is the first step of our attack and its cost will be denoted by $C_{\text{ENUM}(q,\frac{v}{2}-(v-m))}$.

In the second step, we need to find an assignment to the even-indexed variables that also validate the remaining $\frac{m}{2}$ equations. Using the description of the linear solution space obtained from the bilinear equations, this step boils down to solving a quadratic system of $\frac{m}{2}$ equations in $\frac{v-m}{2}$ variables. We denote the complexity of this step by $C_{\text{MQ}(q,\frac{v-m}{2},\frac{m}{2})}$.

4.1 Complexity Analysis

The cost of the first step of the algorithm corresponds to the usual cost of enumeration over $\mathbb{F}_q$. In the second step, the complexity is dominated by the algorithm for solving the quadratic systems of equations. For the choice of $q = 2^8$, as per the MQ-Sign parameters, the best strategy would be to solve the system with a Gröbner-based algorithm (such as F4 or F5 [10,11]), without the use of hybridization. Assuming that the quadratic systems we obtain behave as semi-regular non-boolean systems of s equations in n variables, the complexity [2] of the solving algorithm is approximated by

$$\mathcal{O}\left(sD\binom{n+D-1}{D}^{\omega}\right),$$

where D denotes the *degree of regularity* and is computed as the power of the first non-positive coefficient in the expansion of

$$\frac{(1-t^2)^s}{(1-t)^n}.$$

Then, the complexity of the whole attack is given by

$$C_{\text{ENUM}(q,\frac{v}{2}-(v-m))} \cdot C_{\text{MQ}(q,\frac{v-m}{2},\frac{m}{2})},$$

since the second step has to be repeated until the odd variables are guessed correctly in the first step. In Table 3 we present an overview of the approximate costs for the parameter sets of MQ-Sign. We conclude that because of this attack, the proposed parameters of the MQ-Sign-SR variant slightly fail to provide the required security levels. Note that the algorithm described here uses the most straighforward approach to exploit the bilinearity of the subsystems, but more advanced techniques can potentially result in attacks with lower complexity.

Table 3. Theoretical complexity of our direct attack using the bilinear structure of the odd equations.

Security level	Parameters (q, v, m)	$C_{\text{ENUM}(q, \frac{v}{2}-(v-m))}$	$C_{\text{MQ}(q, \frac{v-m}{2}, \frac{m}{2})}$	Complexity
I	$(2^8, 72, 46)$	2^{80}	2^{31}	2^{111}
III	$(2^8, 112, 72)$	2^{128}	2^{42}	2^{170}
V	$(2^8, 148, 96)$	2^{176}	2^{52}	2^{228}

Our attack again relies on the sparseness property of the vinegar-vinegar quadratic part and the specific structure of the linear transformation $\mathbf{S}$, as per the *equivalent keys* key generation technique.

5 Implementation

5.1 Sparse $\mathcal{F}_{OV}$

To confirm the practicality of our attack in Sect. 3, we provide a verification script in MAGMA [5] where we implement the key generation of MQ-Sign-{S/R}S and then run the main algorithm for recovering the secret key from the public key as input. The running time of the attack on a laptop is 0.6 s for the proposed parameters for security level I, 2.3 s for security level III and 6.9 s for security level V. We also provide an equivalent SageMath [29] script that is slower.

5.2 Sparse $\mathcal{F}_V$

Complexity estimates in Sect. 4 show that MQ-Sign-SR falls below the required security level, but the attack is not practical for the chosen parameter sizes. We nevertheless implemented the attack as a proof-of-concept and to confirm practically our complexity estimations. The cost of enumeration is straightforward, but the second part of the attack involves Gröbner-based algorithms, whose complexity rely on heuristic assumptions of semi-regularity. Hence, our primary goal in this experimental work was to verify that the degree of regularity reached by the F4/F5 algorithm is estimated correctly. The verification script for this attack consists of generating the polynomial system in (9), fixing all variables in $\mathbf{z}_o$ and in the odd-indexed subset, and finally, solving the resulting system using the F4 algorithm implemented in MAGMA. When fixing the variables, we experimented both with a correct assignment that subsequently leads to a solution, and a random assignment that leads to an inconsistent system. As expected, there is no difference in the solving running times between the two cases.

Table 4. Experimental results of the direct attack.

Security level	Parameters (q, v, m)	D estimated	D reached	Runtime (s)	Memory (MB)
I	$(2^8, 72, 46)$	4	4	0.6	32
III	$(2^8, 112, 72)$	5	5	90.2	534
V	$(2^8, 148, 96)$	6			>32000

The results of our experiments are in Table 4. Most notably, we confirm that the degree of regularity reached during the execution of the algorithm matches the theoretical estimation. This holds for both security level I and III. For security level V, the degree of regularity is expected to be six, hence we could not perform the verification due to the high memory requirements. For further assurance, we verified our complexity estimation on other parameter sets that are not part of the MQ-Sign specification, but follow the usual UOV ratios. We conclude that the MQ instances that need to be solved in the second part of the algorithm behave as semi-regular instances and the complexity of finding a solution can reliably be estimated using the analysis in [2].

Verification scripts for both attacks outlined in this paper can be found at

https://github.com/mtrimoska/MQ-Sign-attack.

6 Impact on the MQ-Sign Variants

Both attacks presented in this paper rely on the specific structure of the linear transformation $\mathbf{S}$, as per the *equivalent keys* key generation technique. This technique is used in most modern UOV-based signature schemes, including MQ-Sign. If the equivalent keys structure is removed and $\mathbf{S}$ is a random affine map[2], this change of representation comes with additional memory cost. Specifically, Table 5 shows the impact of this modification on the secret key sizes, compared to the sizes reported in the MQ-Sign specification. The comparison is shown for the three MQ-Sign variants that are concerned by the two attacks proposed in this paper. The fourth variant, MQ-Sign-RR, is equivalent to the traditional UOV scheme and is not affected by our attacks. For this variant, the use of the equivalent keys structure of $\mathbf{S}$ is still a concern for side-channel attacks [1,21].

[2] This was suggested by the authors of MQ-Sign as a countermeasure when the attack in Sect. 3 was first announced.

Table 5. Size (in Bytes) of the secret key of MQ-Sign with and without the equivalent keys structure of $\mathbf{S}$.

Variant	Security Level					
	I		III		V	
	equivalent keys $\mathbf{S}$	random $\mathbf{S}$	equivalent keys $\mathbf{S}$	random $\mathbf{S}$	equivalent keys $\mathbf{S}$	random $\mathbf{S}$
MQ-Sign-SS	15 561	26 173	37 729	63 521	66 421	111 749
MQ-Sign-RS	133 137	143 749	485 281	511 073	1 110 709	1 156 037
MQ-Sign-SR	164 601	175 213	610 273	636 065	1 416 181	1 461 509

Furthermore, this countermeasure was shown to be insufficient for the variants where the vinegar-oil space is sparse. In subsequent work, Ikematsu, Jo, and Yasuda [15] propose an attack that does not rely on the equivalent structure of $\mathbf{S}$ and remains practical: it runs in no more than 30 min for all security levels.

For the MQ-Sign-SR variant, further research is needed to determine whether the sparseness of $\mathcal{F}_V$ can still be exploited in a similar manner when $\mathbf{S}$ is random.

7 Discussion on Using Sparse Matrices

MQ-Sign follows the UOV construction that is widely believed to be solid. Yet, as we have demonstrated, bad choices for optimization have significantly damaged its security. The aforementioned attacks were possible due to mainly two reasons. First, the *secret* polynomials were chosen sparse. Thus, we could derive more equations from the public key entries and their computation in Eq. (6) than there are secret key entries to obscure them. Second, the secret key polynomials were chosen so sparse, that half of the public key equations turned bilinear after fixing certain variables. The question that remains is whether we can still make use of sparseness to reduce the size of the (expanded) keys.

As an alternative, we could, instead of choosing sparse secret submatrices $\mathbf{F}_1^{(k)}$ and $\mathbf{F}_2^{(k)}$, choose the public $\mathbf{P}_1^{(k)}$ and $\mathbf{P}_2^{(k)}$ sparse. Our key-recovery attack does not work anymore, but, we would need to add more coefficients to the matrices, so that the strategy in Sect. 4 does not apply anymore.

The approach complements current UOV instantiations [4] which use key compression techniques. The authors of [4] expand the matrices $\mathbf{P}_1^{(k)}$ and $\mathbf{P}_2^{(k)}$ from a seed $\mathtt{seed}_{pk}$ and only store $cpk = (\mathtt{seed}_{pk}, \mathbf{P}_3^{(k)})$. Therefore, making these two matrices sparse will not result in a smaller compressed public key, but the size of the expanded secret key and the expanded public key would be reduced, which implies a lower overall storage requirement.

However, caution should be put into the choice of the sparse public matrices. The strategy of using "rotating diagonals" seems to work well with regards to the standard attacks against UOV analyzed in the specs. However, the sparse equations introduce enough structure to make a direct attack cheaper than in the no-sparse case. An option could be to slightly increase the number of non-zero coefficients in $\mathbf{P}_1^{(k)}$ and $\mathbf{P}_2^{(k)}$, enough to increase the cost of our attack or

a similar direct attack. This is of course an ad-hoc solution, and more scrutiny is required in order to determine whether a secure balance can be found that is a better solution than simply increasing the parameters. We leave this question as future work.

References

1. Aulbach, T., Campos, F., Krämer, J., Samardjiska, S., Stöttinger, M.: Separating oil and vinegar with a single trace side-channel assisted Kipnis-Shamir attack on UOV. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2023**(3), 221–245 (2023)
2. Bardet, M.: Étude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie. Ph.D. thesis, Université de Paris VI (2004)
3. Beullens, W.: Breaking rainbow takes a weekend on a laptop. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022. LNCS, vol. 13508, pp. 464–479. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-15979-4_16
4. Beullens, W., et al.: Oil and vinegar: modern parameters and implementations. IACR Trans. Cryptogr. Hardw. Embed. Syst. **321–365**, 2023 (2023)
5. Bosma, W., Cannon, J., Playoust, C.: The magma algebra system. I. The user language. J. Symbolic Comput. **24**(3–4), 235–265 (1997). Computational algebra and number theory (London, 1993)
6. Casanova, A., Faugère, J.-C., Macario-Rat, G., Patarin, J., Perret, L., Ryckeghem, J.: GeMSS. Technical report, National Institute of Standards and Technology (2020)
7. Chinese Association for Cryptologic Research (CACR). CACR post-quantum competition (2018)
8. Ding, J., et al.: Rainbow. Technical report, National Institute of Standards and Technology (2020)
9. Ding, J., Hu, L., Yang, B.-Y., Chen, J.-M.: Note on design criteria for rainbow-type multivariates. Cryptology ePrint Archive, Report 2006/307 (2006)
10. Faugère, J.-C.: A new efficient algorithm for computing Gröbner bases (F_4). J. Pure Appl. Algebra **139**, 61–88 (1999)
11. Faugère, J.-C.: A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5). In: Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation, ISSAC, pp. 75–83. ACM Press (2002)
12. I. O. for Standardization ISO/IEC JTC 1/SC 27 (WG2). Information security, cybersecurity and privacy protection: ISO/IEC WD 14888-4 Information technology - Security techniques - Digital signatures with appendix - Part 4: Stateful hash-based mechanisms. https://www.iso.org/standard/80492.html
13. Hulsing, A., et al.: SPHINCS+. NIST PQC Submission (2020)
14. Hülsing, A., Butin, D., Gazdag, S.-L., Rijneveld, J., Mohaisen, A.: XMSS: extended hash-based signatures. RFC 8391 (2018)
15. Ikematsu, Y., Jo, H., Yasuda, T.: A security analysis on MQ-Sign. In: Kim, H., Youn, J. (eds.) WISA 2023. LNCS, vol. 14402, pp. 40–51. Springer, Singapore (2024). https://doi.org/10.1007/978-981-99-8024-6_4
16. Kipnis, A., Patarin, J., Goubin, L.: Unbalanced oil and vinegar signature schemes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 206–222. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_15
17. Kipnis, A., Shamir, A.: Cryptanalysis of the oil and vinegar signature scheme. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 257–266. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0055733

18. Lyubashevsky, V., et al.: Crystals-dilithium. NIST PQC Submission (2020)
19. Mosca, M., Piani, M.: 2021 quantum threat timeline report (2022)
20. National Institute for Standards and Technology. Post-Quantum Cryptography Standardization (2017)
21. Park, A., Shim, K.-A., Koo, N., Han, D.-G.: Side-channel attacks on post-quantum signature schemes based on multivariate quadratic equations **2018**(3), 500–523 (2018). https://tches.iacr.org/index.php/TCHES/article/view/7284
22. Patarin, J.: The oil and vinegar signature scheme (1997)
23. Petzoldt, A.: Selecting and reducing key sizes for multivariate cryptography. Ph.D. thesis, Darmstadt University of Technology, Germany (2013)
24. Petzoldt, A., Bulygin, S., Buchmann, J.: CyclicRainbow - a multivariate signature scheme with a partially cyclic public key based on rainbow. Cryptology ePrint Archive, Report 2010/424 (2010)
25. Prest, T., et al.: FALCON. NIST PQC Submission (2020)
26. Quantum Resistant Cryptography Research Center. Korean post-quantum cryptographic competition (2022)
27. Shim, K.-A., Kim, J., An, Y.: MQ-Sign: a new post-quantum signature scheme based on multivariate quadratic equations: shorter and faster (2022). https://www.kpqc.or.kr/images/pdf/MQ-Sign.pdf
28. Tao, C., Petzoldt, A., Ding, J.: Efficient key recovery for all HFE signature variants. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021. LNCS, vol. 12825, pp. 70–93. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-84242-0_4
29. The Sage Developers. SageMath, the Sage Mathematics Software System (Version 9.5) (2022). https://www.sagemath.org
30. Yang, B.-Y., Chen, J.-M., Chen, Y.-H.: TTS: high-speed signatures on a low-cost smart card. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 371–385. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28632-5_27

Hash Your Keys Before Signing

BUFF Security of the Additional NIST PQC Signatures

Thomas Aulbach[1], Samed Düzlü[1( )], Michael Meyer[1], Patrick Struck[2], and Maximiliane Weishäupl[1]

[1] Universität Regensburg, Regensburg, Germany
{thomas.aulbach,samed.duzlu,maximiliane.weishaeupl}@ur.de,
michael@random-oracles.org
[2] Universität Konstanz, Konstanz, Germany
patrick.struck@uni-konstanz.de

Abstract. In this work, we analyze the so-called Beyond UnForgeability Features (BUFF) security of the submissions to the current standardization process of additional signatures by NIST. The BUFF notions formalize security against maliciously generated keys and have various real-world use cases, where security can be guaranteed despite misuse potential on a protocol level. Consequently, NIST declared the security against the BUFF notions as *desirable features*. Despite NIST's interest, only 6 out of 40 schemes consider BUFF security at all, but none give a detailed analysis. We close this gap by analyzing the schemes based on codes, isogenies, lattices, and multivariate equations. The results vary from schemes that achieve neither notion (e.g., Wave) to schemes that achieve all notions (e.g., Prov). In particular, we dispute certain claims by Squirrels and Vox regarding their BUFF security. Resulting from our analysis, we observe that three schemes (Cross, Hawk and Prov) achieve BUFF security without having the hash of public key and message as part of the signature, as BUFF transformed schemes would have. Hawk and Prov essentially use the lighter PS-3 transform by Pornin and Stern (ACNS'05). We further point out whether this transform suffices for the other schemes to achieve the BUFF notions, with both positive and negative results.

Keywords: Signature Schemes · BUFF · Additional Security Properties

1 Introduction

Nowadays, digital signature schemes are fundamental cryptographic primitives. They allow a signer Alice to generate a signature `sig` of a message `msg`, using her private key `sk`, such that anybody, using Alice's public key `pk`, can verify the validity of the signature. *Existential unforgeability under chosen message attacks* (EUF-CMA) has become the standard security notion for digital signature schemes. EUF-CMA secure schemes come with the guarantee that an adversary, seeing several message-signature pairs generated by Alice, cannot generate a new message-signature pair that is accepted as a signature by Alice.

M.-J. Saarinen and D. Smith-Tone (Eds.): PQCrypto 2024, LNCS 14772, pp. 301–335, 2024.
https://doi.org/10.1007/978-3-031-62746-0_13

Unforgeability is essential for digital signature schemes and in most use cases also sufficient. However, the EUF-CMA security notion only covers scenarios where Alice's key pair is honestly generated. Depending on the use case of a digital signature scheme, other attacks are possible which are not ruled out by using a signature scheme that is unforgeable. This led to the development of additional security notions: *exclusive ownership*, *message-bound signatures*, and *non-resignability*. In the following, we give high-level descriptions of these notions, covering the gist of each.

The first security notion, *exclusive ownership*, provides the adversary with a valid message-signature pair $(\mathtt{msg}, \mathtt{sig})$ under a public key $\mathtt{pk}$ and asks it to find a different public key $\overline{\mathtt{pk}}$ under which $(\mathtt{msg}, \mathtt{sig})$ remains a valid message-signature pair. The lack of exclusive ownership allows an adversary to "claim" signatures as its own by providing $\overline{\mathtt{pk}}$. The relevance can be seen by the real-world attack against the Let's Encrypt protocol, where an adversary can exploit such claimed signatures to obtain certificates for domains despite not owning them [1]. The notion comes in two flavors: the one just described, which is called *conservative exclusive ownership* (S-CEO), and *destructive exclusive ownership* (S-DEO), where the adversary needs to find a different message.

The second security notion, *message-bound signatures* (MBS), asks the adversary to come up with two messages $\mathtt{msg} \neq \overline{\mathtt{msg}}$, a signature $\mathtt{sig}$, and a public key $\mathtt{pk}$, such that both $(\mathtt{msg}, \mathtt{sig})$ and $(\overline{\mathtt{msg}}, \mathtt{sig})$ are valid message-signature pairs under $\mathtt{pk}$. Absence of this property allows adversaries to bypass non-repudiation: when the adversary is accused of having signed $\mathtt{msg}$, it can claim to have signed $\overline{\mathtt{msg}}$ instead. At the first glance, it seems that this should already be covered by standard EUF-CMA—finding $\overline{\mathtt{msg}}$ immediately yields a forged signature. The difference is that EUF-CMA is limited to honestly generated keys whereas the notion we describe here is more permissive by letting the adversary output an arbitrary public key, in particular, not constrained to be the outcome of the key generation algorithm.

The third security notion, *non-resignability* (NR), provides the adversary with a signature $\mathtt{sig}$ of an unknown message $\mathtt{msg}$ under some public key $\mathtt{pk}$ and asks the adversary for a different public key $\overline{\mathtt{pk}}$ and signature $\overline{\mathtt{sig}}$, such that $\overline{\mathtt{sig}}$ verifies correctly under $\overline{\mathtt{pk}}$ for the unknown[1] message $\mathtt{msg}$. Jackson et al. [24] showed that a resignable signature scheme, i.e., one for which the adversary can find $\overline{\mathtt{pk}}$ and $\overline{\mathtt{sig}}$ as described above, allows for attacks against the "Dynamically Recreatable Key" (DRKey) protocol [25]. Here, the adversary has to re-sign a message which contains a—to the adversary unknown—symmetric key.

The additional security properties exclusive ownership, message-bound signatures, and non-resignability were formalized in [14], which also provides a generic transformation—called the BUFF transform—to achieve them. Furthermore, the authors of [14] analyzed the signature schemes selected to be standardized by NIST: Dilithium [27], Falcon [33], and Sphincs$^+$ [23]. Dilithium was shown to achieve the notions and while Falcon does not, the authors of Falcon

[1] This part is crucial. If the adversary was to know the message $\mathtt{msg}$, it could generate a new key pair $(\overline{\mathtt{sk}}, \overline{\mathtt{pk}})$ and sign $\mathtt{msg}$ using $\overline{\mathtt{sk}}$ to obtain $\overline{\mathtt{sig}}$ and output $(\overline{\mathtt{pk}}, \overline{\mathtt{sig}})$.

announced to deploy the BUFF transform in the next update. For SPHINCS$^+$ it is informally argued that it achieves the additional security properties. While the notions are not a requirement in the ongoing NIST standardization process for digital signature schemes [29], the call-for-algorithms mentions them as "additional desirable security properties beyond standard unforgeability". Despite this, only six out of 40 submissions mention these security properties at all, but none give a detailed analysis. Thus, there is a gap with respect to the security achieved by the signature schemes submitted to the NIST standardization process. A gap that we (partially) close in this work.

A Note on Non-resignability. Note that the initial definition of non-resignability in [14] was identified to be flawed in [17]. The problem lies in the auxiliary information which allowed for an (arguably artificial) attack. New proposals for the definition of non-resignability are given in [17] and an updated version of [14]. However, it is unclear which definition will ultimately define non-resignability, and if the BUFF transform achieves either notion. Given these problems, we opt for a weaker form of non-resignability (wNR) in which there is no auxiliary information—thus considering a weaker notion than the one introduced in [17]. Nevertheless, we provide concrete attacks against most schemes. Thus, they are also vulnerable to any stronger form of non-resignability, in particular, to the existing ones [14,17].

1.1 Our Contribution

We analyze the submissions to the NIST standardization process for post-quantum signatures [29]. We focus on the submissions that are based on either codes, isogenies, lattices, or multivariate equations—excluding those for which attacks against EUF-CMA have been identified. More precisely, we analyze four code-based schemes (CROSS [3], LESS [2], MEDS [12], WAVE [4]), the sole isogeny-based scheme (SQISIGN [10]), five lattice-based schemes (HAETAE [11], HAWK [9], HUFU [35], RACCOON [15], SQUIRRELS [19]), and seven multivariate schemes (MAYO [6], PROV [22], QR-UOV [20], SNOVA [34], TUOV [16], UOV [7], VOX [31]). The results are summarized in Table 1.

In the following, we describe the main results. First, we remark that MBS is almost always satisfied and the security can be traced back to the security of the hash function. In the two cases of SQUIRRELS and WAVE, where MBS is not satisfied, the reason is the scheme-dependent construction of a public key that allows multiple messages to verify under the same signature. Note that the specification of SQUIRRELS claims MBS security, which our analysis refutes.

Secondly, we note that all schemes—except for SQISIGN, MEDS and LESS—satisfy either both S-CEO and S-DEO, or neither. Despite the general separation by [14], our results indicate that in practice, these two notions often behave similarly. In fact, both proofs and attacks usually use the same idea for S-CEO and S-DEO, where for S-DEO, one needs to be slightly more careful in the choices. One group among the schemes that satisfy these exclusive ownership notions achieves them by hashing the public key, together with the message, to generate

a target (resp. challenge), which the signature of the message corresponding to the given public key solves. In this way, any modification of the public key uncontrollably changes the target in a random manner. Then, the signature, which is required to be the same as the given one, cannot solve the new target, hence rendering the scheme secure. All schemes that do not satisfy exclusive ownership security are attacked by explicitly constructing new public keys which are compatible with the target generated independently of the public key, and the given signature. Differences between S-CEO and S-DEO can arise, when the message, but not the public key, is used to derive the target. Then it depends on the inherent properties of the scheme if different public keys can be constructed for the same (S-CEO) or a new (S-DEO) target value. An exception to the above rule is Cross, where the security reduces to solving an underdetermined system of linear equations.

Finally, we consider non-resignability. All schemes that satisfy wNR are also secure with respect to both exclusive ownership and MBS. However, there are schemes (SQIsign, Meds and Less), that satify S-DEO but not wNR. Indeed, we see a relationship to their exclusive ownership security: While fixing a signature fixes the public key in a certain sense, one can attack non-resignability by modifying both in a compatible manner, which does not require any knowledge about the message being signed. For the schemes that satisfy wNR, we see a similar argument as for exclusive ownership, namely that producing the target using a hash of the public key and the message, makes the target untraceable, even if one can control the signature. The exception, again, is Cross, where the security results from the Merkle tree structure and an underdetermined system of linear equations. The other schemes that do not satisfy wNR are attacked, as in the case of exclusive ownership, by explicit constructions. Neither of those attacks rely on any auxiliary information about the unknown message, which an adversary is provided in stronger versions of non-resignability.

From our results, we can deduce the following interesting conjecture. Even though [14] shows that in general the BUFF transform is necessary to achieve full BUFF security, it turns out that in practice, it is most often sufficient to use the PS-3 transform as suggested in [32]. That means, instead of using a mere hash-and-sign paradigm, one needs to hash the message *and* the public key, and then sign the hash value. The PS-3 transform is more lightweight than the BUFF transform as the latter requires to also append the hash value to the signature. One important caveat in this regard is that it is often *not* sufficient to hash only a part of the public key. Important examples where such an approach does not help to satisfy BUFF security are given by various multivariate schemes, e.g., Vox, where this approach is used explicitly to gain BUFF security, but is not sufficient.

Structure of the Analyses. The analyses presented in this work follow a common structure, which we explain briefly. To analyze the BUFF security, the relevant information is the structure of the public key and signature, and the verification algorithm. Those are introduced at the beginning of each section, followed by the analysis of S-CEO, S-DEO, MBS, and wNR. In Sect. 6 on multivariate

schemes, we give a more detailed general outline and give a generic proof of MBS and a generic attack on wNR, as the schemes allow such an all-encompassing formulation. The remaining analyses in the section follow the same structure.

Table 1. Overview of our results. A ✓ indicates that a signature scheme achieves a security notion, while a ✗ indicates that there is an attack. A ✦ indicates that we identified an attack that seems not to be relevant in practice. A superscript † indicates that the result disproves a claim made for the scheme. For LESS and MEDS, the results for S-CEO depend on the parameter sets.

Scheme	S-CEO	S-DEO	MBS	wNR	Type
CROSS [3]	✓	✓	✓	✓	Code (Sect. 3)
LESS [2]	✓ | ✗	✓	✓	✗	
MEDS [12]	✓ | ✗	✓	✓	✗	
WAVE [4]	✗	✗	✗	✗	
SQISIGN [10]	✦	✓	✓	✗	Isogeny (Sect. 4)
HAETAE [11]	✓	✓	✓	✓	Lattice (Sect. 5)
HAWK [9]	✓	✓	✓	✓	
HUFU [35]	✗	✗	✓	✗	
RACCOON [15]	✓	✓	✓	✓	
SQUIRRELS [19]	✗	✗	✗†	✗	
MAYO [6]	✗	✗	✓	✗	Multivariate (Sect. 6)
PROV [22]	✓	✓	✓	✓	
QR-UOV [20]	✗	✗	✓	✗	
SNOVA [34]	✗	✗	✓	✗	
TUOV [16]	✗	✗	✓	✗	
UOV [7]	✗	✗	✓	✗	
VOX [31]	✗†	✗†	✓	✗†	

1.2 Related Work

Unforgeability notions can be traced back to [21]. Exclusive ownership originates from [8,28], which introduces a specialized version under the name *Duplicate-Signature Key Selection*. A generalized version was developed in [32] which also coins the term exclusive ownership. Non-resignability was first mentioned in [24] though without a formal definition. Eventually, formal definitions of all beyond unforgeability properties (exclusive ownership, message-bound signatures, and non-resignability) were developed in [14], which also gives two generic transformations to achieve them.

2 Preliminaries

2.1 Notation

For integers m, n with $m < n$, we write $[m]$ and $[m, n]$ for the sets $\{1, 2, \ldots, m\}$ and $\{m, m+1, \ldots, n\}$, respectively. Throughout this work, $\mathtt{H}$ will denote a hash function (optionally with a subscript if multiple hash functions are used) which is often modeled as a random oracle [5]. For a matrix M, we denote the entries by m_{ij}. Similarly, for a vector x_i, its entries are denoted by $x_{i,j}$. We use ϑ to denote a generic bound (used for the lattice-based schemes).

2.2 Signature Schemes and Security Notions

A signature scheme Σ consists of three efficient algorithms:

`KGen:` the key generation gets a security parameter 1^λ as input and outputs a secret key `sk` along with a public key `pk`.
`Sign:` the signing algorithm gets a secret key `sk` and a message `msg` as input and outputs a signature `sig`.
`Verify:` the verification algorithm takes as input a public key `pk`, a message `msg`, and a signature `sig`, and it outputs a bit `v`.

A signature scheme is correct if, for any key pair $(\mathtt{sk}, \mathtt{pk}) = \mathtt{KGen}(1^\lambda)$, we have $\mathtt{Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{Sign}(\mathtt{sk}, \mathtt{msg})) = 1$ with overwhelming probability in the security parameter 1^λ.

In this work, we are using the security notions conservative/destructive exclusive ownership and message-bound signatures as formalized in [14], as well as a weaker form of non-resignability. Below we give the definitions. The corresponding security games S-CEO, S-DEO, MBS, and wNR, are shown in Fig. 1.

For conservative exclusive ownership, the adversary can obtain signatures for arbitrary messages and is then challenged to find a different public key that verifies one of the received message-signature pairs. Destructive exclusive ownership is similar to conservative exclusive ownership. The difference is that the adversary needs to find not just a different public key but also a different message that verify using one of the received signatures. The message-bound signature property guarantees that it is hard to find a signature that verifies two different messages under the same public key.

Definition 1. A signature scheme $\Sigma = (\mathtt{KGen}, \mathtt{Sign}, \mathtt{Verify})$ is said to have *conservative exclusive ownership*, *destructive exclusive ownership*, and *message-bound signatures* if for any efficient adversary $\mathcal{A}$, its probability in winning game S-CEO, S-DEO, and MBS, respectively, is negligible.

Non-resignability provides the adversary with a signature of an unknown message and asks to find a different public key and (not necessarily different) signature that verify the unknown message. We consider a slightly weaker form of non-resignability, which we call weak NR (wNR), which does not grant the adversary auxiliary information about the message. Note that for the majority

Game S-CEO/S-DEO

$\mathcal{Q} \leftarrow \emptyset$
$(\mathtt{sk}, \mathtt{pk}) \leftarrow \mathtt{KGen}()$
$(\overline{\mathtt{pk}}, \overline{\mathtt{msg}}, \overline{\mathtt{sig}}) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathtt{sk},\cdot)}(\mathtt{pk})$
$\mathtt{v}_1 \leftarrow \mathtt{Verify}(\overline{\mathtt{pk}}, \overline{\mathtt{msg}}, \overline{\mathtt{sig}})$
$\mathtt{v}_2 \leftarrow \mathsf{Valid}(\overline{\mathtt{msg}}, \overline{\mathtt{sig}})$
return $(\mathtt{v}_1 = 1 \wedge \mathtt{v}_2 = 1 \wedge \overline{\mathtt{pk}} \neq \mathtt{pk})$

Game wNR

$(\mathtt{sk}, \mathtt{pk}) \leftarrow \mathtt{KGen}()$
$\mathtt{msg} \leftarrow \mathcal{A}_0(\mathtt{pk})$
$\mathtt{sig} \leftarrow \mathtt{Sign}(\mathtt{sk}, \mathtt{msg})$
$(\overline{\mathtt{sig}}, \overline{\mathtt{pk}}) \leftarrow \mathcal{A}_1(\mathtt{pk}, \mathtt{sig})$
$\mathtt{v} \leftarrow \mathtt{Verify}(\overline{\mathtt{pk}}, \mathtt{msg}, \overline{\mathtt{sig}})$
return $(\overline{\mathtt{pk}} \neq \mathtt{pk} \wedge \mathtt{v} = 1)$

Game MBS

$(\mathtt{msg}, \overline{\mathtt{msg}}, \mathtt{sig}, \mathtt{pk}) \leftarrow \mathcal{A}()$
$\mathtt{v}_1 \leftarrow \mathtt{Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig})$
$\mathtt{v}_2 \leftarrow \mathtt{Verify}(\mathtt{pk}, \overline{\mathtt{msg}}, \mathtt{sig})$
return $(\mathtt{msg} \neq \overline{\mathtt{msg}} \wedge \mathtt{v}_1 = 1 \wedge \mathtt{v}_2 = 1)$

Oracle $\mathsf{Sign}(\mathtt{sk}, \mathtt{msg})$

$\mathtt{sig} \leftarrow \mathtt{Sign}(\mathtt{sk}, \mathtt{msg})$
$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\mathtt{msg}, \mathtt{sig})\}$
return sig

$\mathsf{Valid}(\overline{\mathtt{msg}}, \overline{\mathtt{sig}})$ in S-CEO

if $(\overline{\mathtt{msg}}, \overline{\mathtt{sig}}) \in \mathcal{Q}$
  return 1
return 0

$\mathsf{Valid}(\overline{\mathtt{msg}}, \overline{\mathtt{sig}})$ in S-DEO

if $\exists \mathtt{msg} \neq \overline{\mathtt{msg}}$ **s.t.** $(\mathtt{msg}, \overline{\mathtt{sig}}) \in \mathcal{Q}$
  return 1
return 0

Fig. 1. Security games S-CEO, S-DEO, MBS, and wNR, for signature schemes.

of signature schemes we give attacks against wNR which are also valid attacks against any stronger form of non-resignability, in particular, those including auxiliary information for the adversary.

Definition 2. A signature scheme $\Sigma = (\mathtt{KGen}, \mathtt{Sign}, \mathtt{Verify})$ is said to have *non-resignability* if for any efficient adversary $(\mathcal{A}_0, \mathcal{A}_1)$, where $\mathcal{A}_0$ outputs uniformly random message, its probability in winning game wNR is negligible.

We say that a signature scheme Σ has full BUFF security, if it satisfies S-CEO, S-DEO, MBS, and wNR.

2.3 Transformations

There are several generic transformations that turn a signature scheme and a hash function into a signature scheme that achieves the aforementioned BUFF notions. For this work, we mainly need two: The BUFF transform [14] (cf. Fig. 3) and the PS-3 transform [32] (cf. Fig. 2). The former was shown to achieve all the BUFF notions—based on the assumptions on the used hash function. The latter was shown to not achieve all notions, due to a property that [14] calls *weak keys*, i.e., public keys that verify multiple messages. Both transformations work by first computing the hash of the public key and message. This hash value is then

$\mathtt{KGen}^*()$	$\mathtt{Sign}^*(\mathtt{sk}, \mathtt{msg})$	$\mathtt{Verify}^*(\mathtt{pk}, \mathtt{msg}, \mathtt{sig})$
$(\mathtt{sk}, \mathtt{pk}) \leftarrow \mathtt{KGen}()$	$\mathtt{h} \leftarrow \mathtt{H}(\mathtt{msg}, \mathtt{pk})$	$\overline{\mathtt{h}} \leftarrow \mathtt{H}(\mathtt{msg}, \mathtt{pk})$
return $(\mathtt{sk}, \mathtt{pk})$	$\mathtt{sig} \leftarrow \mathtt{Sign}(\mathtt{sk}, \mathtt{h})$	$\mathtt{v} \leftarrow \mathtt{Verify}(\mathtt{pk}, \overline{\mathtt{h}}, \mathtt{sig})$
	return sig	**return** $\mathtt{v} = 1$

Fig. 2. The signature scheme PS-3$[\mathtt{H}, \Sigma] = (\mathtt{KGen}^*, \mathtt{Sign}^*, \mathtt{Verify}^*)$ constructed from a hash function H and a signature scheme $\Sigma = (\mathtt{KGen}, \mathtt{Sign}, \mathtt{Verify})$.

$\mathtt{KGen}^*()$	$\mathtt{Sign}^*(\mathtt{sk}, \mathtt{msg})$	$\mathtt{Verify}^*(\mathtt{pk}, \mathtt{msg}, (\mathtt{sig}, \mathtt{h}))$
$(\mathtt{sk}, \mathtt{pk}) \leftarrow \mathtt{KGen}()$	$\mathtt{h} \leftarrow \mathtt{H}(\mathtt{msg}, \mathtt{pk})$	$\overline{\mathtt{h}} \leftarrow \mathtt{H}(\mathtt{msg}, \mathtt{pk})$
return $(\mathtt{sk}, \mathtt{pk})$	$\mathtt{sig} \leftarrow \mathtt{Sign}(\mathtt{sk}, \mathtt{h})$	$\mathtt{v} \leftarrow \mathtt{Verify}(\mathtt{pk}, \overline{\mathtt{h}}, \mathtt{sig})$
	return $(\mathtt{sig}, \mathtt{h})$	**return** $(\mathtt{v} = 1 \wedge \mathtt{h} = \overline{\mathtt{h}})$

Fig. 3. The signature scheme Buff$[\mathtt{H}, \Sigma] = (\mathtt{KGen}^*, \mathtt{Sign}^*, \mathtt{Verify}^*)$ constructed from a hash function H and a signature scheme $\Sigma = (\mathtt{KGen}, \mathtt{Sign}, \mathtt{Verify})$.

signed[2] by the signature scheme. The difference is that the BUFF transform additionally appends this hash value to the signature (which PS-3 does not).

3 Code-Based Schemes

In this section, we analyze the code-based signature schemes. They rely on two distinct code-related problems: the more classical syndrome decoding problem (Cross and Wave), and the fairly new code equivalence problem (Meds and Less). Although they are based on the same underlying problem, Cross and Wave are still very different, and while Cross satisfies all BUFF properties, we show that Wave is vulnerable with respect to each of the notions. Wave fails to satisfy full BUFF security even after the PS-3 transform. We analyze Cross in Sect. 3.1 and Wave in Sect. 3.4. The two schemes based on code equivalences (Meds and Less) are very similar. We therefore only present Meds in full detail (in Sect. 3.2), as the analysis of Less (in Sect. 3.3) is almost verbatim the same. A surprising result of this analysis is that exclusive ownership notions are satisfied due to the inherent structure of the code equivalence problem. Indeed, for a given signature, there can essentially only be a single public key that verifies the message correctly. As this does not suffice to satisfy wNR, we show that using the PS-3 transform ensures full BUFF security for Meds and Less. Moreover, we note that PS-3-transformed Meds and Less can be considered to implement the full BUFF transform.

[2] Typically, the signature scheme itself first hashes the message. It is understood that in this case, the transformed scheme would in fact replace this hash operation, i.e., it signs $\mathtt{H}(\mathtt{msg}, \mathtt{pk})$ instead of $\mathtt{H}(\mathtt{H}(\mathtt{msg}, \mathtt{pk}))$.

3.1 CROSS

CROSS is a code-based signature scheme based on a zero-knowledge identification protocol, the security of which relies on the NP-complete *restricted syndrome decoding problem*. To increase the soundness of the Fiat-Shamir transform, CROSS incorporates Merkle trees into its signature definition. There are two variants of CROSS, R-SDP and R-SDP(G), where the latter restricts the problem to a subgroup G, to achieve shorter signature sizes. As the analysis regarding BUFF security is the same for both versions, we will only consider CROSS-R-SDP(G).

The protocol uses integers k, m, n, t, w, λ, prime numbers p and z, and an element $g \in \mathbb{F}_p^*$ of order z. The cyclic subgroup generated by g is denoted by $\mathbb{E} \subseteq \mathbb{F}_p^*$ and G denotes a subgroup of $\mathbb{E}^n$. Further, a pseudorandom number generator PRNG is used, which we assume to be ideal throughout our analysis, i.e., the outputs are random.

Key Pair. The public key consists of a tuple $(\mathtt{seed}_{\mathtt{pk}}, s)$ for $\mathtt{seed}_{\mathtt{pk}} \in \{0,1\}^\lambda$ and $s \in \mathbb{F}_p^{n-k}$. The *secret key* is given by $\mathtt{seed}_{\mathtt{sk}} \in \{0,1\}^\lambda$.

Signature. The signature of a message $\mathtt{msg}$ consists of

$$\mathtt{salt}||d_{01}||d_b||\mathtt{MerkleProofs}||\mathtt{SeedPath}||\mathtt{rsp}_0||\mathtt{rsp}_1$$

for $d_{01}, d_b \in \{0,1\}^\lambda$, $\mathtt{MerkleProofs} \in \{0,1\}^{l_m}$, $\mathtt{SeedPath} \in \{0,1\}^{l_s}$ with

$$l_m = 2\lambda \left(1 + (t-w)\log_2\left(\frac{t}{t-w}\right)\right),\ l_s = \lambda(t-w)\log_2\left(\frac{t}{t-w}\right),$$

$\mathtt{rsp}_0 \in (\mathbb{F}_p^n \times \mathbb{F}_z^m)^{t-w}$, and $\mathtt{rsp}_1 \in (\{0,1\}^\lambda)^{t-w}$.

Verify. Given a public key $\mathtt{pk} = (\mathtt{seed}_{\mathtt{pk}}, s)$, a message $\mathtt{msg}$, and a signature $\mathtt{sig} = (\mathtt{salt}||d_{01}||d_b||\mathtt{MerkleProofs}||\mathtt{SeedPath}||\mathtt{rsp}_0||\mathtt{rsp}_1)$, the verification algorithm is shown in Fig. 4.

S-CEO. Given a public key $\mathtt{pk} = (\mathtt{seed}_{\mathtt{pk}}, s)$, a message $\mathtt{msg}$, and a signature $\mathtt{sig}$ such that $\mathtt{Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig}) = 1$, we need to find a different public key $\overline{\mathtt{pk}} = (\mathtt{seed}_{\overline{\mathtt{pk}}}, \overline{s})$ such that $\mathtt{Verify}(\overline{\mathtt{pk}}, \mathtt{msg}, \mathtt{sig}) = 1$. Note that for $b[i] = 0$, the values t_i are computed as $x_i H^\top - \beta[i]s$, then hashed to $\mathtt{cmt}_0[i]$.

First, one sees that a change in the t_i leads to a change of $\mathtt{cmt}_0[i]$, consequently a change in d_0' and d_{01}', hence finally an invalid signature. Here, we use that changing the values in the Merkle tree results in another root, as long as the hash function is collision-resistant. Thus, any change of the public key that results in a change in any of the t_i will not be accepted in the verification.

$\texttt{Verify}(\texttt{pk}, \texttt{msg}, \texttt{sig})$

$(\texttt{seed}_{\texttt{pk}}, s) \leftarrow \texttt{pk}$
$(\texttt{salt}, d_{01}, d_b, \texttt{MerkleProofs}, \texttt{SeedPath}, \texttt{rsp}_0, \texttt{rsp}_1) \leftarrow \texttt{sig}$
$H, M_G \leftarrow \text{PRNG}(\texttt{seed}_{\texttt{pk}})$ // $H \in \mathbb{F}_p^{(n-k)\times k},\ M_G \in \mathbb{F}_z^{m\times n}$
$\beta \leftarrow \text{PRNG}(\texttt{H}(\texttt{H}(\texttt{msg})||d_{01}||\texttt{salt}))$ // $\beta \in (\mathbb{F}_p^*)^t$
$b \leftarrow \text{PRNG}(d_b)$ // $b \in \{0,1\}^t$ with hamming weight w
$(\texttt{seed}_0, \ldots, \texttt{seed}_{t-1}) \leftarrow \texttt{RebuildSeedTreeLeaves}(\texttt{SeedPath}, b, \texttt{salt})$
$j \leftarrow 0$
for $i = 0, \ldots, t-1$ **do**
  if $b[i] = 1$
    $(\texttt{cmt}_1[i], y_i) \leftarrow F_{b1}(\texttt{seed}_i, \texttt{salt}, M_G, \beta[i], i)$
  else
    $(\texttt{cmt}_0[i], \texttt{cmt}_1[i], y_i) \leftarrow F_{b0}(\texttt{rsp}_0[j], \texttt{rsp}_1[j], M_G, H, \beta[i], s, \texttt{salt}, i, j)$
    $j \leftarrow j + 1$
$d'_0 \leftarrow \texttt{RecomputeMerkleRoot}(\texttt{cmt}_0, \texttt{MerkleProofs}, b)$
$d'_1 \leftarrow \texttt{H}(\texttt{cmt}_1[i], \ldots, \texttt{cmt}_1[t-1]),\ d'_{01} = \texttt{H}(d'_0||d'_1),\ d'_b = \texttt{H}(y_0|| \ldots ||y_{t-1})$
if $(d_{01} = d'_{01} \wedge d_b = d'_b)$
  return 1
return 0

$F_{b1}(\texttt{seed}_i, \texttt{salt}, M_G, \beta[i], i)$

$\texttt{cmt}_1[i] \leftarrow \texttt{H}(\texttt{seed}_i||\texttt{salt}||i)$
$(u_i, \xi_i) \leftarrow \text{PRNG}(\texttt{seed}_i)$
// $u_i \in \mathbb{F}_p^n, \xi_i \in \mathbb{F}_z^m$
$\eta_i \leftarrow \xi_i M_G$
$e_i \leftarrow (g^{\eta_i[1]}, \ldots, g^{\eta_i[n]})$
return $(\texttt{cmt}_1[i], y_i = u_i + \beta[i]e_i)$

$F_{b0}(\texttt{rsp}_0[j], \texttt{rsp}_1[j], M_G, H, \beta[i], s, \texttt{salt}, i, j)$

$(y_i, \delta_i) \leftarrow \texttt{rsp}_0[j]$ // $(y_i, \delta_i) \in \mathbb{F}_p^n \times \mathbb{F}_z^m$
verify $\delta_i \in G$
$\sigma_i \leftarrow \delta_i M_G$
$v_i \leftarrow (g^{\sigma_i[1]}, \ldots, g^{\sigma_i[n]})$
$x_i \leftarrow v_i \star y_i$ // component-wise multiplication
$t_i = x_i H^\top - \beta[i]s$
$\texttt{cmt}_0[i] = \texttt{H}(t_i||\delta_i||\texttt{salt}||i)$
$\texttt{cmt}_1[i] = \texttt{rsp}_1[j]$
return $(\texttt{cmt}_0[i], \texttt{cmt}_1[i], y_i)$

Fig. 4. The verification algorithm of CROSS. Note that the PRNG generation of H, M_G, u_i and ξ_i is depicted in a simplified fashion; further observe that $\texttt{RecomputeMerkleRoot}$ only needs the subset $\{\texttt{cmt}_0[i] \mid i \text{ s.t. } b[i] = 0\}$ of commitments. We do not provide definitions for functions that are not relevant for the BUFF analysis.

Hence, we have to find $\texttt{pk} = (\texttt{seed}_{\texttt{pk}}, s) \neq (\texttt{seed}_{\texttt{pk}}, s) = \overline{\texttt{pk}}$ such that $\overline{t}_i = t_i$ holds for all i with $b[i] = 0$. Note that we can assume that b has roughly $t/2$ bits equal to 0 as it is generated with the PRNG. Then the problem corresponds to solving the system $t_i = \overline{x}_i \overline{H}^\top - \beta[i]\overline{s}$ of $t/2$ random equations in the single

indeterminate $\overline{s}$. If we choose $\mathtt{seed}_{\overline{\mathtt{pk}}} = \mathtt{seed}_{\mathtt{pk}}$, we have $\overline{H} = H$ and $\overline{x}_i = x_i$, thus there is no other solution than s. If we choose $\mathtt{seed}_{\overline{\mathtt{pk}}} \neq \mathtt{seed}_{\mathtt{pk}}$, we obtain a different pseudorandom matrix $\overline{H} \neq H$ and vector $\overline{x}_i \neq x_i$ and the probability that the resulting system is solvable is $\left(1/p^{n-k}\right)^{t/2}$. For all parameter sets of CROSS, this is less than $2^{-20\,000}$. Therefore, CROSS fulfills S-CEO security.

S-DEO. Given a public key $\mathtt{pk}$, a message $\mathtt{msg}$, and a signature $\mathtt{sig}$ such that $\mathtt{Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig}) = 1$, we need to find a second public key $\overline{\mathtt{pk}} \neq \mathtt{pk}$ and a second message $\overline{\mathtt{msg}} \neq \mathtt{msg}$ such that $\mathtt{Verify}(\overline{\mathtt{pk}}, \overline{\mathtt{msg}}, \mathtt{sig}) = 1$. Here, the same argument as in the S-CEO analysis is applicable. Even though the message can be changed, this brings no advantage to an adversary as it is directly hashed, so that the value of $\overline{\beta}$ cannot be controlled. Thus, the situation is again that $\overline{s}$ needs to be chosen such that $\overline{s} = (\overline{x}_i \overline{H}^\top - t_i) \cdot \overline{\beta}[i]^{-1}$ holds for all i with $b[i] = 0$. With the same argument as above, this implies that CROSS is S-DEO-secure.

MBS. One needs to find a public key $\mathtt{pk}$, two distinct messages $\overline{\mathtt{msg}} \neq \mathtt{msg}$, and a signature $\mathtt{sig}$, such that $\mathtt{Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig}) = 1$ and $\mathtt{Verify}(\mathtt{pk}, \overline{\mathtt{msg}}, \mathtt{sig}) = 1$. For different messages, but the same signature and public key, only the values for β differ in the computation of t_i for i such that $b[i] = 0$. This implies $\overline{t_i} \neq t_i$ and hence the verification fails, as long as the hash function is collision-resistant. Therefore, MBS security is given.

wNR. Given a public key $\mathtt{pk}$ and a signature $\mathtt{sig}$ to an unknown message $\mathtt{msg}$, one has to find another public key $\overline{\mathtt{pk}} \neq \mathtt{pk}$, and a signature $\overline{\mathtt{sig}}$ such that $\mathtt{Verify}(\overline{\mathtt{pk}}, \mathtt{msg}, \overline{\mathtt{sig}}) = 1$. For unknown messages, the values of β are also unknown. Thus, even though the public key and signature can be chosen freely, an attacker cannot know what to set them to, making this problem as hard as a random search of two hash values, each of size at least 256 bits, depending on the security level. Hence, the success probability is at most 2^{-512}. We conclude that CROSS is wNR-secure.

3.2 MEDS

MEDS is a signature scheme based on the difficulty of finding equivalences of matrix codes in the rank metric. It is constructed from a zero-knowledge identification protocol and involves a technique to increase the soundness and thereby reduce the signature size. The protocol uses integers m, n, s, t, a prime power q, and the field $\mathbb{F}_q$ with q elements. The hash function $\mathtt{H}$ maps to $\{0, \ldots, s\}^t$, its entries are denoted h_i. The standard form of a code is the unique generator matrix in row-reduced echelon form.

Key Pair. The public key consists of matrices $G_0, \ldots, G_s \in \mathbb{F}_q^{k \times nm}$, all in standard form. For $i = 0, \ldots, s$, let C_i denote the code generated by G_i. The secret key consists of code equivalence maps $\pi_{A_i,B_i} \colon C_0 \to C_i$ for $i = 1, \ldots, s$, where A_i and B_i are square matrices of the appropriate sizes. It holds that G_i is the standard form of $A_i G_0 B_i$.

Signature. The signature of a message $\mathtt{msg}$ to a public key $(G_0, \ldots, G_s)$ consists of (h, π_{i,h_i}), where $h = \mathtt{H}(\tilde{G}_0, \ldots, \tilde{G}_t, \mathtt{msg}) \in \{0,1\}^t$, and $\pi_{i,h_i} \colon G_{h_i} \to \tilde{G}_i$ are code equivalences, for $i = 1, \ldots, t$. The matrices $\tilde{G}_i$ are constructed as $\tilde{A}_i G_0 \tilde{B}_i$ using random matrices $\tilde{A}_i$ and $\tilde{B}_i$, for $i = 1, \ldots, t$.

Verify. The verification algorithm computes $\tilde{G}_i$ using $\pi_{i,h_i} G_{h_i}$ and checks if $h = \mathtt{H}(\tilde{G}_1, \ldots, \tilde{G}_t, \mathtt{msg})$ holds.

S-CEO. Let $(G_j)_j$ be a public key and $\mathtt{msg}$ be a message with signature $\mathtt{sig} = (h, \pi_{i,h_i})$. Fix an index i and set $j = h_i$. Then, $\pi_{i,h_i} G_j$ and $\tilde{G}_i$ define the same code. Thus, if $(G'_j)_j$ would be another public key accepting the same signature for the message $\mathtt{msg}$, we find that $\pi_{i,h_i}^{-1} \tilde{G}_i$ and G'_j both define the same code as G_j. Hence, $G_j = G'_j$ by normalization. Thus, a message-signature pair cannot be attacked if the following assumption holds: For each j there is an index i such that $h_i = j$. Conversely, suppose j^* is an index such that $h_i \neq j^*$ for all i. Then we may pick an arbitrary G'_{j^*} different from G_{j^*}, while setting $G'_j = G_j$ for all $j \neq j^*$. As G_{j^*} or G'_{j^*} are not used, the verification succeeds. If an index j^* exists, the new public key is constructed in constant time.

We conclude that a message-signature pair is vulnerable to an S-CEO attack, if and only if for the corresponding h there is such an index j which is not one of the components of h. Assuming that h is uniformly random, this translates to picking uniformly maps $\{1, \ldots, t\} \to \{0, \ldots, s-1\}$ which are non-surjective.

As any such choice depends on a query to a signature oracle, we bound the number of queries by 2^{64}, cf. [29, Section 4.B.2]. We say a parameter set is vulnerable against an S-CEO attacker if, with less than 2^{64} queries, the probability of finding a non-surjective mapping exceeds 50%. Conversely, we declare a parameter set to be secure if, after 2^{64} queries, the probability of finding a non-surjective map is still negligible.

To compute these probabilities, we define A_ℓ as the event that after ℓ queries, no non-surjective map has been found. It is easy to see that

$$1 - \left(1 - \left(\frac{s-1}{s}\right)^t\right)^\ell \leq \mathbb{P}(A_\ell) \leq 1 - \left(1 - s\left(\frac{s-1}{s}\right)^t\right)^\ell.$$

Using standard formulas and approximations for logarithm, we find that for

$$q \approx \frac{\log(2)}{\left(\frac{s-1}{s}\right)^t},$$

the probability of finding non-surjective maps exceeds $\frac{1}{2}$. As can be seen in Table 2, this shows that all but two parameter sets of MEDS are vulnerable to attacks. For the remaining two parameter sets, we can use the upper bound

$$\mathbb{P}(A_{2^\lambda}) \leq 2^\lambda s \left(\frac{s-1}{s}\right)^t,$$

which is valid if $s\left(\frac{s-1}{s}\right)^t$ is sufficiently small. The bounds are given in the final row of Table 2.

Table 2. The third row denotes the number of queries q such that the attack probability is above 50%. The probability in the final row denotes the chance of finding a message-signature pair that is vulnerable after 2^{64} queries.

Security Level	I	I	III	III	V	V
s	4	5	4	5	5	6
t	1152	192	608	160	192	112
Lower bound $\log_2(q)$	477	61	251	50	61	28
Success probability after 2^{64} queries	2^{-412}	≈ 1	2^{-186}	≈ 1	≈ 1	≈ 1

S-DEO. MEDS satisfies S-DEO as any change in the message yields a change in the hash h that is part of the signature, unless a collision of the hash is found.

MBS. MEDS satisfies MBS security trivially, if the hash function is collision-resistant, as distinct messages yield distinct hashes, contained in the signature.

wNR. MEDS does not satisfy wNR security. Indeed, given a public key $(G_i)_i$ and a signature $(h, (\pi_{i,h_i}))$ that verify an unknown message $\mathtt{msg}$, we can adapt the public key and the signature as follows. Pick arbitrary matrices $\overline{A}, \overline{B}$ of the correct size, apply to G_1 the transformation $\pi_{\overline{A},\overline{B}}$, and update this new generator matrix $\overline{G}_1$ as the first component in the public key. For each i such that $h_i = 1$, modify the function $\pi_{i,1}$ to $\pi_{i,1} \circ \pi^{-1}_{\overline{A},\overline{B}}$. The verification will succeed, as by construction, $\pi_{i,1} \circ \pi^{-1}_{\overline{A},\overline{B}} \overline{G}_1 = \pi_{i,1} G_1 = \tilde{G}_i$. Note that h in the signature is unchanged.

Remark 3. The signature scheme MEDS would additionally satisfy wNR, if in the signing process, h would be redefined as $h := \mathtt{H}(\tilde{G}_1, \ldots, \tilde{G}_t, \mathtt{msg}, \mathtt{pk})$, which corresponds to an application of PS-3. Indeed, as h itself is part of the signature, this change can be viewed as applying the BUFF transform to MEDS, making it secure against all BUFF notions.

3.3 LESS

LESS is, like MEDS, a signature scheme that relies on the code-equivalence problem and is based on a zero-knowledge identification protocol. Due to the strong similarity with MEDS, we do not provide all details. In short, LESS does not satisfy wNR, but satisfies S-DEO and MBS. Like in the analysis of MEDS, S-CEO security depends on the parameter set. The detailed results can be found in Table 3. Note that the second parameter set requires fewer queries than the security parameter and after 2^{64} queries, the success probability of an attack is 2^{-35}. While 2^{100} signature queries are too many, this parameter set seems to be an edge case which we cannot safely declare to be secure. Adding the public key in the hash computation makes LESS BUFF secure, as this is essentially the BUFF transform.

Table 3. The third row denotes the number of queries q such that the attack probability is above 50%. The probability in the final row denotes the chance of finding a message-signature pair that is vulnerable after 2^{64} queries.

Security Level	I	I	I	III	III	V	V
s	2	4	8	2	3	2	3
t	247	244	198	759	895	1352	907
Lower bound $\log_2(q)$	246	100	37	758	523	inf	530
Success probability after 2^{64} queries	2^{-182}	2^{-35}	≈ 1	2^{-694}	2^{-457}	≈ 0	2^{-464}

3.4 WAVE

Wave is a code-based signature scheme using the Hamming weight over the field $\mathbb{F}_3$. The security of Wave relies on the syndrome decoding problem and a scheme-specific problem regarding the indistinguishability of the public key.

The Hamming weight of a vector over $\mathbb{F}_3$ is denoted $|\_|$. Wave uses integer parameters n and k, which are the length and dimension of the codes, and ω, a target Hamming weight.

Key Pair. The public key is a matrix $M = M(R) \in \mathbb{F}_3^{k\times(n-k)}$, where $R \in \mathbb{F}_3^{(n-k)\times k}$ is a matrix and $M(R)$ is defined row-wise by

$$\begin{aligned}\mathsf{row}(M, 2i) &= \mathsf{col}(R, 2i) + \mathsf{col}(R, 2i+1)\\ \mathsf{row}(M, 2i+1) &= \mathsf{col}(R, 2i) - \mathsf{col}(R, 2i+1),\end{aligned}$$

for $0 \le i < \frac{k-1}{2}$, and if k is odd, then $\mathsf{row}(M, k-1) = -\mathsf{col}(R, k-1)$.

Signature. A signature $\mathtt{sig} = (\mathtt{salt}, s)$ consists of an element $s \in \mathbb{F}_3^k$ and a random value $\mathtt{salt}$. It defines a valid signature for a message $\mathtt{msg}$ and the public key $M = M(R)$, if and only if

$$|s| + |\mathtt{H}(\mathtt{msg}||\mathtt{salt}) + Rs| = \omega, \tag{1}$$

where $\mathtt{H}$ is a hash function that maps to $\mathbb{F}_3^{n-k}$.

Verify. The verification algorithm checks whether Eq. (1) holds.

S-CEO. Given a public key $M = M(R)$, any message $\mathtt{msg}$, and a signature $\mathtt{sig} = (\mathtt{salt}, s)$, we pick a matrix $\overline{R}$ such that $\overline{R}s = Rs$ but $\overline{R} \neq R$, for instance by extending s to a basis and defining $\overline{R}$ on the other basis vectors randomly. Then, Eq. (1) holds trivially with $\overline{R}$. Setting $\overline{M} = M(\overline{R})$ yields the new public key.

S-DEO. Given a public key $M = M(R)$, any message $\mathtt{msg}$, and a signature $\mathtt{sig} = (\mathtt{salt}, s)$, we randomly pick a new message $\overline{\mathtt{msg}} \neq \mathtt{msg}$ and compute $\overline{\mathtt{h}} := \mathtt{H}(\overline{\mathtt{msg}}||\mathtt{salt})$. We pick a vector $t \in \mathbb{F}_3^{n-k}$ such that $|\overline{\mathtt{h}} - t| = \omega - |s| =: \omega_s$, which can be done by choosing a random t' with hamming weight ω_s and setting $t = \overline{\mathtt{h}} - t'$. Then we choose $\overline{R}$ such that $\overline{R}s = t$ and set $\overline{M} = M(\overline{R})$. We find that Eq. (1) is satisfied, indeed, $|s| + |\overline{\mathtt{h}} - \overline{R}s| = |s| + |t'| = \omega$.

MBS. The MBS security of Wave can be attacked as follows. First, we pick random messages $\mathtt{msg} \neq \overline{\mathtt{msg}}$, and a random $\mathtt{salt}$. We compute $h = \mathtt{H}(\mathtt{msg}||\mathtt{salt})$ and $\overline{h} = \mathtt{H}(\overline{\mathtt{msg}}||\mathtt{salt})$. Then we need to find $t \in \mathbb{F}_3^{n-k}$, such that

$$\omega' := |h - t| = |\overline{h} - t| < \omega.$$

Indeed, if we have found such a t, we define s such that $|s| = \omega - \omega'$ and R such that $Rs = t$. Then, both messages are verified with the signature $\mathtt{sig} = (\mathtt{salt}, s)$ under the public key $M = M(R)$, as Eq. (1) is satisfied for both.

A simple but tedious combinatorial construction shows that such a t can be found in almost all cases.[3]

wNR. The attack against the S-CEO security of Wave applies to wNR, as no information about the message is required.

Remark 4. For Wave, we can show that applying the PS-3 transform *does not suffice* to achieve full BUFF security. Let us suppose that the value h in the signature is set to $\mathtt{H}(\mathtt{msg}||\mathtt{pk}||\mathtt{salt})$, and a signature $(\mathtt{salt}, s)$ is valid, if Eq. (1) holds with this h. Then, the resulting signature scheme is not MBS secure. Indeed, we begin by picking a matrix R from which $\mathtt{pk}$ is deduced and for which we know an efficient decoding algorithm $\mathcal{G}$. We set $\mathtt{salt}$ randomly. We pick random messages $\mathtt{msg}$ and $\overline{\mathtt{msg}}$ and compute $\mathtt{h} = \mathtt{H}(\mathtt{msg}||\mathtt{pk}||\mathtt{salt})$ and $\overline{\mathtt{h}}(\overline{\mathtt{msg}}||\mathtt{pk}||\mathtt{salt})$. As in the attack against MBS security for the original Wave scheme, we can find t such that $\omega' := |\mathtt{h} - t| = |\overline{\mathtt{h}} - t|$. We set $d = \omega - \omega'$ and run $\mathcal{G}$ with target vector t and Hamming weight d to obtain s. Then, $\mathtt{sig} = (\mathtt{salt}, s)$ is a valid signature for both $\mathtt{msg}$ and $\overline{\mathtt{msg}}$ under the public key $\mathtt{pk}$.

Despite this, the PS-3-transformed version of Wave does satisfy S-CEO, S-DEO, and wNR.

4 Isogeny-Based Schemes

In this section, we analyze the BUFF security of SQIsign [10], the sole isogeny-based signature scheme submitted to the NIST standardization process. We first give some background and notation that we require for the analysis.

[3] A Python script is provided online.

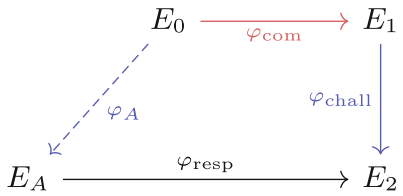


Fig. 5. The SQIsign protocol with three phases: commitment φ_{com}, challenge φ_{chall}, and response φ_{resp}.

Background and Notation. For elliptic curves E, E' over a finite field $\mathbb{F}_q$, an isogeny is a non-constant morphism $\varphi : E \to E'$ such that $\varphi(\infty_E) = \infty_{E'}$ for the respective points at infinity on E resp. E'. A subgroup G of order m uniquely (up to composition with isomorphisms) determines an isogeny $\varphi : E \to E/G$, where the kernel $\ker(\varphi) = G$ and the degree of φ is m. Such a subgroup can be described by a single point $K \in E$ of order m, i.e., $G = \langle K \rangle$. SQIsign uses a compressed representation of subgroups: Given a deterministic basis (P, Q) of the m-torsion subgroup $E[m]$, we can represent a suitable point as $K = P + [s]Q$ or $K = [s]P + Q$ for an $s \in \mathbb{Z}/m\mathbb{Z}$. Hence, given s and a decision bit $b \in \{0, 1\}$, we can compute $K = P + [s]Q$, where b indicates whether P and Q need to swapped prior to computing K. All occurring values s and b (with indices) will be of this form, and we refer to this computation as $\texttt{Decompress}_{P,Q}(s, b)$, where b can be omitted if no point swap is necessary. Each isogeny $\varphi : E \to E'$ has a unique dual isogeny $\widehat{\varphi} : E' \to E$ such that the composition $\widehat{\varphi} \circ \varphi$ resp. $\varphi \circ \widehat{\varphi}$ is the multiplication-by-m map on E resp. E'. We will only use supersingular curves E over $\mathbb{F}_{p^2}$ for a large prime p.

4.1 SQIsign

SQIsign applies the Fiat-Shamir transform to an identification protocol based on isogenies. Following Fig. 5, we define a public starting curve E_0, and the prover computes a secret isogeny $\varphi_A : E_0 \to E_A$, where E_A is published. The prover commits to the codomain E_1 of the commitment isogeny $\varphi_{\text{com}} : E_0 \to E_1$, followed by the challenger providing a challenge isogeny $\varphi_{\text{chall}} : E_1 \to E_2$. The prover answers with an isogeny $\varphi_{\text{resp}} : E_A \to E_2$. For the computation and the zero-knowledge property of φ_{resp} we refer to the SQIsign specification [10]. The standard Fiat-Shamir transform turns this protocol into a non-interactive signature scheme. We note that, due to the exponentially large challenge space, a single round of the protocol suffices.

Key Pair. For a fixed supersingular curve E_0 over $\mathbb{F}_{p^2}$ of known endomorphism ring, a secret key is an isogeny $\varphi_A : E_0 \to E_A$. The public key is given by E_A.

Signature. A signature consists of compressed descriptions of the isogenies φ_{resp} and φ_{chall}. For fixed positive integers e, f, g, n with $e = nf$ it is of the form

$$\texttt{sig} = (b, s^{(1)}, \ldots, s^{(n)}, r, b_2, s_2, b_3, s_3),$$

$\texttt{Verify}(\texttt{pk}, \texttt{msg}, \texttt{sig})$

1 : $(b, s^{(1)}, \ldots, s^{(n)}, r, b_2, s_2, b_3, s_3) \leftarrow \texttt{sig}$
2 : $E^{(1)} \leftarrow \texttt{pk}$
3 : $(P^{(1)}, Q^{(1)}) \leftarrow \texttt{FindBasis}_{2^f}(E^{(1)})$
4 : $K^{(1)} \leftarrow \texttt{Decompress}_{P^{(1)},Q^{(1)}}(s^{(1)}, b)$
5 : **for** $j = 1, \ldots, n-1$ **do**
6 : $\quad \varphi^{(j)} : E^{(j)} \to E^{(j+1)} = E/\langle K^{(j)} \rangle$
7 : $\quad Q^{(j+1)} \leftarrow \varphi^{(j)}(Q^{(j)})$
8 : $\quad P^{(j+1)} \leftarrow \texttt{CompleteBasis}_{2^f}(E^{(j+1)}, Q^{(j+1)})$
9 : $\quad K^{(j+1)} \leftarrow \texttt{Decompress}_{P^{(j+1)},Q^{(j+1)}}(s^{(j+1)})$
10 : $\varphi^{(n)} : E^{(n)} \to E^{(n+1)} = E/\langle K^{(n)} \rangle$
11 : $Q', E_1 = \texttt{Decompress}-$
$\quad \texttt{Challenge}(E^{(n+1)}, b_2, s_2, b_3, s_3)$
12 : **if** $[r]Q' = \mathcal{H}(\texttt{msg}, E_1)$
13 : $\quad$ **return** 1
14 : **return** 0

Fig. 6. Verification algorithm of SQIsign.

where $b, b_2, b_3 \in \{0, 1\}$, $s^{(j)}, s_2 \in \mathbb{Z}/2^f\mathbb{Z}$, $s_3 \in \mathbb{Z}/3^g\mathbb{Z}$, and $r \in \mathbb{Z}/2^f 3^g\mathbb{Z}$, following the notation from [13].

Verify. The verification algorithm, described in Fig. 6, consists of three parts. The most relevant part for the following discussion is the recomputation of $\varphi_{\text{resp}} : E_A \to E_2$ through a chain of n isogenies $\varphi^{(j)}$ of degree 2^f. Each isogeny $\varphi^{(j)}$ is determined by a kernel generator $K^{(j)}$. We compute these $K^{(j)}$ by deterministically sampling a basis $(P^{(j)}, Q^{(j)})$ of $E^{(j)}[2^f]$ through `FindBasis` if no point is given resp. `CompleteBasis` if $Q^{(j)}$ is given, and running `Decompress` with input $s^{(j)}$ (and b for $j = 1$). In particular, for $j > 1$, only $P^{(j)}$ is sampled, while we get $Q^{(j)} = \varphi^{(j-1)}(Q^{(j-1)})$, such that $Q^{(j)}$ generates the kernel of the dual isogenies $\widehat{\varphi^{(j-1)}}$. Therefore, we compute φ_{resp} through the following chain:

$$E_A = E^{(1)} \xrightarrow{\varphi^{(1)}} E^{(2)} \xrightarrow{\varphi^{(2)}} E^{(3)} \xrightarrow{\varphi^{(3)}} \cdots \xrightarrow{\varphi^{(n)}} E^{(n+1)} = E_2$$

In the second step, summarized in `DecompressChallenge`, we recompute the dual $\widehat{\varphi_{\text{chall}}} : E_2 \to E_1$ of order $D_{\text{chall}} = 2^f 3^g$ using `FindBasis` and `Decompress` with input (b_2, s_2, b_3, s_3). For a deterministically sampled point $Q'' \in E_2$ of order D_{chall} that is linearly independent of $\ker(\widehat{\varphi_{\text{chall}}})$, it computes $Q' \leftarrow \widehat{\varphi_{\text{chall}}}(Q'')$. Furthermore, this function verifies that the composition $\widehat{\varphi_{\text{chall}}} \circ \varphi_{\text{resp}}$ is cyclic.

The final step verifies that $[r]Q'$ corresponds to the kernel generator of the challenge isogeny, i.e. $[r]Q' = \mathcal{H}(\texttt{msg}, E_1)$. The function $\mathcal{H}$ is defined to first compute $a = \texttt{H}(\texttt{msg}, j(E_1)) \in \mathbb{Z}/D_{\text{chall}}\mathbb{Z}$ for a hash function `H` and the j-invariant $j(E_1)$, and output $R_1 + [a]S_1$ with a deterministic basis (R_1, S_1) of $E_1[D_{\text{chall}}]$.

S-CEO. Let `sig` be a valid signature for $\texttt{pk} = E_A$ and `msg`, i.e., $\texttt{Verify}(\texttt{pk}, \texttt{msg}, \texttt{sig}) = 1$. Our aim is to construct a public key $\overline{\texttt{pk}} = E_{A'} \neq E_A$ such that $\texttt{Verify}(\overline{\texttt{pk}}, \texttt{msg}, \texttt{sig}) = 1$. This amounts to finding $E_{A'}$ for which the compression in `sig` describes an isogeny $\psi_{\text{resp}} : E_{A'} \to E_2$ that has the same codomain E_2.

In this case, the second and third step of the verification are the same as when running `Verify(pk, msg, sig)`.

A naive way to find such a $E_{A'}$ is to compute random 2^e-isogenies $\psi' : E_2 \to E_{A'}$ and check if $(b, s^{(1)}, \ldots, s^{(n)})$ generates an isogeny $\psi_{\text{resp}} : E_{A'} \to E_2$ mapping to the correct E_2. However, the fact that we know several curves on the path between E_A and E_2 from `sig` allows for an easier S-CEO attack as follows:

1. Find $\widetilde{\psi}^{(1)} : E^{(2)} \to E_{A'}$ of degree 2^f with `FindBasis` and `Decompress`$(s^{(1)}, b)$ generating the 2^f-isogeny $\psi^{(1)} : E_{A'} \to E^{(2)}$ with the desired codomain.
2. Ensure that the following 2^f-isogenies satisfy $\psi^{(j)} = \varphi^{(j)}$ for $j > 1$, and hence ψ_{resp} maps to E_2.

Explicitly generating $E_{A'}$ in the first step seems infeasible, hence we resort to a search approach, going through all 2^f suitable isogenies $\widetilde{\psi}$. We require that the deterministic basis $(\widetilde{P}, \widetilde{Q})$ of $E_{A'}[2^f]$ and $b, s^{(1)} \in$ `sig` construct a suitable kernel generator $\widetilde{K}$ such that $\psi^{(1)} : E_{A'} \to E_{A'}/\langle \widetilde{K} \rangle = E^{(2)}$. Since there are $3 \cdot 2^{f-1}$ isogenies of degree 2^f starting from $E_{A'}$ and `sig` determines exactly one of these, the success probability for this step, given `sig`, is $1/(3 \cdot 2^{f-1})$. Thus, we can expect to find a suitable curve $E_{A'}$ with a probability of roughly 50%.

Assuming we found a suitable $E_{A'}$, we obtain a basis $(P^{(2)}, \widetilde{Q}^{(2)})$ of $E^{(2)}[2^f]$, where $\widetilde{Q}^{(2)} = \psi^{(1)}(\widetilde{Q})$. In contrast, a verification starting from E_A obtains the basis $(P^{(2)}, Q^{(2)})$ with the same sampled point $P^{(2)}$, but $Q^{(2)} = \varphi^{(1)}(Q^{(1)})$. Since the dual isogenies of $\varphi^{(1)}$ and $\psi^{(1)}$ are not equal, we have $\widetilde{Q}^{(2)} \notin \langle Q^{(2)} \rangle$. For the second attack step, we require for $j > 1$ that

$$\langle P^{(j)} + [s^{(j)}]Q^{(j)} \rangle = \langle P^{(j)} + [s^{(j)}]\widetilde{Q}^{(j)} \rangle.$$

All following steps trivially succeed if $s^{(j)} = 0$ for all $j > 1$. Furthermore, if $[2^k]Q^{(2)} = [2^k]\widetilde{Q}^{(2)}$ for $0 < k < f$, we succeed if $s^{(j)} \equiv 0 \mod 2^k$ for all $j > 1$. Even though the attack can only succeed if the signature values $s^{(j)}$ for $j > 1$ have a very special shape, it appears infeasible to enumerate all such possibilities, and compute an explicit success probability.

Instead, we implemented this attack using the AprèsSQI software [13], which closely follows the NIST submission of SQIsign.[4] For reduced parameters that allow feasible running times, i.e., a 36-bit prime p and $f \in \{7, 8, 9, 10\}$, our implementation suggests that the probability of a given $(s^{(1)}, \ldots, s^{(n)})$ to be vulnerable to this attack is below 2^{-f}. If we conjecture that this behavior scales to the SQIsign parameter sizes featuring $f = 75, 97, 145$ for NIST-I/III/V, this means that for each given `sig`, the S-CEO attack has a search complexity of $O(2^f)$ and success probability of 2^{-f}. Although we can conjecture that this attack does not break S-CEO security, we emphasize that better attack avenues might exist, and our probability estimations can only be viewed as a lower bound.

Remark 5. The probability and effort for a possible attack depend on the size of f. E.g., the SQIsign variant AprèsSQI [13] proposes much larger values of f, which push the success probability below the probability of breaking EUF-CMA.

[4] The implementation is available online.

S-DEO. In contrast to S-CEO, we additionally need to find a message $\overline{\texttt{msg}} \neq \texttt{msg}$ such that $\texttt{Verify}(\overline{\texttt{pk}}, \overline{\texttt{msg}}, \texttt{sig}) = 1$. Thus, we can only repeat the S-CEO attack above if the challenge curves E_2 resp. E_2' when signing $\texttt{msg}$ with $\texttt{pk}$ resp. $\overline{\texttt{msg}}$ with $\overline{\texttt{pk}}$ are equal, requiring $\mathcal{H}(\texttt{msg}, E_1) = \mathcal{H}(\overline{\texttt{msg}}, E_1)$, and therefore a hash collision of $\texttt{H}$ modulo D_{chall}.

If $\mathcal{H}(\texttt{msg}, E_1) \neq \mathcal{H}(\overline{\texttt{msg}}, E_1)$, i.e. $E_2' \neq E_2$, this attack is not available, hence we can only pick a random public key $\overline{\texttt{pk}}$. During verification, after recomputing ψ_{resp} and running $\texttt{DecompressChallenge}$, we end up at $E_1' \neq E_1$, such that the check $[r]Q' = \mathcal{H}(\overline{\texttt{msg}}, E_1')$ only succeeds with negligible probability $1/D_{\text{chall}}$. Therefore SQIsign is S-DEO-secure.

MBS. Assume that we have a valid signature $\texttt{sig}$ for $\texttt{pk}$ and $\texttt{msg}$, i.e., $\texttt{Verify}(\texttt{pk}, \texttt{msg}, \texttt{sig}) = 1$, and a message $\overline{\texttt{msg}} \neq \texttt{msg}$ such that $\texttt{Verify}(\texttt{pk}, \overline{\texttt{msg}}, \texttt{sig}) = 1$. In both verification runs, the first and second step that recompute φ_{resp} and $\widehat{\varphi_{\text{chall}}}$ are equal. In the last step, both runs compute $Q' \leftarrow \widehat{\varphi_{\text{chall}}}(Q'')$ and verify that $[r]Q' = \mathcal{H}(\texttt{msg}, E_1)$ resp. $[r]Q' = \mathcal{H}(\overline{\texttt{msg}}, E_1)$. However, if verification for $\texttt{msg}$ and $\overline{\texttt{msg}}$ succeeds, we have $\texttt{H}(\texttt{msg}, j(E_1)) = \texttt{H}(\overline{\texttt{msg}}, j(E_1))$, yielding a hash collision of $\texttt{H}$ modulo D_{chall}. Since this probability is negligible, SQIsign is MBS-secure.

wNR. Given a public key $\texttt{pk}$ and a signature $\texttt{sig}$ for an unknown message $\texttt{msg}$, an attacker has to find a public key $\overline{\texttt{pk}} \neq \texttt{pk}$ and a signature $\overline{\texttt{sig}}$ such that $\texttt{Verify}(\overline{\texttt{pk}}, \texttt{msg}, \overline{\texttt{sig}}) = 1$. To construct $\overline{\texttt{pk}}$ and $\overline{\texttt{sig}}$, we run the first step of $\texttt{Verify}(\texttt{pk}, \texttt{msg}, \texttt{sig})$ to obtain the curve E_2. We choose a random 2^f-isogeny $\widehat{\psi}^{(n)} : E_2 \to \widetilde{E}^{(n)}$ such that the composition $\varphi_{\text{chall}} \circ \widehat{\psi}^{(n)}$ is cyclic. Starting from $j = n - 1$ in decreasing order, we then construct random 2^f-isogenies $\widehat{\psi}^{(j)} : \widetilde{E}^{(j+1)} \to \widetilde{E}^{(j)}$ such that the composition $\widehat{\psi} = \widehat{\psi}^{(1)} \circ \cdots \circ \widehat{\psi}^{(n)}$ is cyclic. For each of the $\widehat{\psi}^{(j)}$, we pick a point $R \in \widetilde{E}^{(j+1)}$ of order 2^f such that R is linearly independent of $\ker(\widehat{\psi}^{(j)})$. Therefore, $K^{(j)} = \widehat{\psi}^{(j)}(R)$ generates the kernel of the dual isogeny $\psi^{(j)}$ of $\widehat{\psi}^{(j)}$.

For the signature $\overline{\texttt{sig}}$ we use $\psi_{\text{resp}} = \psi^{(n)} \circ \cdots \circ \psi^{(1)}$ and the public key $\overline{\texttt{pk}} = E_{A'} = \widetilde{E}^{(1)}$. For a valid signature the kernel generator points $K^{(j)}$ have to be represented in a compressed form. To compute this representation, we follow the approach of SQIsign. For the deterministic basis $(\widetilde{P}^{(1)}, \widetilde{Q}^{(1)})$ this allows us to find $\widetilde{s}^{(1)}$ to get a suitable kernel generator $\widetilde{P}^{(1)} + [\widetilde{s}^{(1)}]\widetilde{Q}^{(1)})$ for $\psi^{(1)}$, potentially swapping $\widetilde{P}^{(1)}$ and $\widetilde{Q}^{(1)}$ by setting $\widetilde{b} = 1$, and $\widetilde{b} = 0$ otherwise. The following steps proceed analogously, computing $\widetilde{s}^{(j)}$ through discrete logarithms without requiring to swap points.

Since E_2 is the codomain of ψ and $\widehat{\varphi_{\text{chall}}} \circ \psi$ is cyclic by construction, we can reuse the values r, b_2, s_2, b_3, s_3 for the second and third step of the verification of $\overline{\texttt{sig}}$. Hence, we have constructed a public key $\overline{\texttt{pk}} \neq \texttt{pk}$ and signature $\overline{\texttt{sig}} \neq \texttt{sig}$ of the form $\overline{\texttt{sig}} = (\widetilde{b}, \widetilde{s}^{(1)}, \ldots, \widetilde{s}^{(n)}, r, b_2, s_2, b_3, s_3)$ such that $\texttt{Verify}(\overline{\texttt{pk}}, \texttt{msg}, \overline{\texttt{sig}}) = 1$ without requiring knowledge of $\texttt{msg}$.

Remark 6. For SQIsign, the PS-3 transform suffices to achieve full BUFF security. In this case, the signer computes the challenge generator through

$\mathcal{H}(\mathtt{msg}, \mathtt{pk}, E_1)$, which uses the hash value $a = \mathtt{H}(\mathtt{msg}, \mathtt{pk}, j(E_1)) \in \mathbb{Z}/D_{\text{chall}}\mathbb{Z}$ as described above. This means that $r \in \mathtt{sig}$, which satisfies $[r]Q' = \mathcal{H}(\mathtt{msg}, \mathtt{pk}, E_1)$ for a deterministic point Q', can be viewed as an encoding of the hash value $\mathtt{H}(\mathtt{msg}, \mathtt{pk}, j(E_1))$, resembling the BUFF transform.

In this case, the problem to solve S-CEO is equivalent to the description of C-DEO above. For wNR, the PS-3 transform implies that the curve E_2 in $\mathtt{Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig})$ is not a valid challenge curve in $\mathtt{Verify}(\overline{\mathtt{pk}}, \mathtt{msg}, \overline{\mathtt{sig}})$. Attacking wNR thus requires to pick $\overline{\mathtt{pk}}, \overline{\mathtt{sig}}$ and hope for $[r]Q' = \mathcal{H}(\overline{\mathtt{msg}}, E_1')$ to hold for the chosen r, which has negligible success probability.

5 Lattice-Based Schemes

The lattice-based schemes we deal with in this section can be divided into two groups: Raccoon and Haetae, which are closely related to Dilithium; Hawk, HuFu, and Squirrels, which follow a GPV-like approach. For Raccoon and Haetae, we give an outline of the analysis from [14] in Sect. 5.4. The cases of Hawk (Sect. 5.1), HuFu (Sect. 5.2), and Squirrels (Sect. 5.3) are quite different, and we do a hands-on analysis. The results turn out to differ case by case. While Hawk achieves full BUFF security and HuFu only lacks wNR security, Squirrels is insecure with respect to all notions. We remark that a PS-3-transformed HuFu would satisfy all BUFF security notions. Finally, Squirrels is vulnerable even after the PS-3 transform and only the full BUFF transform could achieve all notions.

5.1 HAWK

Hawk applies a GPV-like approach. It uses module lattices and its security is based on the *One More Approximate Shortest Vector* problem [18].

Key Pair. Consider the number field $K_n = \mathbb{Q}[X]/(X^n+1)$ and its ring of integers $R_n = \mathbb{Z}[X]/(X^n + 1)$ for $m \in \mathbb{N}$ and $n = 2^m$. The secret key $\mathtt{sk}$ is a matrix

$$B = \begin{pmatrix} f & F \\ g & G \end{pmatrix} \in \mathrm{GL}_2(R_n),$$

and the public key $\mathtt{pk} = (q_{00}, q_{01}) \in R_n^2$ is computed from

$$Q = B^*B = \begin{pmatrix} q_{00} & q_{01} \\ q_{10} & q_{11} \end{pmatrix}.$$

The matrix Q induces the norm $\|\cdot\|_Q : K_n^2 \to \mathbb{Q},\ f \mapsto \sqrt{\frac{1}{n}\mathrm{Tr}(f^*Qf)}$. Since $Q = B^*B$, this norm fulfills $\|f\|_Q = \|Bf\|$ for all $f \in K_n^2$.

Signature. Hawk signatures consist of $\mathtt{sig} = (\mathtt{salt}, s_1)$ for $s_1 \in R_n$.

$\texttt{Verify}(\texttt{pk}, \texttt{msg}, \texttt{sig})$

1 : $(\texttt{salt}, s_1) \leftarrow \texttt{sig}$
2 : $(q_{00}, q_{01}) \leftarrow \texttt{pk}$
3 : $M \leftarrow \texttt{H}(\texttt{msg}||\texttt{H}(\texttt{pk}))$
4 : $(h_0, h_1) \leftarrow \texttt{H}(M||\texttt{salt})$
5 : $s_0 \leftarrow \left\lfloor \frac{h_0}{2} - \frac{q_{01}}{q_{00}} \left(\frac{h_1}{2} - s_1 \right) \right\rceil$
6 : $h \leftarrow (h_0, h_1),\ s \leftarrow (s_0, s_1)$
7 : $w \leftarrow h - 2s$
8 : **if** $\|w\|_Q \leq \vartheta$
9 : **return** 1
10 : **return** 0

Fig. 7. Verification algorithm of HAWK.

Verify. Given a public key $\texttt{pk} = (q_{00}, q_{01})$, a message $\texttt{msg}$, and a signature $\texttt{sig} = (\texttt{salt}, s_1)$, the verification algorithm is shown in Fig. 7.

S-CEO. Given a public key $\texttt{pk} = (q_{00}, q_{01})$, a message $\texttt{msg}$, and a signature $\texttt{sig} = (\texttt{salt}, s_1)$ such that $\texttt{Verify}(\texttt{pk}, \texttt{msg}, \texttt{sig}) = 1$, we need to find a public key $\overline{\texttt{pk}} \neq \texttt{pk}$ with $\texttt{Verify}(\overline{\texttt{pk}}, \texttt{msg}, \texttt{sig}) = 1$. Assuming $\texttt{H}$ to be a random oracle, choosing $\overline{\texttt{pk}} \neq \texttt{pk}$ implies $\overline{h}_1 \neq h_1$ and hence $\overline{w}_1 \neq w_1$. In order for an S-CEO attacker to be successful, $\|(0, \overline{w}_1)\|_{\overline{Q}} \leq \|(\overline{w}_0, \overline{w}_1)\|_{\overline{Q}} \leq \vartheta$ must hold. However, as $\overline{w}_1$ is random in R_n, the probability for this is negligible. Indeed, for the parameters in HAWK, a θ-ball is of size $2^{31.3}$, while the space of possible values is (much larger than) $2^{31 \cdot 256}$. So a random value will be in a θ-ball with probability about $2^{-31 \cdot 253}$.

S-DEO. Given a public key $\texttt{pk}$, a message $\texttt{msg}$, and a signature $\texttt{sig} = (\texttt{salt}, s_1)$ such that $\texttt{Verify}(\texttt{pk}, \texttt{msg}, \texttt{sig}) = 1$, we need to find $\overline{\texttt{pk}} \neq \texttt{pk}$ and $\overline{\texttt{msg}} \neq \texttt{msg}$ with $\texttt{Verify}(\overline{\texttt{pk}}, \overline{\texttt{msg}}, \texttt{sig}) = 1$. As the message is only used in the computation of h, the analysis works completely analogously as for S-CEO.

MBS. One needs to find a public key $\texttt{pk}$, distinct messages $\overline{\texttt{msg}} \neq \texttt{msg}$, and a signature $\texttt{sig} = (\texttt{salt}, s_1)$, s.t. $\texttt{Verify}(\texttt{pk}, \texttt{m}, \texttt{sig}) = 1$ for $\texttt{m} \in \{\texttt{msg}, \overline{\texttt{msg}}\}$. Assume one can find such $\texttt{pk}$, $\overline{\texttt{msg}}$, $\texttt{msg}$, and $\texttt{sig}$. Then, by definition of the verification, $\|w\|_Q, \|\overline{w}\|_Q \leq \vartheta$ and hence $\|w - \overline{w}\|_Q \leq 2\vartheta$ hold. Using the definition of B and $s_0 = \frac{h_0}{2} - \frac{q_{01}}{q_{00}} \left(\frac{h_1}{2} - s_1 \right) + \varepsilon$ for $\varepsilon \in [-\frac{1}{2}, \frac{1}{2})$ (and the analogue for $\overline{s}_0$), we obtain

$$\begin{aligned} \|w - \overline{w}\|_Q = \|B(w - \overline{w})\| &= \left\| \begin{pmatrix} f(h_0 - \overline{h}_0 - 2s_0 + 2\overline{s}_0) + F(h_1 - \overline{h}_1) \\ g(h_0 - \overline{h}_0 - 2s_0 + 2\overline{s}_0) + G(h_1 - \overline{h}_1) \end{pmatrix} \right\| \\ &= \left\| \begin{pmatrix} (h_1 - \overline{h}_1) \left(\frac{q_{01}}{q_{00}} f - F \right) + f(\varepsilon + \overline{\varepsilon}) \\ (h_1 - \overline{h}_1) \left(\frac{q_{01}}{q_{00}} g - G \right) + g(\varepsilon + \overline{\varepsilon}) \end{pmatrix} \right\|. \end{aligned}$$

The probability for this to be smaller than 2ϑ is negligible as $\frac{q_{01}}{q_{00}} f - F$ and $\frac{q_{01}}{q_{00}} g - G$ are fixed values, while $h_1 - \overline{h}_1$ and $\varepsilon + \overline{\varepsilon}$ are random. Hence, the advantage of any attacker against MBS-security of HAWK is similar to the S-CEO advantage.

wNR. Given a public key $\mathtt{pk}$ and a signature $\mathtt{sig} = (\mathtt{salt}, s_1)$ to an unknown message $\mathtt{msg}$, one has to find $\overline{\mathtt{pk}} \neq \mathtt{pk}$ and a signature $\overline{\mathtt{sig}} = (\overline{\mathtt{salt}}, \overline{s}_1)$ (which may be the same as the given signature) such that $\mathtt{Verify}(\overline{\mathtt{pk}}, \mathtt{msg}, \overline{\mathtt{sig}}) = 1$. Independent of the choice of the public key $\overline{\mathtt{pk}} \neq \mathtt{pk}$, the value of $\overline{h}$ is unknown (as $\mathtt{msg}$ is) and as in the S-CEO analysis $\overline{w}_1 \neq w_1$ holds. Hence, it is infeasible to choose $\overline{s}_1$ in a way such that $\overline{w} = \overline{h} - 2\overline{s}$ is small in the $\overline{Q}$-norm. Indeed, $\overline{s}_1$ must be chosen so that $2\overline{s}$ is in the $\overline{Q}$-norm ball about $\overline{h}$, which amounts to the same probability as computed in the proof of S-CEO.

Remark 7. The Hawk specification [9] states that the design facilitates an application of the full BUFF transform. This is the case as the Hawk signature generation already computes $M = \mathtt{H}(\mathtt{msg}||\mathtt{H}(\mathtt{pk}))$, which—in the full BUFF transform—needs to be appended to the signature. In the given form, Hawk can be seen to apply the PS-3 transform, which does not in general guarantee the BUFF properties. However, our analysis shows that in the concrete case of Hawk, BUFF security is fulfilled for this weaker transform, i.e., an application of the full BUFF transform is not necessary, which avoids appending the hash value to a signature. This is especially interesting given the fact that Hawk is based on Falcon. Falcon does not use the public key to construct the target value and was proven to be S-CEO, S-DEO and wNR-insecure.

5.2 HuFu

HuFu applies the GPV approach. It uses unstructured lattices and is based on the short integer solution and learning with errors problems.

Key Pair. Consider a distribution χ over $\mathbb{Z}$, $m, n \in \mathbb{N}$, and $Q = pq$ for p, q some powers of 2. The secret key is a tuple of matrices $\mathtt{sk} = (S, E, L_{22}, L_{32}, L_{33})$ for $(S, E) \leftarrow \chi^{n \times m} \times \chi^{m \times m}$ and $L_{22} \in \mathbb{R}^{n \times n}$, $L_{32} \in \mathbb{R}^{m \times n}$, and $L_{33} \in \mathbb{R}^{m \times m}$. The public key is a pair $\mathtt{pk} = (\mathrm{seed}_{\hat{A}}, B = p \cdot I - (\hat{A}S + E))$ for $\hat{A} \in \mathbb{Z}_Q^{m \times n}$ generated using $\mathrm{seed}_{\hat{A}}$.

Signature. The signature $\mathtt{sig}$ of a message $\mathtt{msg}$ consists of a tuple $(\mathtt{salt}, s)$ for $s = \mathtt{Compress}(x_1, x_2)$, where $x_1 \in \mathbb{Z}^n$ and $x_2 \in \mathbb{Z}^m$.

Verify. Given a public key $\mathtt{pk} = (\mathrm{seed}_{\hat{A}}, B)$, a message $\mathtt{msg}$, and a signature $\mathtt{sig} = (\mathtt{salt}, s)$, the verification algorithm is shown in Fig. 8.

S-CEO. Given a public key $\mathtt{pk} = (\mathtt{seed}_{\hat{A}}, B)$, a message $\mathtt{msg}$ and a signature $\mathtt{sig} = (\mathtt{salt}, s)$ such that $\mathtt{Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig}) = 1$, we need to find a second public key $\overline{\mathtt{pk}} = (\overline{\mathtt{seed}}_{\hat{A}}, \overline{B})$ such that $\mathtt{Verify}(\overline{\mathtt{pk}}, \mathtt{msg}, \mathtt{sig}) = 1$. We choose $\overline{\mathtt{seed}}_{\hat{A}} = \mathtt{seed}_{\hat{A}}$, which expand to the same matrix $\hat{A}$. As $(\mathtt{salt}, s)$ is a valid signature, we know that $||(x_0, x_1, x_2)|| \leq \vartheta$, where $x_0 = (u - \hat{A}x_1 - Bx_2) \mod Q$. Thus, if we find $\overline{B}$ s.t. $\overline{x_0} = x_0$, we obtain $||(\overline{x_0}, x_1, x_2)|| = ||(x_0, x_1, x_2)|| \leq \vartheta$, which shows that S-CEO security is not given. In order to construct such a $\overline{B}$, first note that we can assume that there is at least one i such that $x_{2,i} \neq 0$, as

$\texttt{Verify}(\texttt{pk}, \texttt{msg}, \texttt{sig})$

1 : $(\texttt{salt}, s) \leftarrow \texttt{sig}$
2 : $(x_1, x_2) \leftarrow \texttt{Decompress}(s)$
3 : $(\texttt{seed}_{\hat{A}}, B) \leftarrow \texttt{pk}$
4 : $u \leftarrow \texttt{H}(\texttt{msg}, \texttt{salt})$
5 : $\hat{A} \leftarrow \texttt{XOF}(\texttt{seed}_{\hat{A}})$
6 : $x_0 \leftarrow (u - \hat{A}x_1 - Bx_2) \mod Q$
7 : **if** $||(x_0, x_1, x_2)|| \leq \vartheta$
8 : **return** 1
9 : **return** 0

Fig. 8. Verification algorithm of HuFu. Note that (x_0, x_1, x_2) denotes the vector obtained from concatenating x_0, x_1, and x_2 and $||\cdot||$ is the l_2-norm.

otherwise one can trivially choose $\overline{B} \neq B$ with the desired properties. Without loss of generality, we assume $x_{2,1} \neq 0$. Then we define $\overline{B} \neq B$ as follows: $\overline{b}_{11} = (b_{11} + x_{2,2})$, $\overline{b}_{12} = (b_{12} - x_{2,1})$, and $\overline{b}_{ij} = b_{ij}$ for all other i, j. It holds that $(\overline{B}x_2)_1 = (Bx_2)_1$. Thus $\overline{B}x_2 = Bx_2$ as only the first row differs for $\overline{B}$ and B. This implies $\overline{x_0} = x_0$.

S-DEO. Given a public key $\texttt{pk} = (\texttt{seed}_{\hat{A}}, B)$, a message $\texttt{msg}$, and a signature $\texttt{sig} = (\texttt{salt}, s)$ s.t. $\texttt{Verify}(\texttt{pk}, \texttt{msg}, \texttt{sig}) = 1$, we need to find a second public key $\overline{\texttt{pk}} \neq \texttt{pk}$ and a second message $\overline{\texttt{msg}} \neq \texttt{msg}$ s.t. $\texttt{Verify}(\overline{\texttt{pk}}, \overline{\texttt{msg}}, \texttt{sig}) = 1$. We choose again $\overline{\texttt{seed}}_{\hat{A}} = \texttt{seed}_{\hat{A}}$, which yield the same matrix $\hat{A}$. Further we choose $\overline{\texttt{msg}} \neq \texttt{msg}$ randomly and compute u and $\overline{u}$. If we find $\overline{B}$ such that $\overline{x_0} = \overline{u} - \hat{A}x_1 - \overline{B}x_2 = 0 \mod Q$, we obtain $||(\overline{x_0}, x_1, x_2)|| \leq ||(x_0, x_1, x_2)|| \leq \vartheta$. Then, we have $\texttt{Verify}(\overline{\texttt{pk}}, \overline{\texttt{msg}}, \texttt{sig}) = 1$ for $\overline{\texttt{pk}} = (\texttt{seed}_{\hat{A}}, \overline{B})$, which gives an attack against S-DEO security. A matrix $\overline{B}$ such that $\overline{B}x_2 = \overline{u} - \hat{A}x_1$ can be constructed if $\gcd(x_{2,i}) = 1$.[5] As $m \geq 768$, the coefficients of $x_2 \in \mathbb{Z}^m$ are coprime with overwhelming probability given by $\zeta(m)^{-1} \approx 1$.

MBS. One needs to find a public key $\texttt{pk} = (\texttt{seed}_{\hat{A}}, B)$, two distinct messages $\texttt{msg} \neq \overline{\texttt{msg}}$, and a signature $\texttt{sig} = (\texttt{salt}, s)$ such that $\texttt{Verify}(\texttt{pk}, \texttt{msg}, \texttt{sig}) = 1$ and $\texttt{Verify}(\texttt{pk}, \overline{\texttt{msg}}, \texttt{sig}) = 1$. Assume, we have found $\texttt{pk}, \texttt{msg}, \overline{\texttt{msg}}$, and $\texttt{sig} = (\texttt{salt}, s)$ with these properties. Then $||(x_0, x_1, x_2)||, ||(\overline{x_0}, x_1, x_2)|| \leq \vartheta$ and hence in particular $||x_0||, ||\overline{x_0}|| \leq \vartheta$. Observe that this implies $||u - \overline{u}|| = ||u - (\hat{A}x_1 - Bx_2) + (\hat{A}x_1 - Bx_2) - \overline{u}|| = ||x_0 - \overline{x_0}|| \leq ||x_0|| + ||\overline{x_0}|| \leq 2\vartheta$. As $u = \texttt{H}(\texttt{msg}, r)$ and $\overline{u} = \texttt{H}(\overline{\texttt{msg}}, r)$, the probability to find messages that yield u and $\overline{u}$ which are close to each other is negligible (near-collision resistance of the hash function).[6]

wNR. Given a public key $\texttt{pk} = (\texttt{seed}_{\hat{A}}, B)$ and a signature $\texttt{sig} = (\texttt{salt}, s)$ to an unknown message $\texttt{msg}$, one has to find another public key $\overline{\texttt{pk}} \neq \texttt{pk}$, and a signature $\overline{\texttt{sig}} = (\overline{\texttt{salt}}, \overline{s})$ such that $\texttt{Verify}(\overline{\texttt{pk}}, \texttt{msg}, \overline{\texttt{sig}}) = 1$. To do this, we can

[5] If $\gcd(x_{2,i}) = 1$, then $\langle x_2 \rangle$ is saturated. Equivalently, $\mathbb{Z}^m / \langle x_2 \rangle$ is free, hence x_2 is part of a basis, on which $\overline{B}$ can be defined according to the requirement.

[6] Near-collision resistance is a stronger form of collision resistance, where it is even hard to find inputs whose hash values are close (with respect to some norm).

$\texttt{Verify}(\texttt{pk}, \texttt{msg}, \texttt{sig})$

1 : $(\texttt{salt}, s) \leftarrow \texttt{sig}$
2 : $h \leftarrow \texttt{H}(\texttt{msg}||\texttt{salt})$
3 : $s' \leftarrow \texttt{Decompress}(s)$
4 : $c \leftarrow s' + h$
5 : **if** $c_n = \sum_{i=1}^{n-1} v_i \cdot c_i \mod p \quad \forall p \in P_\Delta$
6 : **if** $\|s'\|^2 \leq \lfloor \vartheta^2 \rfloor$
7 : **return** 1
8 : **return** 0

Fig. 9. Verification algorithm of SQUIRRELS.

proceed exactly as we did for S-CEO. Note that for the attack it is not necessary to know the message and we can choose $\overline{\texttt{sig}} = (\overline{\texttt{salt}}, \overline{s}) = (\texttt{salt}, s) = \texttt{sig}$.

Remark 8. We showed that HUFU only achieves MBS security. We observe, however, that by applying the PS-3 transform, i.e., changing the computation of $u = \texttt{H}(\texttt{msg}, \texttt{salt})$ to $u = \texttt{H}(\texttt{msg}, \texttt{pk}, \texttt{salt})$, full BUFF security can be achieved. This is the case, as the above change prevents an attacker to control x_0 by their choice of $\texttt{pk}$—any change to $\texttt{pk}$ also changes the value of u and hence h in an uncontrollable way. Using this, S-DEO, S-CEO, and wNR security can be proven, while the proof for MBS security given for unmodified HUFU still applies.

5.3 Squirrels

SQUIRRELS incorporates a GPV-like approach. It is based on unstructured lattices and uses lattices modulo various distinct primes simultaneously. The public key is composed of a single vector which is used to check if a target is contained in the lattice modulo each of the primes. r,s Let n and q be positive integers. The target determinant is denoted by $\Delta = \prod_{p \in P_\Delta} p$, for P_Δ a set of primes in $[2^{30}, 2^{31}]$. The hash function $\texttt{H}$ maps to $[0, \ldots, q-1]^{n-1} \times \{0\}$ viewed as an element in $\mathbb{Z}^n$ with last component being 0.

Key Pair. The SQUIRRELS secret key consists of a matrix $B \in \mathbb{Z}^{n \times n}$, which, by design, has a Hermite normal form

$$\mathrm{HNF}(B) = \begin{pmatrix} I_{n-1} & v_i^\mathsf{T} \\ 0 & \Delta \end{pmatrix}.$$

The resulting vector $v \coloneqq (v_i)_{i=1,\ldots,n-1}$ is the public key.

Signature. The signature of a message $\texttt{msg}$ for a public key v consists of $(\texttt{salt}, s)$ where $\texttt{salt}$ is a random string and $s = \texttt{Compress}(s')$ with $s' \in \mathbb{Z}^n$.

Verify. Given a public key $\texttt{pk} = v$, a message $\texttt{msg}$, and a signature $\texttt{sig} = (\texttt{salt}, s)$, the verification algorithm is described in Fig. 9.

In the analysis below, we write $c' \coloneqq (c_1, \ldots, c_{n-1})^\mathsf{T}$ for $c = (c_1, \ldots, c_n)^\mathsf{T}$ and $\langle \cdot, \cdot \rangle$ for the standard inner product. Note that in the search for elements $v \in \mathbb{Z}^{n-1}$ that satisfy a certain algebraic condition modulo Δ, it suffices to give $v \mod p$ for each $p \in P_\Delta$, by making use of the Chinese Remainder Theorem. We make use of this argument, without explicitly stating it again.

S-CEO. Given a public key $\mathtt{pk} = v$, a message $\mathtt{msg}$ and a signature $\mathtt{sig} = (\mathtt{salt}, s)$ such that $\mathtt{Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig}) = 1$, we need to find a distinct public key $\overline{\mathtt{pk}} = \overline{v}$ such that $\mathtt{Verify}(\overline{\mathtt{pk}}, \mathtt{msg}, \mathtt{sig}) = 1$. This translates to finding $\overline{v}$, which is in the kernel of $\langle c', \cdot \rangle - c_n : \mathbb{F}_p^{n-1} \to \mathbb{F}_p$ for all $p \in P_\Delta$. Note that $\dim_{\mathbb{F}_p}(\ker(\langle c', \cdot \rangle) - c_n) = n - 2$. Hence, for each p one can find an element $\overline{v}_p$ such that $\langle c', \overline{v}_p \rangle - c_n = 0 \mod p$. Then, $\overline{\mathtt{pk}} = \overline{v}$ is given by the $\overline{v}_p$.

S-DEO. Given a public key $\mathtt{pk} = v$, a message $\mathtt{msg}$, and a signature $\mathtt{sig} = (\mathtt{salt}, s)$ such that $\mathtt{Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig}) = 1$, we need to find a second public key $\overline{\mathtt{pk}} = \overline{v} \neq v$ and a second message $\overline{\mathtt{msg}} \neq \mathtt{msg}$ such that $\mathtt{Verify}(\overline{\mathtt{pk}}, \overline{\mathtt{msg}}, \mathtt{sig}) = 1$. For this, we choose a random $\overline{\mathtt{msg}} \neq \mathtt{msg}$ and compute $\overline{c} = s' + \mathtt{H}(\overline{\mathtt{msg}} || \mathtt{salt})$. Hence it is left to find $\overline{v}$ such that $\langle \overline{v}, \overline{c}' \rangle - \overline{c}_n = 0 \mod p$ holds for all $p \in P_\Delta$. For this, the same argument as for the S-CEO attack applies.

MBS. One needs to find a public key $\mathtt{pk} = v$, two distinct messages $\mathtt{msg} \neq \overline{\mathtt{msg}}$, and a signature $\mathtt{sig} = (\mathtt{salt}, s)$ such that $\mathtt{Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig}) = 1$ and $\mathtt{Verify}(\mathtt{pk}, \overline{\mathtt{msg}}, \mathtt{sig}) = 1$. For this, we choose s' such that $\|s'\|^2 < \lfloor \vartheta^2 \rfloor$ holds and compute $s = \mathtt{Compress}(s')$. We then set $\mathtt{sig} = (\mathtt{salt}, s)$ for some randomly chosen $\mathtt{salt}$. Further we consider two random messages $\overline{\mathtt{msg}} \neq \mathtt{msg}$ and compute $c = s' + \mathtt{H}(\mathtt{msg} || \mathtt{salt})$ and $\overline{c} = s' + \mathtt{H}(\overline{\mathtt{msg}} || \mathtt{salt})$. Hence it is left to find v such that $\langle v, c' \rangle - c_n = 0 \mod p$ and $\langle v, \overline{c}' \rangle - \overline{c}_n = 0 \mod p$ holds for all $p \in P_\Delta$. Consider for $p \in P_\Delta$ the map

$$f : \mathbb{F}_p^{n-1} \to \mathbb{F}_p^2,\ x \mapsto (\langle x, c' \rangle, \langle x, \overline{c}' \rangle)$$

and observe that $\dim_{\mathbb{F}_p}(\ker(f)) = n - 3$. Hence, we can find v_p with the desired properties, which constitutes v.

Remark 9. In the SQUIRRELS specification it is claimed that MBS security is fulfilled, which the above disproves. While their claim is based on the similarity to FALCON, the MBS security of FALCON still holds. The subtle differences between SQUIRRELS and FALCON are thus important, when it comes to BUFF security.

wNR. Given $\mathtt{pk} = v$, and a signature $(\mathtt{salt}, s)$ which verifies an unknown message $\mathtt{msg}$, we can find a new public key $\overline{\mathtt{pk}} = \overline{v}$ and a new signature $(\mathtt{salt}, \overline{s})$ that verifies $\mathtt{msg}$ as follows. Let $s' = \mathtt{Decompress}(s)$. We can assume that with large probability, s'_n is divisible by a prime ϖ which is not in P_Δ. E.g., if s'_n is close to uniform, it will be even with about 0.5 probability. In this case, we set $\overline{s}'_n \coloneqq \varpi^{-1} s'_n$. Further, we set $\overline{v}_p \coloneqq \varpi^{-1} v_p$ for each p and let $\overline{v} \in \mathbb{Z}^{n-1}$ be the corresponding vector over $\mathbb{Z}$. Choosing $\overline{s}'_i = s'_i$ for $i = 1, \ldots, n-1$, and $\overline{\mathtt{salt}} = \mathtt{salt}$ yields a new public key $\overline{v}$ and signature $(\overline{\mathtt{salt}}, \overline{s})$, with $\overline{s} = \mathtt{Compress}(\overline{s}')$ that verifies the unknown message. Indeed, the hash h did not change by the procedure and for each $p \in P_\Delta$, we have $\sum_{i=1}^{n-1} \overline{v}_{i,p} c_{i,p} = \varpi^{-1} \sum_{i=1}^{n-1} v_{i,p} c_{i,p} = \varpi^{-1} c_{n,p} = \overline{s}'_n = c_{n,p}$ using that $h_n = 0$. Thereby the verification succeeds.

Remark 10. Modifying SQUIRRELS to incorporate the PS-3 transform (i.e., replacing $h \leftarrow \mathtt{H}(\mathtt{msg}||\mathtt{salt})$ by $h \leftarrow \mathtt{H}(\mathtt{msg}||\mathtt{salt},\mathtt{pk})$) does not suffice to achieve full BUFF security. This is the case, as we can still find S-CEO/S-DEO attacks that are successful with probability greater than $\frac{1}{2^{31}}$: As above we can reduce to the case of a single $p \in P_\Delta$. While the above change to the scheme prevents an attacker to choose v in the kernel of $\langle c', \cdot \rangle - c_n : \mathbb{F}_p^{n-1} \to \mathbb{F}_p$, the probability that this holds for a random v is equal to $\frac{1}{p} \geq \frac{1}{2^{31}}$ (finding v by randomly hitting an element from a subset of size p^{n-2} contained in a set of size p^{n-1}).

5.4 Further Lattice Candidates

The remaining NIST candidates based on lattices are HAETAE and RACCOON. Both use the Fiat-Shamir with aborts framework and are based on the module versions of the learning with errors and short integer solution problems. Both schemes are similar to DILITHIUM and their BUFF analyses are analogous to the DILITHIUM analysis in [14]. In short, HAETAE signs the hash of public key and message and appends a hash value generated (among other inputs) from public key and message to the signature. Thus, HAETAE can be considered to use the BUFF transform, and if we assume the used hash function to be collision-resistant and ϕ-non-malleable (as defined in [14]), we obtain BUFF security by [14, Theorem 5.5]. This is also true for RACCOON, which is structurally very similar to DILITHIUM and hence can be viewed to implement the BUFF transform.

6 Multivariate Schemes

In this section we analyze the signatures that belong to the family of multivariate (MV) schemes. After introducing the foundations and basic properties, we will give a short generic BUFF analysis, i.e., present results that hold for (nearly) all MV schemes under consideration. After this, we turn to the scheme-specific analyses: UOV, which is the basis of all remaining candidates, is treated in Sect. 6.1. This is followed by the analysis of MAYO in Sect. 6.2. While MAYO is based on UOV, its polynomials are constructed in a way that makes the analysis more involved. We present the details to show that despite the complex structure of the public key, MAYO dose not achieve full BUFF security. Both UOV and MAYO—and all MV schemes considered in this paper, except PROV—fulfill MBS as the only BUFF notion. The analysis of PROV, which achieves full BUFF security, follows in Sect. 6.3. For the remaining schemes QR-UOV, SNOVA, TUOV, and VOX, the BUFF analyses are similar to the one given for UOV. We provide a short outline for each scheme in Sect. 6.4.

Background and Notation of MV Schemes. The main object in multivariate cryptography is a multivariate quadratic map $\mathcal{P}\colon \mathbb{F}_q^n \to \mathbb{F}_q^m$, which consists of m homogeneous quadratic polynomials $(p^{(1)}(x), \ldots, p^{(m)}(x))$ in n variables $x = (x_1, \ldots, x_n)$. The coefficients of each of these quadratic polynomials $p^{(k)}(x)$

can be stored in a matrix $P^{(k)}$, where the (i,j)-th entry $(p^{(k)})_{i,j}$ represents the coefficient of the monomial $x_i x_j$. Thus, $p^{(k)}(x)$ can be evaluated as $x^\top P^{(k)} x$.

The task of finding a preimage $s \in \mathbb{F}_q^n$ for a given target vector $t \in \mathbb{F}_q^m$ under a given multivariate quadratic map $\mathcal{P}$ is hard in general, as it amounts to solving a system of multivariate quadratic equations, known as the $\mathcal{MQ}$-Problem. Consequently, a trapdoor needs to be included in the map $\mathcal{P}$, that allows to find such $s \in \mathbb{F}_q^m$ with $\mathcal{P}(s) = t$, which constitutes the signature $\mathtt{sig}$. The precise realization of this trapdoor varies from scheme to scheme.

Generic BUFF Analysis of MV Schemes. In the following, we provide the parts of the BUFF analysis that are the same for (nearly) all multivariate schemes under consideration—namely the MBS proof and wNR attack. The arguments for these two notions will hence not be repeated in the scheme-specific sections. Furthermore, we provide a generic result on the BUFF security of the considered MV schemes using the PS-3 transform.

MBS Security for MV Schemes. Since the target vector $t \in \mathbb{F}_q^n$ is computed as the hash of (at least) the message $\mathtt{msg}$, multivariate schemes naturally satisfy MBS. It is not possible that a single signature $\mathtt{sig} = s$ verifies different messages $\mathtt{msg} \neq \overline{\mathtt{msg}}$, because $\mathtt{H}(\mathtt{msg}||\cdot) = \mathcal{P}(s) = \mathtt{H}(\overline{\mathtt{msg}}||\cdot)$ would imply a collision of $\mathtt{H}$.

wNR Attack Against MV Schemes. For an wNR attack, one is given a public key $\mathtt{pk}$, from which we derive the public map $\mathcal{P}$, and a signature $\mathtt{sig} = s$ to an unknown message $\mathtt{msg}$, and has to find $\overline{\mathtt{pk}} \neq \mathtt{pk}$ and $\overline{\mathtt{sig}} = \overline{s}$ such that $\mathcal{P}(s) = \mathtt{H}(\mathtt{msg}||\cdot) = \overline{\mathcal{P}}(\overline{s})$. Firstly, note that $\mathcal{P}(s) = t = \mathtt{H}(\mathtt{msg}||\cdot)$ can be computed without knowing $\mathtt{msg}$, as s is a valid signature. Next, we generate a key pair $(\overline{\mathtt{sk}}, \overline{\mathtt{pk}})$ with $\overline{\mathtt{pk}} \neq \mathtt{pk}$ and use it to sign the target vector t. This results in a signature $\overline{s}$ that fulfills $\overline{\mathcal{P}}(\overline{s}) = t = \mathtt{H}(\mathtt{msg}||\cdot) = \mathcal{P}(s)$.

Note that this attack is not applicable for Prov, as it hashes the whole public key alongside the message, which prevents us from being able to compute the target *before* choosing the second public key $\overline{\mathtt{pk}}$. We give a proof for wNR security of Prov in Sect. 6.3. For all other schemes under consideration the above attack works, however, for Vox and Snova some extra care is necessary, as both schemes hash parts of the public key alongside the message. In Vox the public key consists of a seed SeedPub and the quadratic map Pub, which is generated using SeedPub. By modifying the seed for the secret key while keeping SeedPub the same, we get a new quadratic map $\overline{\mathrm{Pub}} \neq \mathrm{Pub}$. The new secret key is known to the adversary and can be used to sign to the same target. In Snova the public key is of the form $(s_{\mathrm{public}}, (P_i^{22})_{i\in[m]})$. Here, s_{public} is a seed used to generate the remaining components of the public map $\mathcal{P}$, which is done in the signing and verification algorithm. Choosing $\overline{s}_{\mathrm{public}} = s_{\mathrm{public}}$ and $\overline{\mathtt{sk}} \neq \mathtt{sk}$ guarantees $(\overline{P}_i^{22})_{i\in[m]} \neq (P_i^{22})_{i\in[m]}$ and yields a key pair $(\overline{\mathtt{sk}}, \overline{\mathtt{pk}})$ for which we can apply the above attack.

BUFF Security Using PS-3 *Transform.* Our analysis reveals that from the family of multivariate schemes only Prov satisfies full BUFF security. The main design feature that contributes to this is the hashing of the public key alongside the

message. As all multivariate schemes considered in this paper verify signatures by comparing $\mathtt{H}(\mathtt{msg}, \cdot)$ to $\mathcal{P}(s)$, we can achieve BUFF security for all of them, by adding the *complete*[7] public key alongside the message into the hash function. To prove this, the same arguments as for Prov apply—note that in the analysis of Prov, we use little scheme-specific details except for the size of the domain of $\mathcal{P}$. This approach is very similar to applying the PS-3 transform, except for the fact that an application of PS-3 would result in an additional hash computation (see Fig. 2) that can be avoided by modifying the existing computation of $\mathtt{H}(\mathtt{msg}||\cdot)$ instead. In the following we write PS-3, but it is understood that the simpler modification described above is applied if possible.

Proposition 11. *For* $\Sigma \in \{\text{Mayo}, \text{Qr-Uov}, \text{Snova}, \text{Tuov}, \text{Uov}, \text{Vox}\}$ *and* $\mathtt{H}$ *a random oracle, the transformed scheme* PS-3$[\mathtt{H}, \Sigma]$ *fulfills BUFF security.*

6.1 UOV

The unbalanced oil and vinegar (Uov) signature scheme is the oldest candidate and the foundation of the remaining multivariate schemes, [26,30]. The trapdoor information in Uov is a secret linear m-dimensional subspace, the so-called oil space O, which is annihilated by the public key map $\mathcal{P}$, i.e., $\mathcal{P}(o) = 0$ for all $o \in O$. The dimension of the oil space m needs to equal the number of quadratic equations and the number of variables n usually satisfies $n \approx 2.5m$. We introduce the algorithms of classic Uov here, instead of the compressed versions. The analysis holds for all variants similarly, as we argue below.

Key Pair. The public key $\mathtt{pk} = \{P_i\}_{i\in[m]}$ consists of m matrices

$$P_i = \begin{pmatrix} P_i^{(1)} & P_i^{(2)} \\ 0 & P_i^{(3)} \end{pmatrix},$$

where $P_i^{(1)} \in \mathbb{F}_q^{v\times v}, P_i^{(2)} \in \mathbb{F}_q^{v\times m}$ and $P_i^{(3)} \in \mathbb{F}_q^{m\times m}$. Here, the matrices $P_i^{(1)}$ and $P_i^{(2)}$ are generated randomly from a seed and $P_i^{(3)}$ is computed via

$$P_i^{(3)} = -O^\top P_i^{(1)} O - O^\top P_i^{(2)},$$

with a randomly generated oil space $O \in \mathbb{F}_q^{v\times m}$.

The secret key $\mathtt{sk} = (\mathtt{seed}_{\mathtt{sk}}, O, \{P_i^{(1)}, S_i\}_{i\in[m]})$ consists of a seed $\mathtt{seed}_{\mathtt{sk}}$, the oil space O, a part of the public key matrices $\{P_i^{(1)}\}_{i\in[m]}$, and some auxiliary matrices $\{S_i\}_{i\in[m]}$ needed for signing, given by $S_i = (P_i^{(1)} + P_i^{(1)\top})O + P_i^{(2)}$.

Signature. The signature is given by $\mathtt{sig} = (s, \mathtt{salt})$, containing a vector $s \in \mathbb{F}_q^n$ and a random $\mathtt{salt}$.

Verify. Given a public key $\mathtt{pk} = (P_i^{(1)}, P_i^{(2)}, P_i^{(3)})$, a message $\mathtt{msg}$, and a signature $\mathtt{sig} = (s, \mathtt{salt})$, the verification algorithm is shown in Fig. 10.

[7] Vox and Snova hash parts of the public key, which is insufficient for BUFF security.

$\mathtt{Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig})$

$\mathcal{P} \leftarrow \mathsf{Map}(\mathtt{pk})$
$t \leftarrow \mathsf{Target}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig})$
if $\mathcal{P}(\mathtt{sig}) = t$
  return 1
return 0

$\mathsf{Target}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig})$

$(\cdot, \mathtt{salt}) \leftarrow \mathtt{sig}$
$t \leftarrow \mathtt{H}(\mathtt{msg}||\mathtt{salt})$ // UOV
$t \leftarrow \mathtt{H}(\mathtt{H}(\mathtt{msg})||\mathtt{salt})$ // MAYO
$t \leftarrow \mathtt{H}_2(\mathtt{H}_1(\mathtt{pk})||\mathtt{msg}||\mathtt{salt})$ // PROV
return t

$\mathsf{Map}(\mathtt{pk})$ in UOV

$(P_i^{(1)}, P_i^{(2)}, P_i^{(3)}) \leftarrow \mathtt{pk}$
$P_i \leftarrow \begin{pmatrix} P_i^{(1)} & P_i^{(2)} \\ 0 & P_i^{(3)} \end{pmatrix}$
return $\mathcal{P}$

$\mathsf{Map}(\mathtt{pk})$ in PROV

$(\mathtt{seed}_{\mathtt{pk}}, (P_i^{(3)})) \leftarrow \mathtt{pk}$
$(P_i^{(1)}, P_i^{(2)}) \leftarrow \mathtt{E}(\mathtt{seed}_{\mathtt{pk}})$
$P_i = \begin{pmatrix} P_i^{(1)} & P_i^{(2)} \\ 0 & P_i^{(3)} \end{pmatrix}$
return $\mathcal{P}$

$\mathsf{Map}(\mathtt{pk})$ in MAYO

$(\mathtt{seed}_{\mathtt{pk}}, \{P_i^{(3)}\}) \leftarrow \mathtt{pk}$
$\{P_i^{(1)}, P_i^{(2)}\} \leftarrow \mathtt{E}(\mathtt{seed}_{\mathtt{pk}})$
return $\mathcal{P}^*$

Fig. 10. Verification algorithm of UOV, MAYO, and PROV. Recall that the public map $\mathcal{P}$ consists of m homogeneous quadratic polynomials $(p^{(1)}(x), \ldots, p^{(m)}(x))$, and can be computed from $P_1, \ldots, P_k$ using the relation $p^i(x) = x^\top P_i x$. For MAYO the larger map $\mathcal{P}^*$ is used, which can be computed from $\mathcal{P}$ as described in Eq. (2). Lastly, note that $\mathtt{E}(\cdot)$ is used as an abbreviation for $\mathtt{Expand}(\cdot)$.

S-CEO. Given a public key $\mathtt{pk}$, a message $\mathtt{msg}$, and a signature $\mathtt{sig} = (s, \mathtt{salt})$ such that $\mathtt{Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig}) = 1$, we need to find a second public key $\overline{\mathtt{pk}} \neq \mathtt{pk}$ such that for the corresponding public key map $\overline{\mathcal{P}}$ it holds that $\overline{\mathcal{P}}(s) = \mathtt{H}(\mathtt{msg}||\mathtt{salt})$. Let $p_{i,j}^{(k)}$ be the (i,j)-th entry of the public key matrix P_k coming from $\mathtt{pk}$. We define $\overline{p}_{i,j}^{(k)}$, the coefficients of $\overline{P}_k$ from $\overline{\mathtt{pk}}$ as $p_{i,j}^{(k)}$, except for the following adjustment. We pick an arbitrary $i \in [v+1, n-1]$ and change $\overline{p}_{i,i}^{(k)}$ and $\overline{p}_{i+1,i+1}^{(k)}$ s.t. $p_{i,i}^{(k)} s_i^2 + p_{i+1,i+1}^{(k)} s_{i+1}^2 = \overline{p}_{i,i}^{(k)} s_i^2 + \overline{p}_{i+1,i+1}^{(k)} s_{i+1}^2$. Keeping all other coefficients, we get $\overline{P}_k(s) = P_k(s)$ for all k, hence verification succeeds for $\overline{\mathtt{pk}}$.

S-DEO. Given a public key $\mathtt{pk}$, a message $\mathtt{msg}$, and a signature $\mathtt{sig} = (s, \mathtt{salt})$ such that $\mathtt{Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig}) = 1$, we need to find a second public key $\overline{\mathtt{pk}} \neq \mathtt{pk}$ and a second message $\overline{\mathtt{msg}} \neq \mathtt{msg}$ such that $\overline{\mathcal{P}}(s) = \overline{h} = (\overline{h}_k) = \mathtt{H}(\overline{\mathtt{msg}}||\mathtt{salt})$. We take some index $l \in [v+1, n]$, with $s_l \neq 0$. For each $k \in [m]$, set $\overline{p}_{i,i}^k = (\overline{h}_k - \sum_{i<j,(i,j)\neq(l,l)} p_{i,j}^k s_i s_j)/(s_l^2)$. Then we found $\overline{\mathcal{P}}$ with $\overline{\mathcal{P}}(s) = \overline{h}$.

Variants. The statements also hold for the compressed variants pkc and pkc+skc. For these, the public key does not consist of the matrices $\{P_i\}_{i\in[m]}$, but only of the submatrices $\{P_i^{(3)}\}_{i\in[m]}$ and a seed that is used to generate $\{P_i^{(1)}\}_{i\in[m]}$ and $\{P_i^{(2)}\}_{i\in[m]}$. The results of our analysis only require a change of the $P_i^{(3)}$, so that the attacks work for the compressed versions as well.

6.2 MAYO

In MAYO, the public key map $\mathcal{P}$ has the same structure as in UOV, but it is publicly whipped up to a k-fold larger map $\mathcal{P}^* : \mathbb{F}_q^{kn} \to \mathbb{F}_q^m$ via

$$\mathcal{P}^*(s_1, \ldots, s_k) = \sum_{i=1}^{k} E_{ii}\mathcal{P}(s_i) + \sum_{i=1}^{k}\sum_{j=i+1}^{k} E_{ij}\mathcal{P}'(s_i, s_j), \tag{2}$$

where $E_{ij} \in \mathbb{F}_q^{m\times m}$ are system parameters and $\mathcal{P}'$ is the bilinear map associated to $\mathcal{P}$, i.e., component-wise $P'_l(s_i, s_j) = s_i^\top(P_l + P_l^\top)s_j$, for each l. The benefit of this approach is a smaller public key size at the expense of a slightly larger signature and an additional security assumption: the *Multi-Target Whipped MQ* problem [6, Section 5.1].

Key Pair. The secret key is given by a private seed $\mathtt{sk} = \mathtt{seed}_{\mathtt{sk}}$. It is used to derive a public seed $\mathtt{seed}_{\mathtt{pk}}$ and the secret linear oil space $O \in \mathbb{F}_q^{(n-o)\times o}$. The public key is given by $\mathtt{pk} = (\mathtt{seed}_{\mathtt{pk}}, \{P_i^{(3)}\}_{i\in[m]})$, where

$$P_i^{(3)} = -O^\top P_i^{(1)} O - O^\top P_i^{(2)} \in \mathbb{F}_q^{o\times o}.$$

Hereby, $P_i^{(1)} \in \mathbb{F}_q^{(n-o)\times(n-o)}$ and $P_i^{(2)} \in \mathbb{F}_q^{(n-o)\times o}$ are expanded from $\mathtt{seed}_{\mathtt{pk}}$.

Signature. The signature is given by $\mathtt{sig} = (s_1, \ldots, s_k, \mathtt{salt})$, with $s_i \in \mathbb{F}_q^n$.

Verify. Given a public key $\mathtt{pk} = (\mathtt{seed}_{\mathtt{pk}}, \{P_i^{(3)}\}_{i\in[m]})$, a message $\mathtt{msg}$, and a signature $\mathtt{sig} = (s_1, \ldots, s_k, \mathtt{salt})$, the verification is shown in Fig. 10.

S-CEO. Given a public key $\mathtt{pk} = (\mathtt{seed}_{\mathtt{pk}}, \{P_i^{(3)}\}_{i\in[m]})$, a message $\mathtt{msg}$, and a signature $\mathtt{sig} = (s_1, \ldots, s_k, \mathtt{salt})$, such that $\mathtt{Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig}) = 1$, we need to find a second public key $\overline{\mathtt{pk}} \neq \mathtt{pk}$ such that $\overline{\mathcal{P}}^*(s_1, \ldots, s_k) = t = \mathtt{H}(\mathtt{H}(\mathtt{msg})||\mathtt{salt})$ holds, where $\overline{\mathcal{P}}^*$ is derived from $\overline{\mathtt{pk}}$. The main observation to tackle this task, is that the map $\mathcal{P}^*$ is linear with respect to the entries of its corresponding public key matrices P_i.

The strategy is now to generate various $\tilde{\mathtt{pk}}_a$, where we always use the same $\mathtt{seed}_{\mathtt{pk}}$, but randomly generated $(\{P_{i,a}^{(3)}\}_{i\in[m]})_a$ for $a \in \{1, 2, \ldots\}$. Denote by $\tilde{\mathcal{P}}_a$ the quadratic map associated to this public key. Then, we consecutively compute $\tilde{\mathcal{P}}_a^*(s_1, \ldots, s_k) = x_a$ until we gathered m linearly independent vectors x_a. Thus, we find $\lambda_a \in \mathbb{F}_q$, such that $t = \sum_{a=1}^m \lambda_a \cdot x_a$. Now we add up the randomly generated matrices accordingly and define $\overline{pk} = (\mathtt{seed}_{pk}, \{\overline{P}_i^{(3)}\}_{i\in[m]})$, where $\overline{P}_i^{(3)} := \sum_{a=1}^m \lambda_a (P_i^{(3)})_a$ for all $i \in [m]$. Due to the linearity we have

$$\overline{\mathcal{P}}^*(s_1, \ldots, s_k) = \sum_{a=1}^{m} \lambda_a \tilde{\mathcal{P}}_a^*(s_1, \ldots, s_k) = \sum_{a=1}^{m} \lambda_a x_a = t.$$

Thus, an attacker is able to find a different public key $\overline{\mathtt{pk}} \neq \mathtt{pk}$, such that $\mathtt{Verify}(\overline{\mathtt{pk}}, \mathtt{msg}, \mathtt{sig}) = 1$ and MAYO is not S-CEO-secure.

S-DEO. Given a public key $\mathtt{pk} = (\mathtt{seed}_{\mathtt{pk}}, \{P_i^{(3)}\}_{i\in[m]})$, a message $\mathtt{msg}$, and a signature $\mathtt{sig} = (s_1, \ldots, s_k, \mathtt{salt})$ such that $\mathtt{Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig}) = 1$, we need to find a second public key $\overline{\mathtt{pk}} \neq \mathtt{pk}$ and message $\overline{\mathtt{msg}} \neq \mathtt{msg}$ such that $\overline{\mathcal{P}}^*(s_1, \ldots, s_k) = t = \mathtt{H}(\mathtt{H}(\overline{\mathtt{msg}})||\mathtt{salt})$. Since the vectors x_a we generated in the S-CEO analysis give a basis for the complete vector space $\mathbb{F}_q^m$, an attacker can compute $\bar{t} = \mathtt{H}(\mathtt{H}(\overline{\mathtt{msg}})||\mathtt{salt})$ and find $\overline{\lambda}_a$ such that $\bar{t} = \sum \overline{\lambda}_a x_a$ for some randomly chosen message $\overline{\mathtt{msg}} \neq \mathtt{msg}$. Thus, the same attack that was developed to analyze S-CEO, works here and MAYO is not S-DEO-secure.

6.3 PROV

Key Pair. Let $\mathbb{F}$ denote the finite field $\mathbb{F}_{2^8}$ and $\delta := o - m$. The public key $\mathtt{pk}$ is a pair $(\mathtt{seed}_{\mathtt{pk}}, (P_i^{(3)})_{i\in[m]})$ where $P_i^{(3)} \in \mathbb{F}^{(m+\delta)\times(m+\delta)}$ for all i. From $\mathtt{seed}_{\mathtt{pk}}$ the matrices $(P_i^{(1)}, P_i^{(2)})_{i\in[m]}$ with $P_i^{(1)} \in \mathbb{F}^{(n-m-\delta)\times(n-m-\delta)}$ and $P_i^{(2)} \in \mathbb{F}^{(n-m-\delta)\times(m+\delta)}$ for all i, are generated. We denote by P_i the matrix

$$\begin{pmatrix} P_i^{(1)} & P_i^{(2)} \\ 0 & P_i^{(3)} \end{pmatrix}.$$

The secret key is the triple $(\mathtt{seed}_{\mathtt{pk}}, \mathtt{seed}_{\mathtt{sk}}, \mathtt{H}(\mathtt{pk}))$. From $\mathtt{seed}_{\mathtt{sk}}$ the matrix $O \in \mathbb{F}^{(n-m-\delta)\times(m+\delta)}$ is generated.

Signature. A signature is given by $\mathtt{sig} = (\mathtt{salt}, s)$ for $s \in \mathbb{F}^n$.

Verify. Given a public key $\mathtt{pk} = (\mathtt{seed}_{\mathtt{pk}}, (P_i^{(3)})_{i\in[m]})$, a message $\mathtt{msg}$, and a signature $\mathtt{sig} = (\mathtt{salt}, s)$, the verification is shown in Fig. 10.

S-CEO. Given a public key $\mathtt{pk}$, a message $\mathtt{msg}$, and a signature $\mathtt{sig} = (\mathtt{salt}, s)$ such that $\mathtt{Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig}) = 1$, we need to find a different public key $\overline{\mathtt{pk}} = (\overline{\mathtt{seed}}_{\mathtt{pk}}, (\overline{P}_i^3)_{i\in[m]})$ such that $(\bar{t}_i)_{i\in[m]} = \bar{h}$ and hence $(s^\top \overline{P}_i s)_{i\in[m]} = \mathtt{H}_2(\mathtt{H}_1(\overline{\mathtt{pk}})||\mathtt{msg}||\mathtt{salt})$. As both sides of the latter equation depend on $\overline{\mathtt{pk}}$ and the value on the right is random (assuming $\mathtt{H}_1$ and $\mathtt{H}_2$ to be random oracles), the probability to find a suitable $\overline{\mathtt{pk}}$ is $\frac{1}{|\mathbb{F}^m|} = \frac{1}{2^{8m}} \leq 2^{-368}$, for all proposed variants.

S-DEO. Given a public key $\mathtt{pk}$, a message $\mathtt{msg}$, and a signature $\mathtt{sig} = (\mathtt{salt}, s)$ such that $\mathtt{Verify}(\mathtt{pk}, \mathtt{msg}, \mathtt{sig}) = 1$, we need to find a second public key $\overline{\mathtt{pk}} \neq \mathtt{pk}$ and a second message $\overline{\mathtt{msg}} \neq \mathtt{msg}$ such that $\mathtt{Verify}(\overline{\mathtt{pk}}, \overline{\mathtt{msg}}, \mathtt{sig}) = 1$. This is not feasible by the same argument that was used for S-CEO security, as changing the message only influences the hash value $\bar{h} = \mathtt{H}_2(\mathtt{H}_1(\overline{\mathtt{pk}})||\mathtt{msg}||\mathtt{salt})$.

wNR. Given a public key $\mathtt{pk}$ and a signature $\mathtt{sig}$ to an unknown message $\mathtt{msg}$, one has to find another public key $\overline{\mathtt{pk}} \neq \mathtt{pk}$, and a signature $\overline{\mathtt{sig}}$ such that $\mathtt{Verify}(\overline{\mathtt{pk}}, \mathtt{msg}, \overline{\mathtt{sig}}) = 1$. This is not feasible as one would have to find $\overline{\mathtt{pk}}$ such that $(\bar{t}_i)_{i\in[m]} = \bar{h}$, where $\bar{h} = \mathtt{H}_2(\mathtt{H}_1(\overline{\mathtt{pk}})||\mathtt{msg}||\overline{\mathtt{salt}})$ is unknown as $\mathtt{msg}$ is. Note that we can compute $h = (s^\top P_i s)_i$ but not $\mathtt{msg}$ and hence not $\bar{h}$. Thus, the probability for the equality $(\bar{t}_i)_{i\in[m]} = \bar{h}$ to hold is $\frac{1}{2^{8m}}$ and therefore less than 2^{-368} for all variants.

6.4 Further Multivariate Candidates

The remaining NIST signature candidates based on multivariate polynomials are QR-UOV, SNOVA, TUOV and VOX. For all of them, the BUFF analysis follows the same idea as the one given for UOV in Sect. 6.1. We provide a short overview over the main arguments in the following.

The main difference between QR-UOV and UOV is that the public key matrices $P_1^{(i)}, P_2^{(i)}$, and $P_3^{(i)}$ of QR-UOV are block matrices, where each component $\Phi_g^f \in \mathbb{F}_q^{l\times l}$ corresponds to an element g of the quotient ring $\mathbb{F}_q[x]/(f)$, with an irreducible polynomial $f \in \mathbb{F}_q[x]$ of degree l. The polynomial matrices of the subalgebra $\mathcal{A}_f := \{\Phi_g^f \in \mathbb{F}_q^{l\times l} \mid g \in \mathbb{F}_q[x]/(f)\}$ are defined entry-wise such that $(\Phi_g^f)_{ij}$ is the coefficient of x^{i-1} in $x^{j-1} \cdot g$. In the S-CEO/S-DEO analysis for QR-UOV we cannot modify single entries $p_{i,j}^{(k)}$ of the matrices $P_k^{(3)}$ that were used to control the values $y_k = s^\top P_k s$ in the analysis of UOV. Instead, we can only alter one coefficient (or more) of the polynomials $g = \sum_{i=0}^{l-1} a_i x^i \in \mathcal{A}_f$ that are stored in the $P_k^{(3)}$ part of the public key $\mathtt{pk}$. This will change l values in the corresponding block $\Phi_g^f \in \mathbb{F}_q^{l\times l}$ of $P_i^{(3)}$. However, we can still dictate the result $r_k = s_l^\top \Phi_g^f s_l$ by choosing the coefficients of g accordingly. Here s_l denote the l entries of the vector $s \in \mathbb{F}^n$ that are multiplied with this block.

SNOVA differs from UOV in the fact that it works over the non-commutative ring $R = \mathbb{F}_q^{l\times l}$ instead of $\mathbb{F}_q$. Further, SNOVA computes the target vector as $t = \mathtt{H}(\mathrm{seed}_p||\mathtt{H}(\mathtt{msg})||\mathtt{salt})$ for $\mathtt{pk} = (\mathrm{seed}_p, \{P_i^{22}\}_{i\in[m]})$ with $P_i^{22} \in R^{o\times o}$, while for UOV we have $t = \mathtt{H}(\mathtt{msg}||\mathtt{salt})$. However, for neither of the BUFF security notions, the adversary has to provide honestly generated keys, hence it can choose two different public keys $\overline{\mathtt{pk}} \neq \mathtt{pk}$ that have the same seed, which then result in the same target t. Then, S-CEO and S-DEO insecurity follows by using the concrete parameters provided in SNOVA to prove systems of equations solvable.

The TUOV analysis is completely analogous to the UOV analysis, as the additional affine transformation $S\colon \mathbb{F}_q^m \to \mathbb{F}_q^m$ has no impact on the analysis.

VOX is a UOV-based scheme that incorporates the quotient ring technique. Despite their claim to achieve BUFF security, VOX only satisfies MBS. S-CEO and S-DEO can be attacked as in UOV. In short the attack proceeds as follows: One keeps the part SeedPub of the public key $\mathtt{pk} = (\mathrm{SeedPub}, \mathrm{Pub})$ unchanged. The Pub part can be changed independently and is chosen as in the attack against UOV. Note that VOX uses the quotient ring technique, however, the problem of defining Pub is still the same as in UOV, just over the extension field. The wNR security can be attacked as described at the beginning of this section.

7 Conclusion

In this work, we analyzed the signature schemes based on codes, isogenies, lattices, and multivariate polynomials submitted to the additional NIST PQC standardization effort for signatures regarding their BUFF security. Besides the analysis of the original schemes, we included comments on the BUFF security after

a light transform, the so-called PS-3 transform. In fact, we see that often, the PS-3 transform suffices to ensure BUFF security, despite the fact that the PS-3 transform is not sufficient for generic schemes. This gap between the general statement and the empirical evidence on practical schemes can be analyzed further.

In the NIST competition, there are even more signature schemes, which we have not analyzed in this work. An interesting future work is to analyze those. In particular, this would give a chance to assess the empirical evidence regarding the relation of BUFF security and the PS-3 transform.

We considered a weaker form of non-resignability (wNR) as the initial definition turned out to be unachievable—the problem being the auxiliary information. The majority of our results regarding wNR—attacks against 12 out of 17 signature schemes—remain relevant regardless of how the auxiliary information issue gets resolved eventually. The reason is that neither attack relies on any auxiliary information. On the other hand, our positive results only guarantee security against non-resignability in this restricted form. Once the matter of defining non-resignability is completely resolved, our positive results given here should be re-evaluated. Note, however, that for the 5 positive results, the schemes implicitly use either the PS-3 or the BUFF transform. Hence, if either the PS-3 or BUFF transform can be shown to generally satisfy a new definition of NR, the results would apply to the 5 positive results presented here.

Acknowledgements. This work has been funded by the Deutsche Forschungsgemeinschaft (DFG – German Research Foundation) – 505500359 and SFB 1119 – 236615297, by the German Federal Ministry of Education and Research (BMBF) under the projects 6G-RIC (16KISK033) and Quant-ID (16KISQ111), and by the Hector Foundation II.

References

1. Ayer, A.: Duplicate signature key selection attack in let's encrypt (2015). https://www.agwa.name/blog/post/duplicate_signature_key_selection_attack_in_lets_encrypt
2. Baldi, M., et al.: LESS. Technical report, National Institute of Standards and Technology (2023)
3. Baldi, M., et al.: CROSS. Technical report, National Institute of Standards and Technology (2023)
4. Banegas, G., et al.: Wave. Technical report, National Institute of Standards and Technology (2023)
5. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: Denning, D.E., Pyle, R., Ganesan, R., Sandhu, R.S., Ashby, V. (eds.) ACM CCS 1993, pp. 62–73. ACM Press (1993)
6. Beullens, W., Campos, F., Celi, S., Hess, B., Kannwischer, M.J.: MAYO. Technical report, National Institute of Standards and Technology (2023)
7. Beullens, W., et al.: UOV. Technical report, National Institute of Standards and Technology (2023)

8. Blake-Wilson, S., Menezes, A.: Unknown key-share attacks on the station-to-station (STS) protocol. In: Imai, H., Zheng, Y. (eds.) PKC 1999. LNCS, vol. 1560, pp. 154–170. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-49162-7_12
9. Bos, J., et al.: HAWK. Technical report, National Institute of Standards and Technology (2023)
10. Chavez-Saab, J., et al.: SQIsign. Technical report, National Institute of Standards and Technology (2023)
11. Cheon, J.H., et al.: HAETAE. Technical report, National Institute of Standards and Technology (2023)
12. Chou, T., et al.: MEDS. Technical report, National Institute of Standards and Technology (2023)
13. Corte-Real Santos, M., Eriksen, J.K., Meyer, M., Reijnders, K.: AprèsSQI: Extra Fast Verification for SQIsign Using Extension-Field Signing. Cryptology ePrint Archive, Paper 2023/1559 (2023)
14. Cremers, C., Düzlü, S., Fiedler, R., Fischlin, M., Janson, C.: BUFFing signature schemes beyond unforgeability and the case of post-quantum signatures. In: 2021 IEEE Symposium on Security and Privacy, pp. 1696–1714. IEEE Computer Society Press (2021)
15. del Pino, R., et al.: Raccoon. Technical report, National Institute of Standards and Technology (2023)
16. Ding, J., et al.: TUOV. Technical report, National Institute of Standards and Technology (2023)
17. Don, J., Fehr, S., Huang, Y.-H., Struck, P.: On the (in)security of the BUFF transform. IACR Cryptology ePrint Archive 2023:1634 (2023)
18. Ducas, L., Postlethwaite, E.W., Pulles, L.N., van Woerden, W.: Hawk: module LIP makes lattice signatures fast, compact and simple. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022. LNCS, vol. 13794, pp. 65–94. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-22972-5_3
19. Espitau, T., Niot, G., Sun, C., Tibouchi, M.: SQUIRRELS. Technical report, National Institute of Standards and Technology (2023)
20. Furue, H., et al.: QR-UOV. Technical report, National Institute of Standards and Technology (2023)
21. Goldwasser, S., Micali, S., Rivest, R.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. (1988)
22. Goubin, L., et al.: PROV. Technical report, National Institute of Standards and Technology (2023)
23. Hülsing, A., et al.: SPHINCS$^+$. Technical report, National Institute of Standards and Technology (2020)
24. Jackson, D., Cremers, C., Cohn-Gordon, K., Sasse, R.: Seems legit: automated analysis of subtle attacks on protocols that use signatures. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019, pp. 2165–2180. ACM Press (2019)
25. Kim, T.H.-J., Basescu, C., Jia, L., Lee, S.B., Hu, Y.-C., Perrig, A.: Lightweight source authentication and path validation. In: Proceedings of the 2014 ACM Conference on SIGCOMM (2015)
26. Kipnis, A., Patarin, J., Goubin, L.: Unbalanced oil and vinegar signature schemes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 206–222. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_15
27. Lyubashevsky, V., et al.: CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology (2020)

28. Menezes, A., Smart, N.: Security of signature schemes in a multi-user setting. Des. Codes Cryptography **33**, 261–274 (2004). https://doi.org/10.1023/B:DESI.0000036250.18062.3f
29. National Institute of Standards and Technology. Call for additional digital signature schemes for the post-quantum cryptography standardization process (2022). https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/call-for-proposals-dig-sig-sept-2022.pdf
30. Patarin, J.: The oil and vinegar signature scheme (1997)
31. Patarin, J., et al.: VOX. Technical report, National Institute of Standards and Technology (2023)
32. Pornin, T., Stern, J.P.: Digital signatures do not guarantee exclusive ownership. In: Ioannidis, J., Keromytis, A., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 138–150. Springer, Heidelberg (2005). https://doi.org/10.1007/11496137_10
33. Prest, T., et al.: FALCON. Technical report, National Institute of Standards and Technology (2020)
34. Wang, L.-C., et al.: SNOVA. Technical report, National Institute of Standards and Technology (2023)
35. Yu, Y., et al.: HuFu. Technical report, National Institute of Standards and Technology (2023)