Cryptanalysis of Multivariate-Based Post-Quantum Digital Signature Schemes

Laura Maddison

Thesis submitted in partial fulfillment of the requirements for the degree of
Master of Science Mathematics and Statistics[1]

Department of Mathematics and Statistics
Faculty of Science
University of Ottawa

[1]The M.Sc. program is a joint program with Carleton University, administered by the Ottawa-Carleton Institute of Mathematics and Statistics

Abstract

In this thesis, we study three proposed multivariate-based post-quantum digital signature schemes: Triangular Unbalanced Oil and Vinegar (TUOV), Biscuit, and Multivariate Polynomial Public Key Digital Signature Scheme (MPPK/DS). Our work aims to first explore the underlying problems in multivariate cryptography that these schemes employ. We then provide detailed descriptions and concrete security analysis of these three proposed schemes. We prove that a security reduction that was claimed to be a unique feature of the TUOV signature scheme also applies to the more classical UOV scheme, we provide careful analysis and more detailed proofs of two attacks on Biscuit from the literature, and we present a novel classically efficient forgery attack on MPPK/DS that renders it insecure.

Dedications

For my parents. None of this would be possible without your unwavering support.

Acknowledgements

I would first like to thank my supervisor, Dr. Monica Nevins, for her academic and personal support during my studies. Her unflinching optimism and enthusiasm were invaluable as I worked through the more stressful moments of this work. Her mentorship helped me grow and shaped me into the researcher I am today.

I would also like to thank my examiners, Dr. Anne Broadbent and Dr. Daniel Panario, for their thorough review of this thesis and their valuable comments on it. I am grateful as well to Dr. Mateja Šajna for introducing me to the world of mathematical research during my undergraduate studies.

My research was supported by an NSERC Canada Graduate Scholarship (Master's).

I also thank my labmates and other members of the QUASaR research group for their friendship and for sparking my interest in all things cryptography.

I am extraordinarily grateful to my parents, Heather and Ian, and my brother, Kyle, for their unconditional support of all my academic pursuits. I could not have done this without the support of my partner, Graham, whose patience and understanding as I continue my studies is invaluable. Thank you to my friends outside of mathematics for keeping me sane and grounded. Finally, to my dance teachers Peggy and Lisa, as well as all my teammates : you are the most inspiring group of women, thank you for everything.

Contents

Chapter 1

Introduction

Post-quantum cryptography is the growing field concerned with developing public-key cryptosystems that are secure against quantum computers. Quantum computers are "machines that exploit quantum mechanical phenomena to solve mathematical problems that are difficult or intractable for conventional computers" [52]. It has been proven that sufficiently large and reliable quantum computers can efficiently solve many hard classical problems, such as factoring integers and the Discrete Logarithm Problem, so new cryptographic schemes are required [52]. While large-scale quantum computers have yet to be developed, it is believed at this point to be physically possible, so it is important that experts begin working on quantum-secure algorithms as the development and standardization of such algorithms is a time-consuming process. The National Institute of Standards and Technology (NIST) is one of the bodies responsible for the standardization of such algorithms, and they made a public call for additional Digital Signature Schemes, whose submissions were due in June 2023 [48]. Digital Signature Schemes are the algorithms that allow us to securely sign and verify signatures on digital files.

One particular area of cryptography that is promising in the post-quantum context is multivariate cryptography. In the thesis, we study three different proposed post-quantum multivariate-based digital signature schemes, two of which were submitted to the NIST standardization process in 2023, and one of which was not eventually submitted.

The security of such schemes is based on the hardness of solving systems of multivariate equations, which are usually quadratic, over a ring or a finite field [63]. The quadratic case of this problem over a finite field is called the Multivariate Quadratic (MQ) Problem, and is known to be NP-complete [8]. We formally define and further discuss this problem in Section 2.4. The first ideas in the area of multivariate cryptography are attributed to Matsumoto and Imai in 1988 [45] and were formalized in

the Hidden Field Equations (HFE) encryption scheme due to Patarin in 1996 [53]. Patarin later introduced the Oil and Vinegar digital signature scheme, that uses a structured variant of the MQ problem at its core [54]. A slight modification of this scheme, called *Unbalanced* Oil and Vinegar (UOV) is still believed to be secure in the post-quantum context and is used as the basis of many of the new proposals to NIST, for example [13, 22].

Since receiving the submissions for additional digital signatures in June 2023, NIST has published forty post-quantum proposals, twelve of which use some kind of multivariate-based hard problem [48]. The techniques used in these schemes are varied, as some are variants of UOV, some use zero-knowledge proofs, and others use different approaches entirely [48]. These new proposals must be scrutinized by the cryptographic community to ensure that they are secure in the post-quantum context.

We first study Triangular Unbalanced Oil and Vinegar (TUOV), a variant of the more classical UOV digital signature scheme [22]. TUOV was submitted to the NIST standardization process in 2023 and generalizes the polynomials used in UOV slightly in an attempt to boost security. In Section 3.3.3, we prove that a security reduction that was claimed to be unique to TUOV [22, Theorem 1] can also be applied to UOV and discuss why the reduction is not cryptographically relevant to either scheme in practice. In Section 3.3.4, we compare TUOV to the standard UOV submission to NIST [13] and argue that TUOV presents no tangible advantage over its predecessor.

The second scheme we study, which was also submitted to NIST, is the Biscuit digital signature scheme [10]. Biscuit employs a technique using a zero-knowledge proof derived from a secure multiparty protocol, with an underlying hard problem that is a structured variant of the MQ problem, called PowAff2(q, m, n). In Section 4.1, we first provide a comprehensive overview of the general construction of signature schemes derived from secure multiparty protocols. We then study the special structure of the polynomials used in Biscuit in Section 4.2. We provide a more detailed proof of a specialized attack for solving PowAff2(q, m, n) from [10] in Theorems 4.2.3 and 4.2.4. We also outline in Theorem 4.2.6 an attack that was posted to the NIST forum for official comment, which indicated that the parameters used for Biscuit should be increased slightly from the original proposal [15]. Neither attack completely breaks the Biscuit digital signature scheme, but they do suggest that the additional underlying structure in the polynomials could lead to vulnerabilities.

The final scheme that we study is Multivariate Polynomial Public Key Digital Signature (MPPK/DS), which was published in 2022 but was not eventually submitted to NIST [41]. This scheme bases its security on a set of multivariate polynomials over a ring rather than a field. The signing and verification algorithms make use of modular exponentiation, which is a very different technique when compared to most multivariate-based signature schemes. Section 5.2 provides our cryptanalysis

of MPPK/DS, beginning with observations that indicate that the signing algorithm must be slightly tweaked from the original presentation. Proposition 5.2.4 provides our original efficient forgery attack on MPPK/DS and Proposition 5.2.5 combines the ideas from our attack with those of another attack published in [30] to improve upon its efficiency. The contents of Chapter 5 were published in [44] in 2024.

Structured variants of the MQ problem like those used in the schemes we study have historically presented security concerns. For instance, the Rainbow digital signature scheme, a layered variant of UOV, was advanced to the third round of the previous NIST competition [21] before being broken using an instance of a MinRank attack [12]. The question of whether the cryptographic problems underlying TUOV and Biscuit will withstand attempts at specialized attacks, and ultimately be deemed hard in the post-quantum context, remains open.

The outline of the thesis is as follows.

In Chapter 2, we provide the required background material for the thesis, including notation, necessary concepts from cryptography and ring theory, and an overview of the MQ problem.

Chapter 3 is focused on UOV-based digital signature schemes, and in particular, the TUOV digital signature scheme submitted to NIST.

In Chapter 4, we study signature schemes based on Multiparty Computation in-the-Head (MPCitH), and in particular the Biscuit digital signature scheme submitted to NIST.

Chapter 5 provides our novel cryptanalysis of MPPK/DS, a signature scheme that was published in 2022, but not eventually submitted to NIST. The content of this chapter was published in [44] in 2024.

Chapter 2

Background

In this chapter we provide the required background material for the thesis. In Section 2.1, we set some notation that will be used throughout the thesis. In Section 2.2, we define several important cryptographic notions, and in Section 2.3 we state the required mathematical definitions and theorems. Finally, in Section 2.4, we define the Multivariate Quadratic (MQ) problem and discuss algorithms for solving it.

2.1 Notation

For a positive integer m, we let $[m] = \{1, \ldots, m\}$.

We use $\mathbb{Z}_N$ to denote the ring of integers modulo N for any positive integer N. We then define the multiplicative group

$$\mathbb{Z}_N^* = \{a \in \mathbb{Z}_N \mid \gcd(a, N) = 1\}.$$

In the special case that $N = p$ is prime, $\mathbb{Z}_p$ is a field which we denote by $\mathbb{F}_p$. We will use $\mathbb{F}_q$ to denote the finite field with q elements where q is a prime power. When we are discussing a general field (finite or not), we use $\mathbb{F}$.

We use φ to denote the Euler totient function on positive integers, so $\varphi(N) = |\mathbb{Z}_N^*|$.

If R is a ring, then we use $R[x_1, \ldots, x_n]$ to denote the ring of polynomials in n variables over R.

We will use lowercase letters such as x to denote single elements of a set and bold lowercase letters such as $\mathbf{x}$ to denote vectors of elements in a set. We use bold uppercase letters such as $\mathbf{A}$ to denote matrices.

For a positive integer n and a vector space V over $\mathbb{F}^n$, we let

$$V^{\perp} = \{\mathbf{x} \in \mathbb{F}^n \mid \mathbf{x}^{\top}\mathbf{y} = 0 \text{ for all } \mathbf{y} \in V\}.$$

By common overload of notation, we call $V^{\perp}$ the *orthogonal complement* of V in $\mathbb{F}^n$, even though it may happen that $V \cap V^{\perp}$ is nonzero.

We use $x \xleftarrow{\$} X$ to denote that the element x is sampled uniformly at random from a set X.

We use $\{0,1\}^*$ to denote the set of binary strings of arbitrary length.

2.2 Basic Cryptographic Notions

We first provide the required definitions in cryptography. Many of these definitions are from courses in Quantum Computing and Mathematical Cryptography at the University of Ottawa [16, 17]. However, these definitions can also be found in many standard texts such as [31, 59].

We will require the notion of computational complexity to discuss the efficiency of algorithms in the thesis. We use *"Big O" notation* to set bounds on the behaviour of a function.

Definition 2.2.1. *[17, 37] Let $f(n), g(n)$ be functions from positive integers to real numbers. Then $f(n) \in \mathcal{O}(g(n))$ if $\exists c \in \mathbb{R}$ positive and $n_0 \in \mathbb{Z}$ such that $\forall n \geq n_0$, $f(n) \leq cg(n)$. Similarly, $f(n) \in \Omega(g(n))$ if $\exists c \in \mathbb{R}$ and $n_0 \in \mathbb{Z}$ such that $\forall n \geq n_0$, $f(n) \geq cg(n)$.*

Note that $\Omega(g(n))$ and $\mathcal{O}(g(n))$ are not disjoint since, for example, $\frac{1}{2}n^2 \in \Omega(n^2)$ and $\frac{1}{2}n^2 \in \mathcal{O}(n^2)$.

"Big O" notation will allow us to define our notion of efficiency for algorithms.

Definition 2.2.2. *[17] An algorithm with input of length n is* **efficient***, or runs in* **polynomial time***, if it runs in time $\mathcal{O}(n^k)$ for some positive integer k.*

The algorithms used in cryptographic schemes will generally be *PPT algorithms*, which we define below.

Definition 2.2.3. *[16] A* **probabilistic polynomial time (PPT) algorithm** *is an algorithm that takes as part of its input a security parameter n, runs in polynomial time in n, and may use internal randomness.*

One important value related to complexity that we will often use is the *linear algebra constant* $2 \leq \omega \leq 3$ which is the smallest constant such that for all $\varepsilon > 0$, two $n \times n$ matrices can be multiplied using $\mathcal{O}(n^{\omega+\varepsilon})$ operations [62].

We now define an important complexity class of problems that will be used in our work.

Definition 2.2.4. *[57] The class of computational problems for which solutions can be verified (but not necessarily computed) in polynomial time is called* **NP** *(for* **non-deterministic polynomial time***). An NP problem is* **NP-complete** *if all other NP problems can be reduced to it.*

NP-complete problems are thus the hardest problems in the class of NP problems. These are problems for which no efficient algorithm for computing a solution is known, even though solutions can be verified efficiently. NP-complete problems are often used as the basis for cryptographic schemes, since their hardness can be used to prevent against attacks.

We can now formally define a digital signature scheme, which will be the type of cryptographic scheme at the centre in this work. In the following definitions, we let $n \in \mathbb{N}$ be a *security parameter* whose value can be increased to increase the security of the scheme.

Definition 2.2.5. *[16] A* **digital signature scheme** *is an ordered triple of PPT algorithms* $\Pi =$ *(*Gen*,* Sign*,* Vrfy*):*

1. Gen*: key-generation algorithm that upon input* 1^n*, outputs a public-private key pair* (pk, sk)*.*

2. Sign*: signing algorithm that takes as input* sk *and a message* μ*, and outputs a signature* $\sigma \leftarrow Sign_{sk}(\mu)$*.*

3. Vrfy*: verification algorithm that given* pk, μ*, and* σ*, outputs* $b := Vrfy_{pk}(\mu, \sigma)$ *where* $b \in \{0, 1\}$*. In this context, 0 corresponds to "invalid" and 1 corresponds to "valid".*

In order to discuss the validity and security of a digital signature scheme, we require the notion of a *negligible function.*

Definition 2.2.6. *[16] A function* $f : \mathbb{N} \to \mathbb{R}$ *is* **negligible** *if for every polynomial* $p(\cdot)$*, there exists an integer* N *such that for all integers* $n > N$*,* $f(n) < \frac{1}{p(n)}$*. We use* negl *to denote an arbitrary negligible function.*

For example, the function $f(n) = \frac{1}{c^n}$ is negligible for any constant $c > 1$, since it approaches zero faster than the reciprocal of any polynomial.

For a digital signature scheme to be of practical use, the signing and verification algorithms must be compatible in the following way.

Definition 2.2.7. *[16] A digital signature scheme is* **correct** *if*

$$\text{Vrfy}_{pk}(\mu, \text{Sign}_{sk}(\mu)) = 1,$$

except possibly with some probability that is negligible in the security parameter n.

We use $\mathcal{A}$ to denote an adversary in the cryptographic context. For digital signature schemes, such an adversary is generally seen as a malicious party attempting to either learn the private key or forge a signature. We now define the game that allows us to define the security of a digital signature scheme.

Security Game Sig-forge. *[16]* Sig-forge$_{\mathcal{A},\Pi}(n)$*:*

1. $(pk, sk) \leftarrow \text{Gen}(1^n)$.

2. *$\mathcal{A}$ is given pk and access to an oracle* $\text{Sign}_{sk}(\cdot)$*. $\mathcal{A}$ outputs (μ, σ). Let Q denote the set of all queries that $\mathcal{A}$ asked the oracle.*

3. *$\mathcal{A}$ succeeds if and only if* $\text{Vrfy}_{pk}(\mu, \sigma) = 1$ *and $\mu \notin Q$. In this case, the experiment outputs 1.*

It is clear that if an adversary $\mathcal{A}$ can somehow learn the private key, then they can win the Sig-forge game, since they can just produce a signature as an honest signer would. This is not the criteria for winning however : they can win by simply forging a signature on one message, without having learned the private key. For signature schemes that we use in practice, we want the probability that the adversary wins the Sig-forge game to be negligible.

Definition 2.2.8. *[16] A signature scheme* $\Pi = ($Gen, Sign, Vrfy*)$ is* **secure** *if for all adversaries $\mathcal{A}$ (of some particular level of computational power, e.g., PPT), there exists a negligible function* negl *such that*

$$\Pr[\text{Sig-forge}_{\mathcal{A},\Pi}(n) = 1] \leq \text{negl}(n).$$

To evaluate the security of cryptographic schemes proposed in their standardization process, NIST has developed a standard list of security levels for these schemes to achieve. We will generally discuss NIST security levels I, III, and V that correspond to schemes requiring at least $2^{128}, 2^{192}$, and 2^{256} computational steps for an adversary to break [49]. Most schemes present distinct recommended parameter sets to achieve these three different levels of security.

We now define several important kinds of functions that are commonly used in the construction of cryptographic schemes, the first of which is a *pseudorandom function*.

Definition 2.2.9. *[36] Let n be a security parameter, let k be a key of length s bits, and let $f(k, x)$ be a function on keys and inputs x. Then f is a* **pseudorandom function** *if*

1. *f can be computed in polynomial time in s; and*

2. *if k is uniformly random, then f cannot be distinguished from a uniformly random function in polynomial time.*

Next, we define *one-way functions* which, intuitively, are easy to compute but hard to invert.

Definition 2.2.10. *[16] A function $f : \{0,1\}^* \to \{0,1\}^*$ is* **one-way** *if*

1. *There exists a polynomial-time algorithm M_f such that $M_f(x) = f(x)$ for all $x \in \{0,1\}^*$, and*

2. *For all PPT adversaries $\mathcal{A}$, there exists a negligible function* negl *such that*

$$\Pr_{x \xleftarrow{\$} \{0,1\}^n}[\mathcal{A}(1^n, f(x)) \in f^{-1}(f(x))] \leq \text{negl}(n).$$

It has been proven that pseudorandom functions exist if and only if one-way functions exist [29]. The existence of both of these kinds of functions is only conjectured, as the proof of their existence is equivalent to proving that the NP class of problems is not equal to the class of problems that can be solved in polynomial time (the famous conjecture that P≠NP) [32]. Thus, when we discuss one-way functions being used in practice, we are referring to functions that are conjectured to be one-way based on some hard computational problem.

A special class of one-way functions, called *hash functions* will be the most widely used in practice.

Definition 2.2.11. *[51] A* **hash function** *is a function on bits strings with fixed output length satisfying the following properties:*

1. *(One-way) It is computationally infeasible to find any input that maps to any pre-specified output;*

2. *(Collision-resistant) It is computationally infeasible to find any two distinct inputs that map to the same output.*

Finally, we define a special class of hash functions, whose output length does not have to be pre-specified.

Definition 2.2.12. *[50] An* **extendable-output function** *(often abbreviated as* **XOF***) is a hash function whose output string can be extended to any length.*

NIST has a set of standardized hash functions and extendable output functions that are generally used in the cryptographic schemes submitted to their standardization process. The standardized hash functions are SHA3-224, SHA3-256, SHA3-384,

and SHA3-512 (where the number after the hyphen denotes the length of the function's output), and the standardized extendable output functions are SHAKE128 and SHAKE256 (where the suffix denotes the number of bits of security offered by the function) [50].

We will use the conventional term *salt* to refer to initial random data (usually a bit string of some specified length) that is used as input to a one-way function at the beginning of a cryptographic protocol.

The final cryptographic notion we require is that of a *distinguisher*, which is an algorithm that attempts to distinguish between two probability distributions. These algorithms can be used to argue about the security of certain protocols that claim to generate random output.

Definition 2.2.13. *[60] A* **distinguisher** *is, broadly, a testing algorithm that outputs a single bit. In practice, the goal of a distinguisher is to distinguish between two probability distributions, one of which is usually uniformly random, to determine whether a given distribution is* **indistinguishable** *from the random one.*

In practice, given two probability distributions X and Y, they are indistinguishable if, given $x \xleftarrow{\$} X$ and $y \xleftarrow{\$} Y$, *for all* PPT distinguishers D, we have that

$$|\Pr[D(x) = 1] - \Pr[D(y) = 1]|$$

is negligible with respect to the input length.

2.3 Mathematical Background and Theory

We will require several definitions regarding multivariate polynomial rings to discuss multivariate-based digital signature schemes. We first recall the definition of an *ideal* of a ring.

Definition 2.3.1. *[20] A subset I of a commutative ring R is an* **ideal** *if it satisfies:*

1. $0 \in I$;
2. *If* $f, g \in I$, *then* $f + g \in I$; *and*
3. *If* $f \in I$ *and* $h \in R$, *then* $hf \in I$.

The ideals that we will discuss will be those *generated* by some set of polynomials in $\mathbb{F}[x_1, \ldots, x_n]$.

Lemma 2.3.2. *[20] If $f_1, \ldots, f_s \in \mathbb{F}[x_1, \ldots, x_n]$, then*

$$\langle f_1, \ldots, f_s \rangle := \left\{ \sum_{i=1}^{s} h_i f_i \;\middle|\; h_1, \ldots, h_s \in \mathbb{F}[x_1, \ldots, x_n] \right\}$$

is an ideal of $\mathbb{F}[x_1, \ldots, x_n]$. We will call $\langle f_1, \ldots, f_s \rangle$ the **ideal generated by** *$f_1, \ldots, f_s$.*

In algebraic geometry, the notion of an *affine variety* is used to characterize the intersection of several polynomials.

Definition 2.3.3. *[20] Let $f_1, \ldots, f_s \in \mathbb{F}[x_1, \ldots, x_n]$. Then we set*

$$V(f_1, \ldots, f_s) = \{(a_1, \ldots, a_n) \in \mathbb{F}^n \mid f_i(a_1, \ldots, a_n) = 0 \text{ for all } 1 \leq i \leq s\}.$$

We call $V(f_1, \ldots, f_s)$ the **affine variety** *defined by $f_1, \ldots, f_s$.*

We will define the notion of *monomials* and their *orderings*. These will be important tools when discussing algorithms that solve underlying hard problems in multivariate-based cryptography.

Definition 2.3.4. *[20] A* **monomial** *in $x_1, \ldots, x_n$ is a product of the form $x_1^{\alpha_1} \cdot x_2^{\alpha_2} \cdots x_n^{\alpha_n}$, where all of the exponents $\alpha_1, \ldots, \alpha_n$ are non-negative integers. We simplify the notation by letting $\alpha = (\alpha_1, \ldots, \alpha_n)$ so we can write the previous monomial as x^α.*

Definition 2.3.5. *[20] A* **monomial ordering** *$>$ on $\mathbb{F}[x_1, \ldots, x_n]$ is a relation $>$ on $\mathbb{Z}^n_{\geq 0}$, or equivalently, a relation on the set of monomials x^α, $\alpha \in \mathbb{Z}^n_{\geq 0}$, satisfying:*

1. *$>$ is a total (or linear) ordering on $\mathbb{Z}^n_{\geq 0}$. This means that for all pairs $\alpha, \beta \in \mathbb{Z}^n_{\geq 0}$, we have exactly one of $\alpha > \beta$, $\alpha = \beta$, or $\alpha < \beta$.*

2. *If $\alpha > \beta$ and $\gamma \in \mathbb{Z}^n_{\geq 0}$, then $\alpha + \gamma > \beta + \gamma$.*

3. *$>$ is a well-ordering on $\mathbb{Z}^n_{\geq 0}$. This means that every nonempty subset of $\mathbb{Z}^n_{\geq 0}$ has a smallest element under $>$.*

We define two particular monomial orderings that will be useful to us.

Definition 2.3.6. *[20]* **Lexicographic order**. *Let $\alpha = (\alpha_1, \ldots, \alpha_n), \beta = (\beta_1, \ldots, \beta_n) \in \mathbb{Z}^n_{\geq 0}$. We say $\alpha >_{lex} \beta$ if the leftmost nonzero entry of the vector $\alpha - \beta \in \mathbb{Z}^n$ is positive.*

Intuitively, when we compare two monomials using the lexicographic order, we first compare their exponents on x_1, then if these are equal, we compare their exponents on x_2, and so on.

Definition 2.3.7. *[20]* **Graded reverse lexicographic order**. *Let $\alpha, \beta \in \mathbb{Z}^n_{\geq 0}$. We say $\alpha >_{grevlex} \beta$ if*

- $|\alpha| = \sum_{i=1}^n \alpha_i > \sum_{i=1}^n \beta_i = |\beta|$ *(i.e., the total degree of α is greater than that of β), or*
- $|\alpha| = |\beta|$ *and the rightmost nonzero entry of $\alpha - \beta \in \mathbb{Z}^n$ is negative.*

Intuitively, to compare two monomials using the graded reverse lexicographic order, we first compare their total degrees, and if these are equal then we compare their exponents on x_n, followed by x_{n-1}, and so on.

To see how these two orders are fundamentally different, consider the monomials x_2^3 and x_1x_2 in $\mathbb{F}_2[x_1, x_2]$. We have $x_1x_2 >_{lex} x_2^3$ since x_1x_2 has a higher power of x_1. However, we have $x_2^3 >_{grevlex} x_1x_2$ since x_2^3 has a higher total degree.

In general, any monomial ordering that first compares the total degree of monomials is called *graded*. We will return to these orderings in Section 2.4.2.1 on Gröbner bases.

We will also require the notion of a *homogeneous* polynomial.

Definition 2.3.8. *A polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ is called* **homogeneous** *if it is a linear combination of monomials of the same total order.*

We also define the notion of a *symmetric bilinear map* on a vector space over a field.

Definition 2.3.9. *[46] Let V be a vector space over $\mathbb{F}$. Then $p : V \times V \to \mathbb{F}$ is a* **symmetric bilinear map** *if for all $\mathbf{u}, \mathbf{v}, \mathbf{w} \in V, \lambda \in \mathbb{F}$, the following hold.*

1. $p(\mathbf{u}, \mathbf{v}) = p(\mathbf{v}, \mathbf{u})$.
2. $p(\lambda\mathbf{u}, \mathbf{v}) = \lambda p(\mathbf{u}, \mathbf{v})$.
3. $p(\mathbf{u} + \mathbf{v}, \mathbf{w}) = p(\mathbf{u}, \mathbf{w}) + p(\mathbf{v}, \mathbf{w})$.

We now state two important theorems from group theory, beginning with the *Chinese Remainder Theorem*.

Theorem 2.3.10. *[47]* **Chinese Remainder Theorem**. *Let R be a ring and let I, J be ideals of R such that $I + J = R$. Then*

$$R/I \cap J \cong R/I \times R/J.$$

In particular, for $R = \mathbb{Z}$ and coprime integers n and m, the Chinese Remainder Theorem asserts that $\mathbb{Z}_{nm} \cong \mathbb{Z}_n \times \mathbb{Z}_m$.

Next, we state *Euler's Theorem*.

Theorem 2.3.11. *[17]* **Euler's Theorem**. *For any positive integer N, $a^{\varphi(N)} \equiv 1$ (mod N) for all $a \in \mathbb{Z}_N^*$.*

In particular, if $N = p$ is prime, then Euler's theorem tells us that for any $a \in \mathbb{Z}_p^*$, $a^{p-1} \equiv 1 \bmod p$.

Finally, we state a classical group theoretic problem that was historically widely used in cryptography.

Definition 2.3.12. *[16] Let G be a cyclic group with generator $g \in G$. The* **discrete logarithm problem** *asks, given an element $h \in G$, to find x such that $g^x = h$.*

The discrete logarithm problem is considered hard classically, but Shor proved that it can be solved in polynomial time on a quantum computer [55], so it cannot be used as the basis for post-quantum cryptographic schemes.

2.4 The Multivariate Quadratic Problem

In this section, we discuss the Multivariate Quadratic (MQ) problem and discuss algorithms for solving it. This problem, or some variation of it, is used as the basis for most multivariate-based digital signature schemes, including the ones we will discuss in Chapters 3 and 4.

2.4.1 The Generic Problem

We now state the Multivariate Quadratic (MQ) problem, which is the underlying hard problem used in most multivariate-based post-quantum digital signature schemes. Quadratic systems are used in practice since any higher degree system of equations can simply be reduced to a quadratic one by introducing additional variables, so there is no loss of generality in restricting to the quadratic case.

Let $\mathbb{F}_q$ be a finite field of size q, $a_{ij}^{(k)}, b_i^{(k)}, c^{(k)} \in \mathbb{F}_q$ for $k = 1, \ldots, m$ and $i, j = 1, \ldots, n$. Then for $k = 1, \ldots, m$, define $f_k \in \mathbb{F}_q[x_1, \ldots, x_n]$ by

$$f_k(x_1, \ldots, x_n) = \sum_{1 \le i \le j \le n} a_{ij}^{(k)} x_i x_j + \sum_{1 \le i \le n} b_i^{(k)} x_i + c^{(k)}.$$

Suppose there exists $\mathbf{x} \in \mathbb{F}_q^n$ such that $f_k(\mathbf{x}) = d_k$ for $k = 1, \ldots, m$.

The *MQ Problem* is to find any $\mathbf{x} \in \mathbb{F}_q^n$ satisfying this, given $\mathbf{f} = (f_k)_{k=1,\ldots,m}$ and $\mathbf{d} = (d_k)_{k=1,\ldots,m}$.

Note that we can also uniquely express the polynomials f_k in an instance of the MQ problem as

$$f_k(\mathbf{x}) = \mathbf{x}^\top \mathbf{A}^{(k)} \mathbf{x} + \mathbf{b}^{(k)^\top} \mathbf{x} + c^{(k)}$$

where $\mathbf{A}^{(k)} \in \mathbb{F}_q^{n\times n}$ is upper triangular, $\mathbf{b}^{(k)} \in \mathbb{F}_q^n$, and $c^{(k)} \in \mathbb{F}$. We can also allow the matrices $\mathbf{A}^{(k)}$ to be non-upper triangular, however the corresponding representation of f_k is no longer unique in this case.

The MQ problem is really one in algebraic geometry, as finding a solution to the MQ problem above is equivalent to finding a point in the affine variety

$$V(f_1 - d_1, \ldots, f_m - d_m).$$

If the system of equations for an instance of the MQ problem is not underdetermined, then we expect the number of solutions to be small (i.e., equal to 0, 1, 2, or 3). If it is underdetermined, then we expect there to be about q^{n-m} solutions [8].

This problem in its general form with random coefficients is NP-complete [8, 63] and many variations of it have been used to build cryptosystems, including digital signature schemes.

2.4.2 Algorithms for Solving the MQ Problem

We now give the complexity of what are currently considered the best algorithms for solving the MQ Problem. It should be noted that these algorithms solve the problem in its generic form, and more efficient algorithms could exist to break cryptosystems that use a more structured variant of the MQ Problem.

An important parameter upon which the complexity of many algorithms to solve the MQ problem depends is the *degree of regularity* of an ideal.

Definition 2.4.1. *[63, Definition 3.1] Let $(h_1, \ldots, h_m) \in \mathbb{F}_q[x_1, \ldots, x_n]^m$ be homogeneous polynomials. The* **degree of regularity** *of a homogeneous ideal $\mathcal{I} = \langle h_1, \ldots, h_m\rangle$ is defined by*

$$d_{reg} = \min\left\{ d \geq 0 \;\middle|\; \dim_{\mathbb{F}_q}(0 \cup \{f \in \mathcal{I}, \deg(f) = d\}) = \binom{n+d-1}{d} \right\}.$$

For non-homogeneous polynomials $(f_1, \ldots, f_m) \in \mathbb{F}_q[x_1, \ldots, x_n]^m$, the **degree of regularity** *is defined by that of the ideal $\langle f_1^h, \ldots, f_m^h\rangle$, where f_i^h is the homogeneous part of f_i of the highest degree.*

Intuitively, in the homogeneous case, the degree of regularity of an ideal $\mathcal{I}$ is the smallest $d \geq 0$ such that $\mathcal{I}$ contains all monomials in $\mathbb{F}_q[x_1, \ldots, x_n]$ of degree d.

Example 2.4.2. *If we take $\mathcal{I} = \langle x_1, \ldots, x_n\rangle$ in $\mathbb{F}_q[x_1, \ldots, x_n]$, the degree of regularity of $\mathcal{I}$ is 1 because $\mathcal{I}$ clearly contains all monomials of degree 1, but does not contain the monomial of degree 0 (i.e., the constant polynomial 1). If we instead take $\mathcal{I} =$*

$\langle x_1, \ldots, x_{n-1} \rangle$ in $\mathbb{F}_q[x_1, \ldots, x_n]$, the degree of regularity does not exist (i.e., $d_{reg} = \infty$), since for any $d \geq 0$, the monomial x_n^d is not in $\mathcal{I}$, so $\mathcal{I}$ does not contain all monomials of degree d.

2.4.2.1 Gröbner Bases

We now define and provide an overview of the use of *Gröbner bases* for solving the MQ problem. They are generally considered to be the most efficient way to solve a non-linear system of equations [8].

Definition 2.4.3. *[20] Given a monomial order for $\mathbb{F}_q[x_1, \ldots, x_n]$, a finite non-zero subset $G = \{g_1, \ldots, g_t\}$ of the ideal $\mathcal{I} \subseteq \mathbb{F}_q[x_1, \ldots, x_n]$ is a* **Gröbner basis** *if $\langle LT(g_1), \ldots, LT(g_t) \rangle = \langle LT(\mathcal{I}) \rangle$ where LT denotes the leading term of a polynomial with respect to the monomial order and $LT(\mathcal{I})$ is the set of leading terms of non-zero elements in $\mathcal{I}$.*

While the above definition is often the canonical description of a Gröbner basis, it is not the most useful one for actually computing such bases. The one most often used in practice is Buchberger's Criterion, which is based on so-called *S-polynomials*, which we now define.

Definition 2.4.4. *[20] Let $f, g \in \mathbb{F}_q[x_1, \ldots, x_n]$. Then the* ***S*-polynomial** *of f and g is defined as*

$$S(f, g) = \frac{x^\gamma}{LT(f)} \cdot f - \frac{x^\gamma}{LT(g)} \cdot g$$

where x^γ is the least common multiple of the leading monomials of f and g (the leading terms, but with coefficient 1).

In the following theorem, when we refer to dividing a polynomial by the set G, we refer to a specific division algorithm in which the polynomials of G are ordered and division proceeds in this order by subsequently dividing leading terms of the dividend by leading terms of the divisors.

Theorem 2.4.5. *[20]* **Buchberger's Criterion**. *Let $\mathcal{I}$ be a polynomial ideal. Then a basis $G = \{g_1, \ldots, g_t\}$ of $\mathcal{I}$ is a Gröbner basis of $\mathcal{I}$ if and only if for all pairs $i \neq j$, the remainder upon division of $S(g_i, g_j)$ by G (listed in some order) is zero.*

Buchberger's Criterion is easy to verify and allows for the following simple algorithm to compute a Gröbner basis for a given ideal. This was the first such algorithm developed, and though it is not used in practice today due to the development of more efficient modern algorithms, it provides an intuitive idea of how one might systematically find a Gröbner basis.

Theorem 2.4.6. *[20]* **Buchberger's Algorithm**. *Let $\mathcal{I} = \langle f_1, \ldots, f_s \rangle \neq \{0\}$ be a polynomial ideal. Then a Gröbner basis for $\mathcal{I}$ can be constructed in a finite number*

of steps by the following algorithm:

Input: $F = (f_1, \ldots, f_s)$
Output: a Gröbner basis $G = (g_1, \ldots, g_s)$ *for* $\mathcal{I}$ *with* $F \subseteq G$
$G := F$
REPEAT
    $G' := G$
    FOR *each pair* $\{p, q\}, p \neq q$ *in* G' DO
        $r :=$ *remainder upon division of* $S(p, q)$ *by* G'
        IF $r \neq 0$ THEN $G := G \cup \{r\}$
UNTIL $G = G'$
RETURN G

Intuitively, the above algorithm divides all possible S-polynomials by the basis and adds the remainder to the basis if it is non-zero. This algorithm will, in general, produce a large Gröbner basis that may contain some redundancies. In order to remove such redundancies, we define the concept of a *reduced* Gröbner basis, which can simply be computed from an existing Gröbner basis. While general Gröbner bases for an ideal are not unique, their equivalent reduced Gröbner bases are.

Definition 2.4.7. *[20] A Gröbner basis G for an ideal $\mathcal{I}$ is a* **reduced Gröbner basis** *if the leading coefficient of every polynomial in G is 1 and for all $p \in G$, no monomial of p lies in $\langle LT(G \setminus \{p\})\rangle$.*

The reason that Gröbner bases are of interest for solving the MQ problem is that when one is computed with respect to the lexicographic monomial order, it will contain a polynomial in a single variable that can be solved. This solution can then be back-substituted into the other equations, which will give rise to another single-variable equation that can be solved. This chain of back substitution allows for the whole system to be solved. This idea is formalized in the theorem below.

Theorem 2.4.8. *[20]* **The Elimination Theorem**. *Let $\mathcal{I} \subseteq \mathbb{F}_q[x_1, \ldots, x_n]$ be an ideal and let G be a Gröbner basis of $\mathcal{I}$ with respect to the lexicographic order where $x_1 > x_2 > \cdots > x_n$. Then for every $0 \leq l \leq n$, the set $G_l = G \cap \mathbb{F}_q[x_{l+1}, \ldots, x_n]$ is a Gröbner basis for the l-th* **elimination ideal** *$\mathcal{I}_l = \mathcal{I} \cap \mathbb{F}_q[x_{l+1}, \ldots, x_n]$.*

In practice, newer optimized algorithms such as the F_4 and F_5 algorithms due to Faugère are used to compute Gröbner bases [6, 24]. These algorithms transform the polynomial long division steps for adding new polynomials to the basis into steps involving matrices and linear algebra. They use the *Macauley matrix* of a set of polynomials, which has columns representing possible monomials up to some maximal degree d and rows representing the coefficients of the polynomials. This allows us to generalize the step in Buchberger's algorithm that picks pairs of polynomials and instead pick a larger subset of the polynomials in the basis to compute remainders.

The computational complexity of Macauley matrix-based algorithms is dominated by the complexity of Gaussian elimination on the largest matrix that will be encountered during the algorithm. The maximal monomial degree (which dictates the number of columns in the Macauley matrix) required to find a Gröbner basis is called the *solving degree*, but this is hard to compute in general. There are experimental results that indicate that the solving degree and the degree of regularity of an ideal are very close (in fact, almost equal), but this has not been proven in general [23]. The complexity can also vary depending on the monomial order chosen for the Gröbner basis. This is usually smallest for the graded reverse lexicographic order, so this is often used in the computation. A basis in this ordering can then be efficiently transformed into one in the lexicographic ordering, allowing for the application of the Elimination Theorem to solve the system of polynomials [19]. We give the complexity of the favoured F_5 algorithm in the summary table in Section 2.4.2.4.

2.4.2.2 Linearization Algorithms

Linearization algorithms, the most notable of which is the Xtended Linearization (XL) algorithm, transform an instance of the MQ problem into a very large system of linear equations, which can then be solved using Gaussian elimination. This is achieved by assigning a new variable to each monomial in $\mathbb{F}_q[x_1, \ldots, x_n]$ of degree less than or equal to 2, so that all quadratic terms are eliminated, and solving the resulting linear system of equations in these new variables arising from the polynomials f_k [4]. It has been proven that the XL algorithm can be represented as a redundant version of Gröbner basis algorithms such as F_4 and F_5 [4, 8], since it keeps track of all of the same polynomials and more as these algorithms. The complexity of linearization algorithms is dominated by the complexity of Gaussian elimination for the new linear system defined by the monomials of the quadratic system, and we provide the complexity of the XL algorithm in the summary table in Section 2.4.2.4.

2.4.2.3 Hybrid Algorithms

Another approach for systems with at least one solution is the hybrid approach, which consists of choosing k variables, evaluating them at randomly chosen values and attempting to solve the resulting system in $n - k$ variables by computing a Gröbner basis. If no solution exists, then a new guess of the values of k variables must be made. The optimal choice of k can vary depending on the context, but it is generally small relative to n. The complexity of the hybrid approach is also given in Table 2.1.

Algorithm	Complexity
F_5 [10]	$\mathcal{O}((m \cdot \binom{n+d_{reg}}{d_{reg}})^{\omega})$
XL [28]	$\mathcal{O}(\binom{n+d_{n,m}}{d_{n,m}}^{\omega})$
Hybrid [63]	$\mathcal{O}\left(min_{0\leq k\leq n}\left\{ q^k \cdot \left(m^{\omega} \cdot \binom{n-k+d_{reg}(k)-1}{d_{reg}(k)}^{\omega} + (n-k)2^{(n-k)\omega} \right) \right\} \right)$

Table 2.1: Computational complexity of various algorithms for solving the MQ problem.

2.4.2.4 Complexities of These Algorithms

To state the complexities of the three algorithms we discussed, we let d_{reg} be the degree of regularity of the system of polynomials and $2 \leq \omega \leq 3$ be the linear algebra constant. Furthermore, for the hybrid approach, we define $d_{\text{reg}}(k)$ to be the maximum degree of regularity of all the systems in which we have guessed the values of the last k variables,

$$\{\{f_1(x_1, \ldots, x_{n-k}, v_1, \ldots, v_k), \ldots, f_m(x_1, \ldots, x_{n-k}, v_1, \ldots, v_k)\} \mid (v_1, \ldots, v_k) \in \mathbb{F}_q^k\}.$$

Finally, we define the *operating degree* $d_{n,m}$ of the XL algorithm to be the smallest $d > 0$ such that the coefficient of t^d in the power series expansion of

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

is less than or equal to 0.

Table 2.1 provides the computational complexity of the F_5, XL, and hybrid algorithms [8, 22]. We observe that the complexity of these algorithms is generally exponential in the number of variables n or the degree of regularity, but is less sensitive to the number of equations m. The hybrid algorithm is also exponential in the size of the field over which the polynomials are defined, since it involves making guesses for the values of some variables. The designers of cryptographic schemes based on the MQ problem must balance their choice for all of these parameters to ensure no attack becomes efficient, while still maintaining reasonable running times of the algorithms themselves. Note that the degree of regularity is hard to compute in general [10], so predicting the precise complexity of one of these algorithms for a given instance of the MQ problem can be difficult.

Chapter 3

Unbalanced Oil and Vinegar Signature Schemes

The Unbalanced Oil and Vinegar (UOV) problem is a structured variant of the MQ problem that is used as the basis for many proposed post-quantum digital signature schemes. In these schemes, which were first introduced by Patarin in 1997 [54], the public key polynomials are disguised to appear to be a random MQ instance, but the secret key includes a trapdoor transformation that allows the Signer to easily find a pre-image of the quadratic system.

3.1 The UOV Problem

We first define the special kinds of polynomials used in the UOV scheme, in which the variables are divided into two sets : "vinegar" variables, which can appear in any quadratic terms, and "oil" variables, which can only appear in quadratic terms with vinegar variables.

Definition 3.1.1. *[22, Definition 1] Given* $1 \leq m < n$*, an* (n, m)**-OV-polynomial** $f \in \mathbb{F}_q[x_1, \ldots, x_n]$ *is defined as*

$$f(x_1, \ldots, x_n) = \sum_{i=1}^{n-m} \sum_{j=1}^{n} \alpha_{i,j} x_i x_j + \sum_{j=1}^{n} \beta_j x_j + \gamma$$

where $\alpha_{i,j}, \beta_j, \gamma \in \mathbb{F}_q$ *for all* $i \in [n-m], j \in [n]$*. Such a polynomial* f *has unique matrix representation as*

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{b}^\top \mathbf{x} + \gamma$$

where

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}^{(1)} & \mathbf{A}^{(2)} \\ \mathbf{0}_{m\times(n-m)} & \mathbf{0}_{m\times m} \end{bmatrix}$$

for some $\mathbf{A}^{(1)} \in \mathbb{F}_q^{(n-m)\times(n-m)}$ *upper-triangular,* $\mathbf{A}^{(2)} \in \mathbb{F}_q^{(n-m)\times m}$*, and* $\mathbf{b} \in \mathbb{F}_q^n$*.*

Recall that for the matrix representing the quadratic part of any MQ polynomial, we can instead take an equivalent non-upper-triangular representation, which is no longer unique. In the case of an OV-polynomial, this means that we relax the restriction that the bottom left $\mathbf{0}_{m\times(n-m)}$ submatrix be zero, and only require the bottom right submatrix $\mathbf{0}_{m\times m}$ to remain.

In Definition 3.1.1, the first $n-m$ variables are the vinegar variables and the last m are the oil variables. This leads to the definition of two key subspaces of $\mathbb{F}_q^n$ that we will work with throughout this chapter.

Definition 3.1.2. *[39, Definition 1,2] We define the* **oil subspace** *of* $\mathbb{F}_q^n$ *as*

$$\mathcal{O} = \{\mathbf{x} = (x_1, \ldots, x_n)^\top \in \mathbb{F}_q^n \mid x_1 = \cdots = x_{n-m} = 0\}.$$

Similarly, the **vinegar subspace** *is*

$$\mathcal{V} = \{\mathbf{x} = (x_1, \ldots, x_n)^\top \in \mathbb{F}_q^n \mid x_{n-m+1} = \cdots = x_n = 0\}.$$

The map that forms the basis for the UOV problem consists of a collection of OV-polynomials with the same parameters.

Definition 3.1.3. *[22, Definition 3] A* **UOV central map** *in relation to* $\mathsf{params} = (n, m, q)$ *is* $\mathcal{F} : \mathbb{F}_q^n \to \mathbb{F}_q^m$*,* $\mathbf{x} \mapsto \mathcal{F}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}))^\top$ *where the* f_k*'s are* (n, m)*-OV polynomials over* $\mathbb{F}_q$*. A* **UOV map** *in relation to* $\mathsf{params} = (n, m, q)$ *is* $\mathcal{P} = \mathcal{F} \circ \mathcal{T}$ *where* $\mathcal{F}$ *is a UOV central map in relation to* params *and* $\mathcal{T} : \mathbb{F}_q^n \to \mathbb{F}_q^n$ *is an invertible affine transformation. We call* $\mathcal{P}$ **random** *if the coefficients of the polynomials* f_k *in* $\mathcal{F}$ *as well as the affine transformation* $\mathcal{T}$ *are chosen uniformly at random.*

Given a UOV central map $\mathcal{F}$, for any $\mathbf{t} \in \mathbb{F}_q^m$, we can efficiently solve the system

$$\mathcal{F}(\mathbf{x}) = \mathbf{t} \tag{3.1.1}$$

as follows.

We first fix the vinegar variables $x_1, \ldots, x_{n-m}$ to random values in $\mathbb{F}_q$. Then, the matrix $\mathbf{x}^\top\mathbf{A}$ for each OV-polynomial in $\mathcal{F}$ is a constant matrix independent of the oil variables, so the system (3.1.1) becomes a linear system in $x_{n-m+1}, \ldots, x_n$. We can thus attempt to solve for the oil variables efficiently using Gaussian elimination. If no solution exists, we simply try a new random choice for the vinegar variables. Note that this is a system of m equations in m variables, so the probability that it will

have a solution is greater than the probability that a random $m \times m$ matrix over $\mathbb{F}_q$ is invertible [38], which is

$$\left(1 - \frac{1}{q}\right)\left(1 - \frac{1}{q^2}\right) \dots \left(1 - \frac{1}{q^{m-1}}\right).$$

Therefore, with high probability, a solution $\mathbf{x}$ will be found with a low number of iterations of Gaussian elimination, whose complexity is $\mathcal{O}(m^\omega)$.

Since it can be used to solve systems of equations efficiently, the UOV central map $\mathcal{F}$, along with the transformation $\mathcal{T}$, will be the secret key used by the Signer. The UOV map $\mathcal{P}$ will be the public key used by the Verifier. The transformation $\mathcal{T}^{-1}$ maps $\mathcal{O}$ and $\mathcal{V}$ to random complementary affine subspaces of $\mathbb{F}_q^n$. Without identifying $\mathcal{T}^{-1}(\mathcal{V})$ (or $\mathcal{T}^{-1}(\mathcal{O})$, as we will discuss in Section 3.2.1), the Verifier cannot create a simple linear system as the Signer does.

We now define the hard cryptographic problem that forms the foundation for the UOV signature scheme.

Definition 3.1.4. *Fix Setup(·) that outputs* params $= (n, m, q)$ *on input* 1^κ. *The* **UOV problem** *in relation to Setup(·) is* (t, ϵ)-**hard** *if there exists no algorithm that, given* params *and a random UOV map* $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ *in relation to* params, *on input* $\mathbf{z} = \mathcal{P}(\mathbf{w})$ *with* $\mathbf{w} \xleftarrow{\$} \mathbb{F}_q^n$, *outputs* $\mathbf{w}'$ *such that* $\mathcal{P}(\mathbf{w}') = \mathbf{z}$ *with probability no less than* $\epsilon(\kappa)$ *in processing time* $t(\kappa)$.

The UOV problem is indeed believed to be hard with certain restrictions on its parameters. In particular, it is called *Unbalanced* Oil and Vinegar because we require $n > 2m$ (i.e., more vinegar variables than oil variables), due to the 1999 attack by Kipnis and Shamir [39] on *Balanced* Oil and Vinegar, where $n = 2m$. We discuss this attack in greater detail in Section 3.2.1.1.

However, when $n \in \mathcal{O}(m)$, the hardness of the UOV problem is not considered to be equivalent to the hardness of the generic MQ problem [13], which is known to be NP-complete, as mentioned in Section 2.4.

3.2 The UOV Signature Scheme

We now describe the general structure of UOV-based digital signature schemes.

Key Generation: The algorithm KeyGen takes as input params $= (n, m, q)$ and outputs the key pair (pk,sk). We have sk=$(\mathcal{F}, \mathcal{T})$ where $\mathcal{F}$ is a random UOV central map in relation to params and $\mathcal{T} : \mathbb{F}_q^n \to \mathbb{F}_q^n$ is a random invertible affine transformation. Then pk=$\mathcal{P}$ where $\mathcal{P} := \mathcal{F} \circ \mathcal{T}$.

Signing: The algorithm Sign takes as input params, sk, and a message $\mu \in \{0,1\}^*$. The Signer first generates $\mathbf{t} = \text{Hash}(\mu) \in \mathbb{F}_q^m$ where Hash is a hash function into $\mathbb{F}_q^m$. They now wish to find a pre-image of $\mathbf{t}$ under $\mathcal{F}$. They randomly sample $\mathbf{v} \in \mathbb{F}_q^{n-m}$ (this corresponds to randomly assigning values to the vinegar variables) and solve the resulting linear system for $\mathbf{u} \in \mathbb{F}_q^m$ such that $\mathcal{F}\left(\begin{bmatrix}\mathbf{v}\\ \mathbf{u}\end{bmatrix}\right) = \mathbf{t}$. If no such $\mathbf{u}$ exists, then they choose a new $\mathbf{v}$ and try again. Once suitable $\mathbf{v}$ and $\mathbf{u}$ are found, they return the signature $\sigma := \mathcal{T}^{-1}\left(\begin{bmatrix}\mathbf{v}\\ \mathbf{u}\end{bmatrix}\right)$.

Verification: The algorithm Verify takes as input params, pk, and the message/signature pair (μ, σ). The Verifier generates $\mathbf{t} = \text{Hash}(\mu)$. They accept the signature if and only if $\mathcal{P}(\sigma) = \mathbf{t}$.

This digital signature scheme is correct for an honest Signer and Verifier since

$$\mathcal{P}(\sigma) = (\mathcal{F} \circ \mathcal{T})\left(\mathcal{T}^{-1}\left(\begin{bmatrix}\mathbf{v}\\ \mathbf{u}\end{bmatrix}\right)\right) = \mathcal{F}\left(\begin{bmatrix}\mathbf{v}\\ \mathbf{u}\end{bmatrix}\right) = \mathbf{t}.$$

3.2.1 Known Attack Models

We now discuss specialized attacks on UOV that take advantage of the special structure of its polynomials. These generally perform better than applying general Gröbner basis or hybrid approaches for solving the MQ problem, as in Section 2.4.2, to the public UOV map [13].

3.2.1.1 Kipnis-Shamir Attack

We present an overview of the Kipnis-Shamir attack on *Balanced* Oil and Vinegar, which was published in 1999 [39]. The ideas behind this attack can be extended to the unbalanced case, however, they become inefficient and thus do not break the signature scheme.

We consider the balanced case of Oil and Vinegar, i.e., $n = 2m$. For simplicity, we present the attack only in the case that the central map $\mathcal{F}$ is comprised of homogeneous quadratic polynomials and the transformation $\mathcal{T}$ is linear (as opposed to affine). The attack on the more general case requires more technical detail to describe, but has similar complexity and uses the same core ideas [39].

Let f_i be one of the OV-polynomials in $\mathcal{F}$. Then $f_i(\mathbf{x}) = \mathbf{x}^\top \mathbf{F}^{(i)} \mathbf{x}$ where we choose the $n \times n$ matrix $\mathbf{F}^{(i)}$ to be of the general form $\begin{bmatrix}\mathbf{F}_1 & \mathbf{F}_2\\ \mathbf{F}_3 & \mathbf{0}_{m\times m}\end{bmatrix}$, so that it may be invertible. Let $\mathbf{T}$ be the matrix representing the linear transformation $\mathcal{T}$. Then the matrix representing the corresponding public polynomial is $\mathbf{P}^{(i)} = \mathbf{T}^\top \mathbf{F}^{(i)} \mathbf{T}$.

Note that our requirement that $\mathcal{F}$ and $\mathcal{T}$ be homogeneous ensured that the public map $\mathcal{P}$ is comprised of homogeneous quadratic polynomials as well.

The following lemma provides a connection between the oil and vinegar subspaces of $\mathbb{F}_q^n$ and the matrices representing the polynomials in a UOV map.

Lemma 3.2.1. *[39, Lemma 3] Suppose that $(\mathcal{F}, \mathcal{T})$ is the private key pair for an (n, m) UOV digital signature scheme where $\mathcal{F}$ is comprised of homogeneous quadratic polynomials represented by the matrices $\mathbf{F}^{(i)}$ for $i \in [m]$. Suppose that $\mathbf{P}^{(i)}$ is the matrix representing the i-th polynomial in the corresponding public key $\mathcal{P}$. Then for any $1 \leq m < n$, we have the following.*

1. *For any $\mathbf{u_1}, \mathbf{u_2} \in \mathcal{O}$, we have $\mathbf{u_1}^\top \mathbf{F}^{(i)} \mathbf{u_2} = 0$ for all $i = 1, \ldots, m$;*
2. *For any $\mathbf{v_1}, \mathbf{v_2} \in \mathcal{T}^{-1}(\mathcal{O})$, we have $\mathbf{v_1}^\top \mathbf{P}^{(i)} \mathbf{v_2} = 0$ for all $i = 1, \ldots, m$.*

The proof of the lemma is immediate from matrix multiplication, Definition 3.1.2 of the oil and vinegar subspaces of $\mathbb{F}_q^n$, and the structure of OV-polynomials.

The goal of the Kipnis-Shamir attack is to find the space $\mathcal{T}^{-1}(\mathcal{O})$, and to use this to reconstruct the "relevant" part of the transformation $\mathcal{T}$. This in turn will allow an attacker to create their own central map that allows for efficient forgery of signatures. The following lemma provides the mathematical tools that we will require to compute $\mathcal{T}^{-1}(\mathcal{O})$.

Lemma 3.2.2. *[39, Lemma 4] Suppose $n = 2m$ and let $\mathbf{H}_1 = \sum_{i=1}^m a_i \mathbf{F}^{(i)}$ and $\mathbf{H}_2 = \sum_{i=1}^m b_i \mathbf{F}^{(i)}$ be linear combinations over $\mathbb{F}_q$ of the matrices $\mathbf{F}^{(i)}$ and furthermore assume that $\mathbf{H}_1$ is invertible. Then the following hold.*

1. *$\mathbf{H}_2(\mathcal{O}) \subseteq \mathcal{V}$;*
2. *$\mathbf{H}_1(\mathcal{O}) = \mathcal{V}$ and $\mathbf{H}_1^{-1}(\mathcal{V}) = \mathcal{O}$;*
3. *$\mathcal{O}$ is an invariant subspace of $\mathbf{H} = \mathbf{H}_1^{-1}\mathbf{H}_2$.*

Proof: Note that $\mathbf{H}_1$ and $\mathbf{H}_2$ still have the same form as the matrices $\mathbf{F}^{(i)}$ with a bottom right $m \times m$ zero submatrix, which guarantees that $\mathbf{H}_i(\mathcal{O}) \subseteq \mathcal{V}$ for $i = 1, 2$.

Then, if we assume that $\mathbf{H}_1$ is invertible, we know it must be an injective map. Since $n = 2m$, we have $\dim(\mathcal{O}) = \dim(\mathcal{V})$, which gives the second assertion.

The third assertion then follows immediately from 1. and 2. ∎

Similarly, if

$$\mathbf{W}_1 = \sum_{i=1}^{m} a_i \mathbf{P}^{(i)} = \mathbf{T}^\top \left(\sum_{i=1}^{m} a_i \mathbf{F}^{(i)} \right) \mathbf{T}$$

and

$$\mathbf{W}_2 = \sum_{i=1}^{m} b_i \mathbf{P}^{(i)} = \mathbf{T}^\top \left(\sum_{i=1}^{m} b_i \mathbf{F}^{(i)} \right) \mathbf{T}$$

are linear combinations of the matrices $\mathbf{P}^{(i)}$ with $\mathbf{W}_1$ invertible, then it follows from Lemma 3.2.2 that $\mathcal{T}^{-1}(\mathcal{O})$ is an invariant subspace of the matrix $\mathbf{W} = \mathbf{W}_1^{-1}\mathbf{W}_2$.

We can thus compute $\mathcal{T}^{-1}(\mathcal{O})$ as the common invariant subspace of all matrices that have the form of $\mathbf{W}$. There exist efficient algorithms to do so, which we will not detail here but can be found in [39].

Proposition 3.2.3. *Suppose an attacker has learned the subspace $\mathcal{T}^{-1}(\mathcal{O})$. Then they can create a UOV central map that is equivalent to the secret key that they can use to forge signatures.*

Proof: Suppose an attacker has a basis for $\mathcal{T}^{-1}(\mathcal{O})$. Then they can construct an invertible linear transformation $\tilde{\mathcal{T}} : \mathbb{F}_q^n \to \mathbb{F}_q^n$ such that $\tilde{\mathcal{T}}(\mathcal{T}^{-1}(\mathcal{O})) = \mathcal{O}$ (i.e., $\mathcal{O}$ has the same pre-image under $\mathcal{T}$ and $\tilde{\mathcal{T}}$).

Let $\tilde{\mathbf{T}}$ be the matrix representing $\tilde{\mathcal{T}}$. For all $i \in [m]$, the attacker can set

$$\mathbf{G}^{(i)} = {\tilde{\mathbf{T}}^{-1}}^\top \mathbf{P}^{(i)} \tilde{\mathbf{T}}^{-1}.$$

Then, for any $\mathbf{u_1}, \mathbf{u_2} \in \mathcal{O}$, we have

$$\mathbf{u_1}^\top \mathbf{G}^{(i)} \mathbf{u_2} = \mathbf{u_1}^\top {\tilde{\mathbf{T}}^{-1}}^\top \mathbf{P}^{(i)} \tilde{\mathbf{T}}^{-1} \mathbf{u_2} = 0$$

since $\tilde{\mathbf{T}}^{-1}\mathbf{u_1}, \tilde{\mathbf{T}}^{-1}\mathbf{u_2} \in \mathcal{T}^{-1}(\mathcal{O})$.

Since this is true for any $\mathbf{u_1}, \mathbf{u_2} \in \mathcal{O}$, we conclude that $\mathbf{G}^{(i)}$ must have a bottom right $m \times m$ zero submatrix, and is thus the matrix representing an (n, m)-OV polynomial.

Therefore, the attacker can use $(\mathcal{G}, \tilde{\mathcal{T}})$, where $\mathcal{G}$ is the map defined by the matrices $\mathbf{G}^{(i)}$, as an equivalent private key to sign signatures. This is because, for any message μ, they can proceed as does the honest Signer by guessing values of the vinegar variables $\mathbf{v}$ to solve the linear system $\mathcal{G}\left(\begin{bmatrix} \mathbf{v} \\ \mathbf{u} \end{bmatrix} \right) = \text{Hash}(\mu)$ for the oil variables $\mathbf{u}$. Then they can send the forged signature $\sigma = \tilde{\mathcal{T}}^{-1}\left(\begin{bmatrix} \mathbf{v} \\ \mathbf{u} \end{bmatrix} \right)$.

Then, a Verifier will accept this signature since

$$\mathcal{P}(\sigma) = (\mathcal{G} \circ \tilde{\mathcal{T}})\left(\tilde{\mathcal{T}}^{-1}\left(\begin{bmatrix} \mathbf{v} \\ \mathbf{u} \end{bmatrix} \right) \right) = \mathcal{G}\left(\begin{bmatrix} \mathbf{v} \\ \mathbf{u} \end{bmatrix} \right) = \mathbf{t}.$$

Note that the proof of Proposition 3.2.3 did not require that $n = 2m$, so it applies in the unbalanced case as well. Applying this attack to UOV requires approximately

$$q^{n-2m}n^{2.8}(2(\log(q))^2 + \log(q)) \tag{3.2.1}$$

operations [13], which is in $\mathcal{O}(q^{n-2m}n^{2.8}(\log(q))^2)$.

The Kipnis-Shamir attack introduced the key idea behind many subsequent attacks on UOV : if an attacker can find the pre-image of the oil subspace by the affine transformation $\mathcal{T}$, then the scheme is broken.

3.2.1.2 Intersection Attack

We now present the intersection attack that was first published in 2021 [11]. This attack combines ideas from the Kipnis-Shamir attack along with a more direct approach for solving MQ systems, but does not completely break the UOV signature scheme for large enough parameters. For this attack, we assume that $n < 3m$. This assumption is not restrictive, since most instances of UOV in the literature use $2m < n < 3m$ for the sake of efficiency and to ensure that the public systems of equations are not too underdetermined [13, 22].

We are once again concerned with finding $\mathcal{T}^{-1}(\mathcal{O})$, but we take a slightly different approach by first attempting to find just two vectors in the space, and then using a system of equations formed using these vectors to find others.

As in the Kipnis-Shamir attack, we will assume for simplicity that the public polynomials for the UOV map are homogeneous quadratic maps.

First, for any public polynomial p_i in $\mathcal{P}$, we define the *polar form* (sometimes also called the *differential*) of p_i as

$$p_i'(\mathbf{x}, \mathbf{y}) = p_i(\mathbf{x} + \mathbf{y}) - p_i(\mathbf{x}) - p_i(\mathbf{y})$$

for all $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$.

It is straightforward to observe using Definition 2.3.9 that p_i' is a symmetric bilinear map.

Lemma 3.2.4. *For any two vectors $\mathbf{v}_1, \mathbf{v}_2 \in \mathcal{T}^{-1}(\mathcal{O})$ and public polynomial p_i, we have $p_i'(\mathbf{v}_1, \mathbf{v}_2) = 0$.*

Proof: Let $\mathbf{P}^{(i)}$ be the matrix representing p_i and let $\mathbf{v}_1, \mathbf{v}_2 \in \mathcal{T}^{-1}(\mathcal{O})$. Then $(\mathbf{v}_1 + \mathbf{v}_2) \in \mathcal{T}^{-1}(\mathcal{O})$ as well, so by Lemma 3.2.1,

$$\begin{aligned} p_i'(\mathbf{v_1}, \mathbf{v_2}) &= p_i(\mathbf{v_1} + \mathbf{v_2}) - p_i(\mathbf{v_1}) - p_i(\mathbf{v_2}) \\ &= (\mathbf{v_1} + \mathbf{v_2})^\top \mathbf{P}^{(i)}(\mathbf{v_1} + \mathbf{v_2}) - \mathbf{v_1}^\top \mathbf{P}^{(i)}\mathbf{v_1} - \mathbf{v_2}^\top \mathbf{P}^{(i)}\mathbf{v_2} \\ &= 0. \end{aligned}$$

∎

We will use this lemma to solve for subsequent nonzero vectors in $\mathcal{T}^{-1}(\mathcal{O})$ once we have found just two, since if we have found $\mathbf{v}_1, \mathbf{v}_2 \in \mathcal{T}^{-1}(\mathcal{O})$, then any further $\mathbf{v} \in \mathcal{T}^{-1}(\mathcal{O})$ will satisfy

$$\begin{cases} \mathcal{P}(\mathbf{v}) = 0 \\ \mathcal{P}'(\mathbf{v_1}, \mathbf{v}) = 0 \\ \mathcal{P}'(\mathbf{v_2}, \mathbf{v}) = 0 \end{cases}$$

where the last two equations are linear, allowing us to solve for $\mathbf{v}$ more efficiently than in the case of a generic quadratic system. The advantage of this approach is that each time we find another $\mathbf{v} \in \mathcal{T}^{-1}(\mathcal{O})$, we can append another linear equation to this system, reducing the search space by yet another dimension.

Consider for each $i \in [m]$ a matrix $\mathbf{M}_i$ such that

$$p_i'(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{M}_i \mathbf{y}.$$

This matrix will be invertible with high probability [11], so choose two indices $i, j \in [m]$ such that $\mathbf{M}_i$ and $\mathbf{M}_j$ are invertible.

Now, our goal is to find a vector $\mathbf{x} \in \mathbf{M}_i(\mathcal{T}^{-1}(\mathcal{O})) \cap \mathbf{M}_j(\mathcal{T}^{-1}(\mathcal{O}))$, because we will then have two vectors $\mathbf{v}_1 = \mathbf{M}_i^{-1}\mathbf{x}$ and $\mathbf{v}_2 = \mathbf{M}_j^{-1}\mathbf{x}$ that are in the subspace $\mathcal{T}^{-1}(\mathcal{O})$. It is shown in [11] that this intersection has dimension at least $3m - n$, so if we assume that $n < 3m$, then we are guaranteed to have non-trivial solutions.

Since the public map $\mathcal{P}$ vanishes on $\mathcal{T}^{-1}(\mathcal{O})$, we know that such a vector $\mathbf{x}$ must satisfy the following system of equations:

$$\begin{cases} \mathcal{P}(\mathbf{M}_i^{-1}\mathbf{x}) = 0 \\ \mathcal{P}(\mathbf{M}_j^{-1}\mathbf{x}) = 0 \\ \mathcal{P}'(\mathbf{M}_i^{-1}\mathbf{x}, \mathbf{M}_j^{-1}\mathbf{x}) = 0 \end{cases}$$

where $\mathcal{P}' = (p_1', \ldots, p_m')$ is the map comprised of the polar forms of the public polynomials.

This is a quadratic system of $3m$ equations in n variables, but since we know that the intersection $\mathbf{M}_i(\mathcal{T}^{-1}(\mathcal{O})) \cap \mathbf{M}_j(\mathcal{T}^{-1}(\mathcal{O}))$ has dimension at least $3m - n$, there are at least $3m - n$ linearly independent solutions. Therefore, we can impose up to $3m - n - 1$ linear restrictions (such as setting this many variables equal to zero) and still be guaranteed a nontrivial solution $\mathbf{x}$. This results in a quadratic system of $3m$ equations in $2n - 3m < n$ variables. This system, with a smaller number of variables, can then be solved using one of the methods for the general MQ problem discussed in Section 2.4.2.

We will then use the two vectors $\mathbf{v}_1 = \mathbf{M}_i^{-1}\mathbf{x}$ and $\mathbf{v}_2 = \mathbf{M}_j^{-1}\mathbf{x}$ to solve for subsequent vectors in $\mathcal{T}^{-1}(\mathcal{O})$ as above.

In [11], the author shows that the intersection attack is more efficient than previous known attacks on UOV, such as applying a direct variant of Kipnis-Shamir to the unbalanced case. They also demonstrate an optimized version of the attack in the case where $n < 2.5m$, where they start by finding a vector in the intersection of more than two subspaces $\mathbf{M}_i(\mathcal{T}^{-1}(\mathcal{O}))$, in order to reduce the number of variables even further.

Since the intersection attack, unlike Kipnis-Shamir, does not circumvent the need to solve quadratic systems, it is inefficient when the parameters m and n are chosen to be sufficiently large.

3.2.1.3 MinRank Attack

MinRank-based attacks on UOV are based on the well-known cryptographic primitive of the MinRank problem. This problem, defined below, is used as the basis for multiple submissions to the NIST round 1 additional signature call [1, 2], but can also be used to attack certain multivariate and code-based signature schemes.

Definition 3.2.5. *[11] Given k $n \times m$ matrices $L_1, \ldots, L_k$ over $\mathbb{F}_q$ and a target rank r, find coefficients $y_1, \ldots, y_k \in \mathbb{F}_q$ not all zero such that the linear combination $\sum_{i=1}^{k} y_i L_i$ has rank at most r.*

To obtain an instance of the MinRank problem from one of the UOV problem, we can derive the matrices $L_1, \ldots, L_m$ from the matrices representing the quadratic part of the public polynomials $p_1, \ldots, p_m$, or their corresponding polar forms. However, in order to determine the rank r that should be used, we must rely on some structure in the particular UOV-based signature scheme we are working with. For instance, taking $r = n - m$, finding even just one of the matrices representing the quadratic part of the public polynomials with rank r would allow us to compute $\mathcal{T}^{-1}(\mathcal{O})$, but this is nontrivial because of the non-uniqueness of these matrix representations.

The MinRank attack is not nearly as efficient as the other attacks described previously

on standard UOV [13], however it has been used to break more structured variants of UOV. Most notably, a MinRank attack was used to break the Rainbow Digital Signature Scheme [12] which was advanced to Round 3 of the previous NIST call for digital signature schemes [21] prior to the publication of the attack.

3.3 NIST Submission: TUOV

We studied one of the UOV-based submissions to the NIST standardization process in 2023 called TUOV, or Triangular Unbalanced Oil and Vinegar [22]. The polynomials used in this scheme are a slight generalization of OV-polynomials, which they claim makes the scheme more secure than the standard UOV signature scheme.

In this section, we first define TUOV maps and the TUOV problem, which is the hard problem underlying this signature scheme. We then describe the key generation, signing, and verification algorithms of TUOV. Finally, we discuss the security analysis of TUOV.

We show how a security reduction from [22] that was claimed to be unique to TUOV can also be applied to standard UOV, disproving the notion that TUOV is more secure than UOV.

In their implementation, TUOV only makes use of OV-polynomials, rather than the new polynomials they define. It also uses parameters very similar to a standard UOV digital signature scheme, which raises the question of the practical advantage of TUOV over the more classical UOV. We discuss this comparison further in Section 3.3.4.

3.3.1 TUOV Maps and the TUOV Problem

We first define the new kind of polynomials defined for the TUOV signature scheme in [22].

Definition 3.3.1. *[22, Definition 2] Given $1 \leq d < m < n$, an (n, m, d)-***TOV-polynomial** *f over $\mathbb{F}_q$ is an MQ-polynomial over $\mathbb{F}_q$ of the form*

$$\sum_{i=n-m+1}^{n-m+d} \sum_{j=n-m+1}^{n-m+d} \alpha_{i,j} x_i x_j + \sum_{i=1}^{n-m} \sum_{j=1}^{n} \alpha_{i,j} x_i x_j + \sum_{j=1}^{n} \beta_j x_j + \gamma.$$

It has a unique matrix representation as

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{b}^\top \mathbf{x} + \gamma$$

where

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}^{(1)} & \mathbf{A}^{(2,d)} & \mathbf{A}^{(3,d)} \\ \mathbf{0}_{d\times(n-m)} & \mathbf{A}^{(4,d)} & \mathbf{0}_{d\times(m-d)} \\ \mathbf{0}_{(m-d)\times(n-m)} & \mathbf{0}_{(m-d)\times d} & \mathbf{0}_{(m-d)\times(m-d)} \end{bmatrix}$$

with $\mathbf{A}^{(1)} \in \mathbb{F}_q^{(n-m)\times(n-m)}$ *and* $\mathbf{A}^{(4,d)} \in \mathbb{F}_q^{d\times d}$ *upper-triangular,* $\mathbf{A}^{(2,d)} \in \mathbb{F}_q^{(n-m)\times d}$, $\mathbf{A}^{(3,d)} \in \mathbb{F}_q^{(n-m)\times(m-d)}$, $\mathbf{b} \in \mathbb{F}_q^n$ *and* $\gamma \in \mathbb{F}_q$.

Note that the superscript d is used in some of the submatrices in the above definition as a way to indicate that their dimension depends on d.

In essence, TOV-polynomials are a slight generalization of OV-polynomials, in which more non-zero entries are permitted in the matrices representing the quadratic part. The MQ-maps used in [22] are TUOV-maps rather than UOV-maps, defined as follows.

Definition 3.3.2. *[22, Definition 4] Given* $1 \leq m_1 \leq m_2 < m < n$, *a* **TUOV central map** *in relation to* $\mathsf{params} = (n, m, m_1, m_2, q)$ *is a function* $\mathcal{F} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ *sending* $\mathbf{x} \mapsto \mathcal{F}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}))^\top$, *where*

$$f_k \text{ is an } \begin{cases} (n, m) - \text{OV-polynomial } [1], & \text{if } k \in [m_1] \\ (n, m, k - m_1) - \text{TOV-polynomial}, & \text{if } k \in [m_2] \setminus [m_1] \\ (n, m - m_2 + m_1 - 1) - \text{OV-polynomial}, & \text{if } k \in [m] \setminus [m_2]. \end{cases}$$

A **TUOV map** *in relation to* $\mathsf{params} = (n, m, m_1, m_2, q)$ *is* $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ *where* $\mathcal{S} : \mathbb{F}_q^m \to \mathbb{F}_q^m$ *and* $\mathcal{T} : \mathbb{F}_q^n \to \mathbb{F}_q^n$ *are invertible affine transformations and* $\mathcal{F} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ *is a TUOV central map in relation to* params. *We call* $\mathcal{P}$ **random** *if the coefficients of the polynomials* f_k *in* $\mathcal{F}$ *as well as the affine transformations* $\mathcal{S}$ *and* $\mathcal{T}$ *are chosen uniformly at random.*

Note that the inclusion of the affine transformation $\mathcal{S}$ for a TUOV map differs from the construction of UOV maps that we saw in Section 3.1. This transformation provides additional obfuscation of the special structure of the TUOV central map. This is needed because, for example, the structure of the polynomials in a TUOV central map differs depending on their index, and this order is preserved under the affine transformation $\mathcal{T}$.

Instead of having just vinegar and oil variables as in UOV, we now divide the variables $x_1, \ldots, x_n$ into three subsets. We have vinegar variables $x_1, \ldots, x_{n-m}$, *triangular* variables $x_{n-m+1}, \ldots, x_{n-m+m_2-m_1+1}$ and oil variables $x_{n-m+m_2-m_1+2}, \ldots, x_n$. The vinegar variables appear in quadratic terms with any other variables in the TOV-polynomials

[1] Note that in the TUOV specifications [22], this was incorrectly defined as an (n, m_1)-OV polynomial, however one can infer from the rest of the documentation that they meant to write (n, m).

of the TUOV central map. The triangular variables $x_{n-m+1}, \ldots, x_{n-m+m_2-m_1+1}$ appear in the additional quadratic terms of the TOV-polynomials, with only x_{n-m+1} appearing in additional terms in f_{m_1+1}, only x_{n-m+1} and x_{n-m+2} appearing in additional terms in f_{m_1+2}, and so on. Finally, the oil variables $x_{n-m+m_2-m_1+2}, \ldots, x_n$ only appear in quadratic terms with the vinegar variables in any of the polynomials in the TUOV central map (just as in UOV). This means we will be able to solve a linear system for the oil variables once the vinegar and triangular variables have been found.

Remark 3.3.3. In their implementation, the authors of [22] take $m_1 = m_2 = \frac{1}{2}m$, so there are no TOV-polynomials in their TUOV central map. This results in the first half of the polynomials in the TUOV central map being (n, m)-OV polynomials and the remaining half being $(n, m-1)$-OV polynomials, which creates just one triangular variable.

We can see that an (n, m, m_1, m_2, q) TUOV central map is very similar to a UOV central map in the following way.

Proposition 3.3.4. *An (n, m, m_1, m_2, q) TUOV central map can be seen as a collection of m $(n, m - m_2 + m_1 - 1)$-OV polynomials. Moreover, if we consider the case $m_1 = \frac{1}{2}m$, we can see it as a $(n, m-1, q)$ UOV central map with one extra polynomial.*

Proof: We will count how many zero rows must be in the matrices representing the quadratic parts of each of the three different kinds of polynomials in an (n, m, m_1, m_2, q) TUOV central map. For the first m_1 polynomials, there are m zero rows, for the next $m_2 - m_1$ polynomials, there are $m-(k-m_1)$ zero rows respectively for $k = m_1+1, \ldots, m_2$, and for the final $m-m_2$ polynomials, there are $m-m_2+m_1-1$ zero rows. We wish to know which matrices have the fewest zero rows to know what the parameters of the related UOV central map would be. First, note that $m-(k-m_1)$ for $k = m_1 + 1, \ldots, m_2$ is minimized when $k = m_2$, for which there are $m - m_2 + m_1$ zero rows.

Thus, each matrix representing the quadratic part of one of the polynomials in the (n, m, m_1, m_2, q) TUOV central map has at least m' rows of zeros where

$$\begin{aligned} m' &= \min(m, m - m_2 + m_1, m - m_2 + m_1 - 1) \\ &= m - m_2 + m_1 - 1 \text{ (since } m_1 \leq m_2) \end{aligned}$$

When $m_1 = m_2 = \frac{1}{2}m$, we have $m' = m - 1$. In this case, the (n, m, m_1, m_2, q) TUOV central map is a collection of m $(n, m-1)$-OV polynomials, which is just an $(n, m-1, q)$ UOV central map with one extra polynomial. ∎

Proposition 3.3.4 will be relevant to our security analysis as we consider the application of known UOV attacks to the TUOV digital signature scheme.

Conversely, an (n, m, q) UOV central map can be seen as a TUOV central map in the following way.

Proposition 3.3.5. *An (n, m, q) UOV central map is an (n, m_1, m_2, m, q) TUOV central map for any integers m_1, m_2 such that $1 \leq m_1 \leq m_2 < m < n$.*

Proof: Note that the matrix representing the quadratic part of any polynomial in a TUOV central map has at most m rows of zeroes (this is achieved by the first m_1 polynomials). Thus, since all of its matrices have m rows of zeroes, an (n, m, q) UOV central map can be seen as an (n, m_1, m_2, m, q) TUOV central map with any integers m_1, m_2 satisfying the required constraints. ∎

Let κ denote a security parameter. Fix Setup$(\cdot)$ that outputs $\mathsf{params} = (n, m, m_1, m_2, q)$ on input 1^κ. We will now define the hardness of the TUOV problem analogously to the hardness of the UOV problem.

Definition 3.3.6. *[22, Definition 7] The* **TUOV problem** *in relation to Setup$(\cdot)$ is* (t, ϵ)**-hard** *if there exists no algorithm that, given* params *and a random TUOV map $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ in relation to* params*, on input $\mathbf{z} = \mathcal{P}(\mathbf{w})$ with $\mathbf{w} \xleftarrow{\$} \mathbb{F}_q^n$, outputs $\mathbf{w}'$ such that $\mathcal{P}(\mathbf{w}') = \mathbf{z}$ with probability no less than $\epsilon(\kappa)$ in processing time $t(\kappa)$.*

The security of the TUOV signature scheme is conjecturally based on the hardness of the TUOV problem [22].

3.3.2 TUOV Algorithms

The TUOV signature scheme follows a very similar structure to the UOV signature scheme, but with a slightly more complex process for the Signer to find a pre-image of the TUOV central map.

Key Generation: The algorithm KeyGen takes as input $\mathsf{params} = (n, m, m_1, m_2, q)$ and ouputs the key pair (pk,sk). We have sk=$(\mathcal{S}, \mathcal{F}, \mathcal{T})$ where $\mathcal{F}$ is a random TUOV central map in relation to params, and $\mathcal{S} : \mathbb{F}_q^m \to \mathbb{F}_q^m$ and $\mathcal{T} : \mathbb{F}_q^n \to \mathbb{F}_q^n$ are random invertible affine transformations. Then pk=$\mathcal{P}$ where $\mathcal{P} := \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$.

Signing: We provide a high-level description of the signing algorithm here and give further details and the pseudocode in Appendix A.

The algorithm Sign takes as input params, sk, and a message $\mu \in \{0, 1\}^*$. The Signer first generates $\mathbf{z} = \text{Hash}(\mu) \in \mathbb{F}_q^m$ where Hash is a hash function into $\mathbb{F}_q^m$ and computes $\mathbf{y} := \mathcal{S}^{-1}(\mathbf{z})$. They now wish to find a pre-image of $\mathbf{y}$ under $\mathcal{F}$.

In order to do this, we note that, by setting $h(\mathbf{x}) := f(\mathbf{x}) - x_{n-m+1}$, any (n, m)-OV polynomial f can be written as

$$f(\mathbf{x}) = x_{n-m+1} + h(\mathbf{x})$$

where h is an (n, m)-OV polynomial.

Furthermore, any $(n, m, k - m_1)$-TOV polynomial f_k can be written as

$$f_k(\mathbf{x}) = x_{n-m+k-m_1+1} + g_k(x_{n-m+1}, \ldots, x_{n-m+k-m_1}) + h_k(\mathbf{x})$$

where g_k is some quadratic polynomial and h_k is an (n, m)-OV polynomial.

Then, we can write

$$\begin{aligned}&(f - h, f_{m_1+1} - h_{m_1+1}, \ldots, f_{m_2} - h_{m_2}) \\ =&(x_{n-m+1}, x_{n-m+2} + g_{m_1+1}(x_{n-m+1}), \ldots, \\ &x_{n-m+m_2-m_1+1} + g_{m_2}(x_{n-m+1}, \ldots, x_{n-m+m_2-m_1})),\end{aligned} \tag{3.3.1}$$

which we call a *triangular map* since it can be efficiently solved given some target vector $\mathbf{u} \in \mathbb{F}_q^{m_2-m_1+1}$ by setting $x_{n-m+1} = u_1$, $x_{n-m+2} = u_2 - g_{m_1+1}(u_1)$, and so on.

The signing algorithm takes advantage of this triangular structure by significantly modifying how it chooses to assign the $n-m$ vinegar variables; see details in Appendix A.

Once a pre-image $\mathbf{s}$ such that $\mathcal{F}(\mathbf{s}) = \mathbf{y}$ is found, the Signer returns the signature $\sigma := \mathcal{T}^{-1}(\mathbf{s})$.

Verification: The algorithm Verify takes as input params, pk, and the message/ signature pair (μ, σ). The Verifier generates $\mathbf{z} = \text{Hash}(\mu)$. They accept the signature if and only if $\mathcal{P}(\sigma) = \mathbf{z}$.

This digital signature scheme is correct for an honest Signer and Verifier since

$$\begin{aligned}\mathcal{P}(\sigma) &= (\mathcal{S} \circ \mathcal{F} \circ \mathcal{T})(\mathcal{T}^{-1}(\mathbf{s})) \\ &= (\mathcal{S} \circ \mathcal{F})(\mathbf{s}) \\ &= \mathcal{S}(\mathbf{y}) \\ &= \mathbf{z}.\end{aligned}$$

The actual algorithm used for the implementation of TUOV contains some additional optimizations to reduce the sizes of the keys, but the general structure follows the above description [22].

3.3.3 Security Analysis

In this section, we discuss the security of TUOV. We first show that Theorem 1 of [22], which showed that the MQ Problem reduces to the TUOV problem under very specific parameters, can also apply to the UOV problem, which was stated not to be the case in [22, Introduction]. Then we discuss how known attacks on UOV from Section 3.2.1 can be applied to TUOV.

3.3.3.1 On the Randomness of UOV and TUOV Maps

We first define an operation on matrices that will be used in this section. For a square matrix $\mathbf{M}$, we will define $\mathrm{UT}(\mathbf{M})$ to be the unique upper triangular matrix such that $\mathrm{UT}(\mathbf{M}) - \mathbf{M}$ is skew-symmetric. More precisely, we have

$$(\mathrm{UT}(\mathbf{M}))_{ij} = \begin{cases} \mathbf{M}_{ij} & \text{for } i = j \\ \mathbf{M}_{ij} + \mathbf{M}_{ji} & \text{for } i < j. \end{cases}$$

Note that $\mathrm{UT}(\mathbf{M})$ represents the same quadratic form as $\mathbf{M}$. This operation is useful because it provides a way for us to transform the matrix representing the quadratic part of a polynomial into an upper triangular matrix without changing the polynomial that it defines.

Recall the goal of the affine transformations used to create the UOV and TUOV public keys: mask the structure of the central map to make the public map appear like a "random" instance of the MQ problem. If we could achieve this, then the MQ problem would reduce to the UOV (respectively, TUOV) problem. Note, however, that for the range of parameters used in practice, we saw in Section 3.2.1 several attacks on UOV that suggest that it is easier to solve than the MQ problem.

We will show that when $n \in \Omega(m^2)$, one can, with high probability, achieve such a reduction from the MQ problem to the UOV problem. Our argument is based on the proof of [22, Theorem 1], which was only for TUOV (and purported not to apply to UOV). However, as we showed in Propositions 3.3.4 and 3.3.5, UOV and TUOV maps are closely related, so should satisfy similar properties.

For our analysis, we let $P(n, m, q)$ be the probability that a random system of m quadratic equations in n variables over $\mathbb{F}_q$ has a solution. We will use the heuristic, as the authors of [22] do, that this probability is almost 1 when $n > m$ (i.e., the system is underdetermined) and almost 0 when $n < m$ (i.e., the system is overdetermined).

Theorem 3.3.7. *Given Setup(·) that outputs* $\mathsf{params} = (n, m, q)$ *with* $n > \frac{1}{2}m(m+3)$ *on input* 1^κ*, if the MQ problem in relation to Setup(·) is* (t, ϵ)*-hard, then the UOV problem in relation to Setup(·) is* (t, ϵ)*-hard.*

Proof: We want to show that with high probability, for a given arbitrary (n, m, q)-MQ map $\mathcal{M}$ with $n > \frac{1}{2}m(m+3)$, there exists an instance of UOV such that $\mathcal{M}$ is the public key. This would imply that an efficient attack on UOV would lead to an efficient solution to MQ.

More precisely, we must find an invertible affine transformation $\mathcal{Q} : \mathbb{F}_q^n \to \mathbb{F}_q^n$ such that $\mathcal{F} := \mathcal{M} \circ \mathcal{Q}$ is an (n, m, q)-UOV central map. Since the only structural restrictions on a UOV central map come from the quadratic terms, it suffices to specify $\mathcal{Q}$ based on its transformation on only the quadratic part of polynomials in $\mathcal{M}$, as it can be applied on the linear terms without affecting the specifications of a UOV central map.

We consider the matrix representation of the quadratic part of the k-th polynomial f_k in $\mathcal{M}$, which we write in upper triangular form as

$$\mathbf{M}_k = \begin{bmatrix} \mathbf{M}_k^{(1)} & \mathbf{M}_k^{(2)} \\ \mathbf{0}_{m\times(n-m)} & \mathbf{M}_k^{(4)} \end{bmatrix},$$

where $\mathbf{M}_k^{(1)} \in \mathbb{F}_q^{(n-m)\times(n-m)}$ and $\mathbf{M}_k^{(4)} \in \mathbb{F}_q^{m\times m}$ are upper triangular and $\mathbf{M}_k^{(2)} \in \mathbb{F}_q^{(n-m)\times m}$.

We also consider the matrix representation of an arbitrary invertible linear transformation $\mathcal{Q} : \mathbb{F}_q^n \to \mathbb{F}_q^n$ of the form

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}^{(1)} & \mathbf{Q}^{(2)} \\ \mathbf{0}_{m\times(n-m)} & \mathbf{I}_m \end{bmatrix}.$$

Then the quadratic part of the k-th polynomial of $\mathcal{M} \circ \mathcal{Q}$ has matrix representation

$$\mathbf{Q}^\top \mathbf{M}_k \mathbf{Q} = \begin{bmatrix} \mathbf{Q}^{(1)\top}\mathbf{M}_k^{(1)}\mathbf{Q}^{(1)} & \mathbf{Q}^{(1)\top}\mathbf{M}_k^{(1)}\mathbf{Q}^{(2)} + \mathbf{Q}^{(1)\top}\mathbf{M}_k^{(2)} \\ \mathbf{Q}^{(2)\top}\mathbf{M}_k^{(1)}\mathbf{Q}^{(1)} & \mathbf{Q}^{(2)\top}\mathbf{M}_k^{(1)}\mathbf{Q}^{(2)} + \mathbf{Q}^{(2)\top}\mathbf{M}_k^{(2)} + \mathbf{M}_k^{(4)} \end{bmatrix}.$$

In [22], the authors had the incorrect expression $\mathbf{Q}^{(2)\top}\mathbf{M}_k^{(1)}\mathbf{Q}^{(1)} + \mathbf{Q}^{(2)\top}\mathbf{M}_k^{(2)} + \mathbf{M}_k^{(4)}$ in the bottom right block instead. This made it appear that the system of equations we will now discuss was linear, when it is indeed quadratic.

We first apply UT to $\mathbf{Q}^\top \mathbf{M}_k \mathbf{Q}$ to simplify the analysis, since we would like to transform it into an upper triangular matrix anyway. Since we want this to represent the k-th polynomial in a UOV central map, $\mathbf{A}_k = \mathrm{UT}(\mathbf{Q}^\top \mathbf{M}_k \mathbf{Q})$ should have the form

$$\mathbf{A}_k = \begin{bmatrix} \mathbf{A}_k^{(1)} & \mathbf{A}_k^{(2)} \\ \mathbf{0}_{m\times(n-m)} & \mathbf{0}_{m\times m} \end{bmatrix}.$$

Without loss of generality, we will fix the entries of $\mathbf{Q}^{(1)}$ and treat the entries of $\mathbf{Q}^{(2)}$, of which there are $(n-m)m$, as variables.

In order to obtain $\mathbf{A}_k$ of the required form, we must have

$$\mathrm{UT}(\mathbf{Q}^{(2)^\top}\mathbf{M}_k^{(1)}\mathbf{Q}^{(2)} + \mathbf{Q}^{(2)^\top}\mathbf{M}_k^{(2)} + \mathbf{M}_k^{(4)}) = \mathbf{0}_{m\times m}.$$

Since it is already upper triangular, this gives rise to $\frac{m(m+1)}{2}$ quadratic equations in the entries of $\mathbf{Q}^{(2)}$ for each $k \in [m]$. In order to expect that a solution for the entries of $\mathbf{Q}^{(2)}$ exists, we thus require the system to be underdetermined to ensure that $P((n-m)m, m' = \frac{m^2(m+1)}{2}, q)$ is high. This is the case when

$$(n-m)m > \frac{m^2(m+1)}{2} \implies n > \frac{m(m+3)}{2}.$$

Therefore with probability $P((n-m)m, m', q)$, we can expect that such an invertible transformation $\mathcal{Q}$ exists and that $\mathcal{M}$ could have been the public key for a UOV signature scheme. ∎

We now state the analogous result for TUOV that was presented in [22]. A similar argument of counting the number of variables and equations arising from the system to be solved for $\mathbf{Q}$ is used to prove this corollary.

Corollary 3.3.8. *[22, Theorem 1] Given Setup(·) that outputs*

$$\mathsf{params} = \left(n = \frac{1}{2}m^2, m, m_1 = \frac{1}{2}m, m_2 = \frac{3}{4}m, q\right)$$

on input 1^κ *and its restriction Setup' that outputs*

$$\mathsf{params}' = \left(n = \frac{1}{2}m^2, m, q\right),$$

if the MQ problem in relation to Setup'(·) is (t, ϵ)*-hard, then the TUOV problem in relation to Setup(·) is* (t, ϵ)*-hard.*

These reductions, while an interesting study of UOV and TUOV maps, are not cryptographically relevant to these signature schemes as presented. In practice, using $n = \frac{1}{2}m^2$ would reduce the efficiency of the signing algorithms used, as well as greatly increase the number of solutions to the public system of equations (i.e., forged signatures). The recommended parameters for TUOV (see Table 3.1) do not satisfy either $n = \frac{1}{2}m^2$ or $m_2 = \frac{3}{4}m$.

(n, m, m_1, q)	$(n-m)m$	m'	m''
(112,44,22,256)	2992	43560	42592
(160,64,32,16)	6144	133120	131072
(184,72,36,256)	8064	189216	186624
(244,96,48,256)	14208	446976	442368

Table 3.1: Comparison between number of variables $(n-m)m$ and number of equations m' as in Theorem 3.3.7, respectively m'' as in [22, Theorem 1], for reducing the MQ Problem to the UOV, respectively TUOV, Problem, given each recommended parameter set (n, m, m_1, q) from [13, 22].

In fact, let $m' = \frac{m^2(m+1)}{2}$ be the number of equations in the system to solve from Theorem 3.3.7 for the entries of $\mathbf{Q}^{(2)}$, and let m'' be the number of equations to solve in the case of TUOV (see [22, Theorem 1] for the precise expression for m''). Then Table 3.1 demonstrates that with all four sets of proposed parameters for UOV and TUOV [13, 22], $(n-m)m$, which is the number of entries of $\mathbf{Q}^{(2)}$ (and thus the number of variables in the new quadratic system to be solved), is significantly smaller than both m' and m'', suggesting that the probability that a random MQ map could be a UOV or TUOV public map at these parameter levels is very low.

3.3.3.2 A Brief Nonexistence Result

To emphasize the probabilistic nature of the arguments used to prove Corollary 3.3.8, we provide an example in which the desired transformation $\mathcal{Q}$ does not exist.

Example 3.3.9. *In this example, we demonstrate an (8,4,2)-MQ map that cannot be transformed into an (8,4,2,3,2)-TUOV map. Note that these parameters match the constraints given in Corollary 3.3.8. Consider the following first two polynomials in the MQ map:*

$$\begin{aligned} f_1(\mathbf{x}) =& x_1^2 + x_1x_2 + x_1x_4 + x_1x_5 + x_1x_6 + x_1x_7 + x_2^2 + x_2x_6 + x_2x_7 + x_2x_8 + x_3^2 \\ &+ x_3x_7 + x_3x_8 + x_4x_5 + x_4x_8 + x_5x_7 + x_6x_7 + x_7^2 + x_7x_8 \end{aligned}$$

and

$$\begin{aligned} f_2(\mathbf{x}) =& x_1^2 + x_1x_2 + x_1x_4 + x_1x_5 + x_1x_6 + x_1x_7 + x_2^2 + x_2x_6 + x_2x_7 + x_2x_8 + x_3^2 \\ &+ x_3x_7 + x_3x_8 + x_4x_5 + x_4x_8 + x_5x_7 + x_6x_7 + x_7^2 + x_7x_8 + x_8^2 + 1 \\ =& f_1(\mathbf{x}) + x_8^2 + 1. \end{aligned}$$

Then the matrices representing their quadratic parts are, respectively:

$$\mathbf{M}_1 = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

and

$$\mathbf{M}_2 = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Consider an arbitrary affine transformation $\mathcal{Q}$ as in our above discussion of Corollary 3.3.8 with top right 4×4 submatrix

$$\mathbf{Q}^{(2)} = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \\ q_5 & q_6 & q_7 & q_8 \\ q_9 & q_{10} & q_{11} & q_{12} \\ q_{13} & q_{14} & q_{15} & q_{16} \end{bmatrix}.$$

Now, it is essential to note that the UT operation leaves the diagonal of a matrix unchanged, so the bottom right entry of $\mathbf{Q}^{(2)^\top}\mathbf{M}_k\mathbf{Q}^{(2)}$ for $k = 1, 2$ is also the bottom right entry of $UT(\mathbf{Q}^{(2)^\top}\mathbf{M}_k\mathbf{Q}^{(2)})$.

If we let a denote the bottom right entry of $\mathbf{Q}^{(2)^\top}\mathbf{M}_1\mathbf{Q}^{(2)}$, then

$$a = q_4^2 + q_4q_8 + q_4q_{16} + q_8^2 + q_{12}^2 + q_8 + q_{12} + q_{16},$$

and if we let b denote the bottom right entry of $\mathbf{Q}^{(2)^\top}\mathbf{M}_2\mathbf{Q}^{(2)}$, then

$$\begin{aligned} b &= q_4^2 + q_4q_8 + q_4q_{16} + q_8^2 + q_{12}^2 + q_8 + q_{12} + q_{16} + 1 \\ &= a + 1. \end{aligned}$$

Since these two entries are unchanged under the UT operation, and we require the two matrices $UT(\mathbf{Q}^{(2)^\top}\mathbf{M}_k\mathbf{Q}^{(2)})$ for $k = 1, 2$ to have a row of zeroes in order to construct a TUOV central map, we need both a and $a+1$ to be equal to 0, which is a contradiction.

Therefore, there is no affine transformation $\mathcal{Q}$ that transforms the given MQ map into a TUOV central map.

3.3.3.3 Known UOV Attacks Applied to TUOV

We now briefly discuss the attacks on UOV from Section 3.2.1 that also apply to TUOV. There are currently no known attacks that specifically exploit the structure of TUOV beyond its similarity to UOV [22].

TUOV can of course be attacked directly by simply solving the MQ problem defined by the public TUOV map with any of the generic algorithms from Section 2.4.2 that do not take into account any special structure.

The authors of TUOV state that the complexity of applying the Kipnis-Shamir attack to their scheme is the same as that of applying it to a UOV scheme with n variables and m equations as in Equation 3.2.1.

The optimized version of the intersection attack as discussed in Section 3.2.1.2 can be applied to TUOV since $n < 2.5m$ in their recommended parameters. Since the complexity of the attack depends only on the number of equations to be solved, it is the same as the case of applying the attack to UOV with n variables and m equations. If we let k be the number of subspaces whose intersection we are studying, then the complexity of this attack is dominated by the time it takes to solve a system of $\binom{k+1}{2}m - 2\binom{k}{2}$ equations in $kn - (2k-1)m$ variables, and any algorithm for solving the MQ problem can be applied.

The authors of TUOV state that a MinRank attack applies to the scheme in the same way as it applies to UOV. The known MinRank attacks on UOV are not efficient and indeed the complexity quoted in [22, Section 4.2.4] is higher than that for the other attacks outlined above.

3.3.4 TUOV vs. UOV

We discussed in Section 3.3.1 how TUOV central maps can be seen as UOV central maps and vice versa. We saw in Section 3.3.3.1 that the security reduction which was meant to set TUOV apart can also be applied to UOV. In Section 3.3.3.3, we also saw that the complexity of attacks against TUOV are generally no different than the same attacks on UOV.

A standard UOV digital signature scheme was also submitted to the NIST standardization process in 2023 [13], and it uses the same recommended parameters for n, m, and q as the TUOV specifications. The key sizes for the two signature schemes are very similar, with TUOV having a slightly larger secret key and slightly smaller public key and signature lengths than the submission of [13]. However, in the speed of its

algorithms, TUOV far under-performs [13], with significantly slower key generation and moderately slower signing and verification, because it requires additional steps to find a pre-image of the central map.

The question of whether there exist specialized attacks that exploit the particular structure of TUOV beyond its similarity to UOV remains open.

Chapter 4

Multiparty Computation in-the-Head (MPCitH) Based Signature Schemes

In this chapter, we describe the general constructions used to develop digital signature schemes from zero-knowledge proofs, and how secure multiparty protocols can be used to achieve this. Signature schemes of this form must use some hard cryptographic problem as the basis for the multiparty protocol, and two submissions to the NIST standardization process [10, 25] use a multivariate-based problem for this. In Section 4.2 we study one of these schemes, the Biscuit digital signature scheme [10], and discuss its security based on the use of multiparty computation as well as a variant of the MQ problem. In particular, we discuss two attacks in detail, one from the specifications [10] which we prove in greater detail than the original presentation, and one from the NIST official comment process [14, 15].

4.1 Secure Multiparty Computation (MPC) Protocols and Multiparty Computation in-the-Head (MPCitH)

In this section, we define secure multiparty protocols and zero-knowledge proofs, and describe how the former can be transformed into the latter. We then describe Multi-Party Computation in-the-Head (MPCitH), a type of digital signature scheme derived from the previous ideas using a construction called the Fiat-Shamir transform.

All definitions and results in Sections 4.1.1, 4.1.2, and 4.1.3 are adapted from [33]. Throughout this section, k denotes a security parameter.

4.1.1 Secure Multiparty Protocols

We first define and study the concept of a multiparty protocol, in which multiple "players" collectively compute the output of a function. We let A and B be sets (we will often take A to be a field and $B = \{0, 1\}$), and $n \geq 2$ be an integer. We are given a function $f : A^{n+1} \to B^n$.

A *multiparty protocol* Π_f involves n players $P_1, \ldots, P_n$ who all receive (from a trusted third-party) a *public input* $x \in A$, and each additionally receive individual *local private inputs* $w_1, \ldots, w_n \in A$ and *local random inputs* $r_1, \ldots, r_n$. Each player also receives one coordinate of the output of $f(x, w_1, \ldots, w_n)$, and the parties collectively verify the correctness of the output without revealing their private inputs. *Collective verification* can mean that each party computes what they believe to be the output, or some coordinate/part of the output, and they combine the outputs of all parties to obtain the full output of the function. We sometimes take the codomain of f to be B rather than B^n, in which case we assume that all parties receive the output.

The protocol Π_f is played in rounds with parties performing computations on their local and public inputs and broadcasting messages to each other. We begin with an example to illustrate the idea.

Example 4.1.1. *Consider an example with two parties P_1 and P_2 who wish to verify that, for a triple of integers (x, y, z), we have $x \cdot y = z$, without explicitly learning the values x, y, and z.*

To accomplish this, we will require additive sharings of integers. An additive sharing *(with two parts) of an integer x is a pair $[[x]] = ([[x]]_1, [[x]]_2) \in \mathbb{Z}^2$ such that $[[x]]_1 + [[x]]_2 = x$. When we say that the parties P_1 and P_2 have this sharing, we mean that P_1 has the value $[[x]]_1$ and P_2 has the value $[[x]]_2$. It is clear that this concept generalizes beyond two parts, but we use only two in our example.*

For the protocol, P_1 and P_2 take additive sharings $[[x]], [[y]]$, and $[[z]]$, as well as additive sharings $[[a]], [[b]]$, and $[[c]]$ for integers a, b, and c such that the trusted third-party knows that $a \cdot b = c$, as their local private inputs. They both also receive the same local random input $\alpha \in \mathbb{Z}$.

In the first round, the parties locally compute (i.e., on their shares only) two additive sharings $[[\rho]] = \alpha \cdot [[x]] + [[a]]$ and $[[\sigma]] = [[y]] + [[b]]$. They then broadcast the computed shares to each other, so that they can both compute the values ρ and σ. They are able to correctly compute ρ and σ since the sum of two additive sharings of integers is an additive sharing of the sum of these two integers. Moreover, multiplying an additive sharing of an integer by a constant produces an additive sharing of that integer times the constant.

In the second round, they locally compute $[[v]] = \alpha \cdot [[z]] - [[c]] + \sigma[[a]] + \rho \cdot [[b]] - \rho \cdot \sigma$.

They then broadcast these shares to each other, so they can both compute the value v. Note that when we write $\rho \cdot \sigma$, we mean it to represent the vector $(\rho \cdot \sigma, 0)$ so that only the first party will add this term to their share and the computation will be correct.

Each party checks that $v = 0$ and outputs 1 (i.e., accepts that $x \cdot y = z$) if this is the case.

Indeed if we have $x \cdot y = z$, then:

$$\begin{aligned} v &= [[v]]_1 + [[v]]_2 \\ &= \alpha[[z]]_1 - [[c]]_1 + \sigma[[a]]_1 + \rho[[b]]_1 - \rho\sigma + \alpha[[z]]_2 - [[c]]_2 + \sigma[[a]]_2 + \rho[[b]]_2 \\ &= \alpha z - c + \sigma a + \rho b - \rho\sigma \\ &= \alpha z - c + (y+b)a + (\alpha x + a)b - (\alpha x + a)(y+b) \\ &= \alpha z - c + ay + ab + \alpha bx + ab - \alpha xy - ay - \alpha bx - ab \\ &= \alpha(z - xy) + ab - c \\ &= 0 \ \textit{(since } c = ab, z = xy\textit{)} \end{aligned}$$

This particular protocol is secure in the sense that throughout the rounds of computation and broadcasting, the values x, y, and z remain secret, but the two parties are still able to collectively verify that $x \cdot y = z$. The full analysis of this protocol and its security can be found in [43].

Formally, the set of *messages* broadcast by P_i in round $j+1$ given the public input x, its local private input w_i, local random input r_i, and the sets of messages $m_1, \ldots, m_j$ it received in the first j rounds of the protocol is denoted by

$$\Pi_f(i, x, w_i, r_i, (m_1, \ldots, m_j)).$$

Note that Π_f can depend on the sets of messages $m_1, \ldots, m_j$ as a player may use information that they received from other players as a part of their computations. Then a full protocol Π_f with m rounds is defined by the disjoint union

$$\bigcup_{i \in [n], j \in [m]} \Pi_f(i, x, w_i, r_i, (m_1, \ldots, m_j)).$$

Let $\mathrm{In}_k(P_i)$ be the set of all incoming messages during the protocol to party P_i from party P_k. Then if we let $m_j^{(k)}$ be the element of m_j that was sent by party P_k, we have

$$\mathrm{In}_k(P_i) = \bigcup_{j \in [m]} m_j^{(k)}.$$

Note that we consider messages sent during the protocol to be "broadcast" so that all parties can see them. This means that we have $\mathrm{In}_k(P_i) = \mathrm{In}_k(P_j)$ for all $i \neq j$.

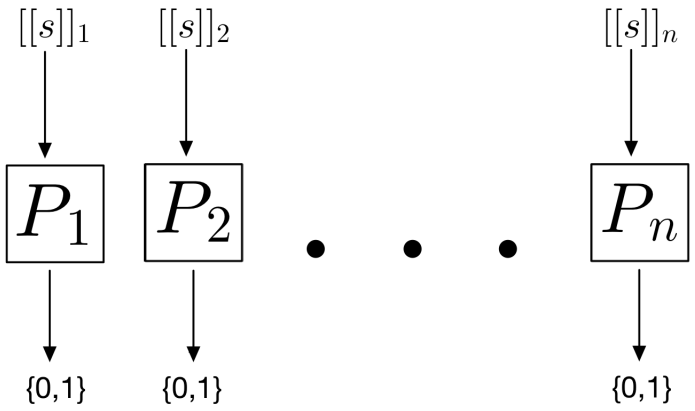

Figure 4.1: An MPC protocol given a sharing of a secret s, executed by n parties.

Let $\mathrm{Out}(P_i)$ be the set of all outgoing messages that should be sent from P_i during the protocol. So

$$\mathrm{Out}(P_i) = \bigcup_{j\in[m]} \Pi_f(i, x, w_i, r_i, (m_1, \ldots, m_j)).$$

We denote by

$$V_i = \Big(w_i, r_i, \bigcup_{k\in[n];k\neq i} \mathrm{In}_k(P_i), \mathrm{Out}(P_i)\Big)$$

the *view* of player P_i. This captures the input to P_i as well as all messages they send and receive throughout the protocol.

Figure 4.1 illustrates an MPC protocol with private inputs which are additive sharings of some value s and outputs in $\{0, 1\}$.

We now define several important concepts related to the correctness and security of MPC protocols.

Definition 4.1.2. *[33, Definition 2.2] A pair of views V_i, V_j are* **consistent** *(with respect to the protocol Π_f and some public input x) if $In_j(P_i) = Out(P_j)$ and $In_i(P_j) = Out(P_i)$.*

We will consider our protocols to be taking place in the *malicious model*, where corrupted players may not follow the protocol, so the broadcast messages of P_i may not correctly follow the definition of $\mathrm{Out}(P_i)$. Note that an honest execution of the

protocol can be reconstructed from a set of views $V_1, \ldots, V_n$ if and only if these views are all pairwise consistent. More intuitively, if there is some pair of inconsistent views, then we know that the protocol must not have been executed honestly, since there is some party P_i whose broadcast messages do not match $\text{Out}(P_i)$.

We now define the notion of *robustness* of an MPC protocol in the malicious model.

As previously noted, we often take the codomain of f to be $\{0, 1\}$, in which case, the goal of the protocol is to verify that $f(x, w_1, \ldots, w_n) = 1$. We can often see the output of f to be the truth value of some statement that the parties wish to collectively verify (i.e., the output is 1 if the statement is true and 0 if the statement is false). Thus, an individual player outputs 1 at the end of the protocol if they are convinced that the statement is true. If all players output 1 (i.e., all players individually accept that $f(x, w_1, \ldots, w_n) = 1$), then we say that the protocol *accepts*.

Definition 4.1.3. *[33, Definition 2.6] The protocol Π_f realizes f with* **perfect** *(respectively,* **statistical***)* **t-robustness** *if for any computationally unbounded malicious adversary corrupting a set T of at most t players, and for any inputs $(x, w_1, \ldots, w_n)$, the following holds. If there is no $(w'_1, \ldots, w'_n)$ such that $f(x, w'_1, \ldots, w'_n) = 1$, then the probability that some uncorrupted player outputs 1 in an execution of Π_f in which the inputs of the honest players are consistent with $(x, w_1, \ldots, w_n)$ is 0 (respectively, is negligible in k). Moreover, if there are no corrupted players, then the probability that the output of some player is different from $f(x, w_1, \ldots, w_n)$ is 0 (respectively, negligible in k).*

Intuitively, t-robustness means that if t parties are behaving maliciously (i.e., potentially not following the protocol) in the MPC protocol, then it is very unlikely that the protocol will accept in the end.

The probability that the parties will accept the protocol on public input x and local inputs $w_1, \ldots, w_n$ when $f(x, w_1, \ldots, w_n) = 0$ (i.e., the statement is false) is called the *false positivity rate* of the MPC protocol.

We finally define the notion of privacy of an MPC protocol. In the following definition, $f_T(x, w_1, \ldots, w_n)$ refers to the n-tuple of outputs of all n parties in the protocol given that the parties corresponding to an index set T are corrupted. The notion of t-privacy ensures that a set of t corrupted players cannot gain additional information about the private inputs of the honest players that they could not gain if they were following the protocol honestly. This ensures that the secret value shared among the parties remains secret during the protocol.

Definition 4.1.4. *[33, Definition 2.5] Let $1 \leq t \leq n$. The protocol Π_f realizes f with* **perfect t-privacy** *if there is a PPT simulator* SIM *such that for any inputs $(x, w_1, \ldots, w_n)$ and every set of corrupted players $T \subseteq [n]$ where $|T| \leq t$, the joint view $\text{View}_T(k, x, w_1, \ldots, w_n) = \bigcup_{i \in T} V_i$ of players in T is distributed identically (as a*

distribution in the random local inputs of the parties) to $\mathrm{SIM}(k, T, x, (w_i)_{i\in T}, f_T(x, w_1, \ldots, w_n))$. *For* **statistical** *(respectively,* **computational***)* t**-privacy**, *we require that for every distinguisher* D, *as defined in Definition 2.2.13, (respectively,* D *with circuit size polynomial in* k*) there is a negligible function* $\delta(\cdot)$ *such that*

$$\begin{aligned}&|\Pr[D(\mathrm{View}_T(k, x, w_1, \ldots, w_n)) = 1]\\ &- \Pr[D(\mathrm{SIM}(k, T, x, (w_i)_{i\in T}, f_T(x, w_1 \ldots, w_n))) = 1]| \leq \delta(k).\end{aligned}$$

4.1.2 Zero-Knowledge Proofs

Zero-knowledge proofs are a key building block in theoretical computer science, and are becoming a popular tool for creating post-quantum digital signature schemes. We provide the general definition here, which we will later apply to the context of multiparty protocols and digital signature schemes in Sections 4.1.3 and 4.1.4.

An *NP-relation* is an efficiently decidable binary relation $R(x, w)$ which is polynomially bounded (i.e., there exists a polynomial p such that if $R(x, w) = 1$, then $|w| \leq p(|x|)$, where $|w|$ denotes the *length* of w, usually measured in bits or bytes) such that given x only, computing w such that $R(x, w) = 1$ is not efficient. We will use X to denote the domain of x, which is the set of possible *statements* and W to denote the domain of w, which is the set of possible *witnesses*. The actual sets X and W can vary depending on the context, but X will usually be a set of mathematical propositions and W will be a set of elements about which these propositions can be stated (e.g., W could be a finite field and X could be a set of statements about the possible properties of elements in the field).

A *zero-knowledge proof protocol* of such a relation is a pair (P, V) of interactive PPT algorithms (denoting the Prover and Verifier) where P is given an NP statement x and a corresponding witness w, and V is given only x. We are concerned with NP-relations because we would not like for the Verifier V to be able to efficiently compute a witness w from the statement x (since otherwise, V can compute their own witness and does not require any proof from P). The role of P in a zero-knowledge proof is to ensure that V cannot learn information about w through their interactions. The Verifier V should *accept* the protocol if $R(x, w) = 1$, and should reject if there is no w satisfying this relation. Throughout their interactions, the Verifier provides the Prover with *challenges* that may be random values the Prover must use in their algorithms or requests for specific information. The Prover must answer these challenges with the required information in order to convince the Verifier that they possess a witness.

Example 4.1.5. *We could consider an NP-relation defined using a one-way function* g *in which the statement* x *would be "*$g(a) = b$*" where* V *knows* g *and* b*, but only* P *knows the pre-image* $w = a$*, which is a witness to the relation. In this case, since* $g(a) = b$ *is true, we would have* $R(x, w) = 1$. *In a zero-knowledge proof protocol for*

this relation, P would somehow prove through interactions with V that they have a value a such that $g(a) = b$ and V would accept if and only if they are convinced that this is the case.

Let x be an NP-statement, w be the witness possessed by the Prover, and r be some random input that allows the algorithm V to include randomness. We also denote by $\text{In}_P(V)$ the set of all messages that V has received from P during their interaction. Then the *view* of the Verifier V is the tuple $\text{View}_V(x, w) = (x, r, \text{In}_P(V))$. This is very similar to the concept of views used in multiparty protocols. For the proof protocol to be a zero-knowledge proof, it must then satisfy the following definition.

Definition 4.1.6. *[33, Definition 2.1] The protocol (P, V) is a* **zero-knowledge proof protocol** *for the relation R if for all $x \in X$ and $w \in W$, it satisfies the following:*

- **Completeness.** *If $R(x, w) = 1$ and both players are honest, the Verifier always accepts.*

- **Soundness.** *For every malicious and computationally unbounded Prover P^*, there is a negligible function $\epsilon(\cdot)$ such that if x is a false statement (i.e., $R(x, w) = 0$ for all w), the interaction of P^* with V on input x makes V reject except with at most probability $\epsilon(|x|)$. We call $1/\epsilon(|x|)$ the* soundness error *of the zero-knowledge proof.*

- **Zero-Knowledge.** *For any malicious PPT Verifier V^* (i.e., a Verifier that wants to learn the witness w through their interactions with P) there is a PPT simulator M^* such that the view of V^*, when interacting with P on inputs (x, w) for which $R(x, w) = 1$, is computationally indistinguishable from the output distribution of $M^*(x)$. That is, there exists a negligible function $\delta(\cdot)$ such that for every efficient non-uniform distinguisher D and every $(x, w) \in R$, we have*

$$|\Pr[D(\text{View}_{V^*}(x, w)) = 1] - \Pr[D(M^*(x)) = 1]| \leq \delta(|x|).$$

Intuitively, soundness ensures that the Verifier will only accept a proof from a malicious Prover (i.e., one that does not have the secret) with negligible probability. The zero-knowledge property tells us that a Verifier cannot gain any non-negligible amount of information about the secret from an honest Prover during their interaction, even if they behave with some sort of strategic maliciousness.

4.1.3 From Secure MPC to Zero-Knowledge

We now provide the construction to transform a secure MPC protocol into a zero-knowledge proof in the malicious model, which was originally presented in [33]. This construction works by having one person simulate an entire MPC protocol.

This construction requires the Prover to be able to *commit* to the value of some string during the protocol. To achieve this, we will define the **commitment-hybrid** model to be a protocol in which a string can be committed to by secretly sending it to a trusted third party, and later *opened* by having the committer instruct this party to reveal the committed string to the Verifier. This process must obey the properties of any commitment protocol, meaning that it is both binding (the string cannot be changed after being committed to) and concealing (the string cannot be seen by other parties until it is opened by the trusted third party) [17].

Let R be an NP-relation with witnesses in $\mathbb{Z}_2^m$ for some positive integer m. Let $w_1, \ldots, w_n \in \mathbb{Z}_2^m$. Then we define the function f to be computed by the MPC protocol as

$$f(x, w_1, \ldots, w_n) = R(x, w_1 \oplus \ldots \oplus w_n) \in \{0, 1\}.$$

Let Π_f be an n-party protocol for computing the function f and x be an NP-statement for the relation R. Let $t \geq 1$ be an integer. We define the following proof protocol based on Π_f.

Zero-Knowledge Protocol $\Pi_{R,t}$ in the Malicious Model

1. The Prover picks a witness $w \in \mathbb{Z}_2^m$ such that $R(x, w) = 1$ and chooses $w_1, \ldots, w_{n-1} \in \mathbb{Z}_2^m$ at random. They set $w_n = w \oplus \bigoplus_{i=1}^{n-1} w_i$ so that $(w_1, \ldots, w_n)$ is an additive sharing of w. They then emulate "in their head" the execution of Π_f on input $(x, w_1, \ldots, w_n)$. This involves choosing uniform random inputs $r_1, \ldots, r_n$ for each of the n players and running the protocol. Based on this execution, the Prover prepares the views $V_1, \ldots, V_n$ and separately commits to each one.

2. The Verifier picks at random t distinct player indices $i_1, \ldots, i_t \in [n]$ and sends them to the Prover.

3. The Prover opens the commitments corresponding to the t views $V_{i_1}, \ldots, V_{i_t}$.

4. The Verifier accepts if and only if:
   (a) The Prover indeed opened the requested views,

   (b) The outputs of all players in these views are 1, and

   (c) The t opened views are pairwise consistent with each other with respect to Π_f and x.

For the above proof protocol to be zero-knowledge, we will require the MPC protocol Π_f to realize f with perfect t-robustness and perfect or statistical t-privacy. Under

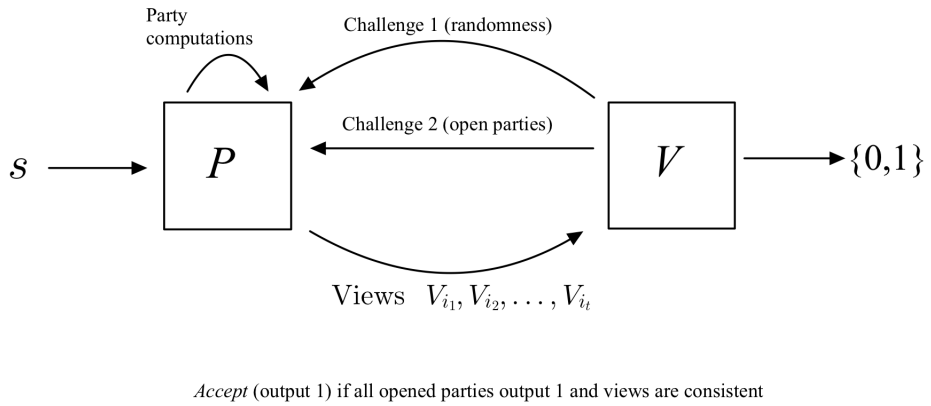

Figure 4.2: A zero-knowledge proof with Prover P and Verifier V based on an MPC protocol.

the assumption that Π_f is t-robust, the Prover must open the views of t parties to the Verifier in Step 3 in order to convince them that they are acting non-maliciously. This is because, by definition, if there are at most t corrupted parties who are not following the protocol, then the protocol will not accept (except maybe with negligible probability), and the Verifier will be able to identify malicious behaviour in Step 4. For applications, t is often as large as $n-1$ to account for a possibly very malicious Prover (and this is the maximum value for t allowed). To further increase the probability of detecting a corrupted player, the protocol may be repeated several times.

Example 4.1.7. *If we consider the MPC protocol from Example 4.1.1 for verifying multiplicative triples, then Step 1 of the above zero-knowledge protocol consists of the Prover executing the whole MPC protocol for P_1 and P_2 and committing to their views. Then in Step 2, the Verifier will request the views of just one of the parties P_{i_1} at random. Then, the Prover will open the view of P_{i_1} and the Verifier will accept if and only if this view is indeed opened, the output of P_{i_1} is 1, and the view of P_{i_1} is consistent with an honest execution of the MPC protocol.*

Figure 4.2 illustrates how a zero-knowledge proof based on an MPC protocol is executed, where s denotes the witness possessed by the Prover. Note that Challenge 1 in this figure, where the Verifier provides some random values to the Prover, is sometimes replaced by the Prover generating the random input to the parties themselves, as is the case in our protocol $\Pi_{R,t}$.

Theorem 4.1.8. *[33, Theorem 4.1]Suppose that Π_f realizes the n-party function f with perfect t-robustness and perfect or statistical t-privacy, where $t = \Omega(k)$ is a constant multiple of the soundness parameter k, and $n = ct$ for some $c > 1$. Then $\Pi_{R,t}$ is a zero-knowledge proof protocol for R in the commitment-hybrid model, with*

soundness error $2^{\Omega(k)}$.

Proof: We must show that $\Pi_{R,t}$ satisfies the three conditions of Definition 4.1.6.

Completeness: If $(x, w) \in R$ and the Prover is honest, then since $w = w_1 \oplus \ldots \oplus w_n$ and Π_f is perfectly t-robust, the parties $P_1, \ldots, P_n$ always have output 1. Since the views come from an honest execution of Π_f, they must be pairwise consistent. Thus, the Verifier will accept in this case.

Soundness: The proof of soundness reduces to computing the probability that the Verifier will have chosen a set of consistent views when in fact not all are consistent. This is quite technical in one case, so we omit the full details of the proof from [33, Theorem 4.1], but provide the intuition. Let $V_1, \ldots, V_n$ be the views committed to by the Prover. Suppose x is a false statement, i.e., $R(x, w) = 0$ for all $w \in \{0,1\}^m$.

First suppose that there are at least $n - t$ consistent views. Then, the protocol could be realized by the Prover corrupting at most t players. Then, by the perfect t-robustness of Π_f, since $R(x, w) = 0$ for any w, all of the honest players, of which there are at least $n - t$, must output 0. The only way that the Prover will not open one of these parties to the Verifier is if there are exactly t corrupted players, and it is this set of t parties whose views are revealed to the Verifier. Thus, the probability that the Prover will not open any of the views of these honest players to the Verifier is less than or equal to $\frac{1}{\binom{n}{t}}$. Moreover,

$$\frac{1}{\binom{n}{t}} \leq \frac{1}{(\frac{n}{t})^t} = \frac{1}{c^t}$$

since $t = \frac{n}{c}$. So, except with this negligible probability, the Prover must open a view of a party whose output it 0, and therefore the Verifier will reject the proof.

Otherwise, assume that there are less than $n - t$ consistent views. The authors of [33] use a clever argument in graph theory involving counting the edges of a matching in a particular graph to prove that the probability that the Verifier chooses a set of consistent views is again negligible; see [33, Theorem 4.1].

Zero-Knowledge: If we assume that Π_f is perfectly t-private, then there exists a PPT simulator SIM such that for any inputs $x, w_1, \ldots, w_n$ and every set of corrupted players $T \subseteq [n]$ where $|T| \leq t$, the joint view $\text{View}_T(k, x, w_1, \ldots, w_n) = \bigcup_{i \in T} V_i$ of players in T is distributed identically to $\text{SIM}(k, T, x, (w_i)_{i \in T}, f_T(x, w_1, \ldots, w_n))$.

Let V^* be a malicious Verifier. We describe a simulator M^* for the view of V^* which invokes both V^* and the MPC simulator SIM, as defined in Definition 4.1.4, as a black box. The simulator M^* does the following:

1. Run V^* on input x. Let $i_1, \ldots, i_t$ be the indices requested by V^* in Step 2 of

the protocol.

2. Simulate the views $V_{i_1}, \ldots, V_{i_t}$ received from the (honest) Prover in Step 3 of the protocol by picking $w_{i_1}, \ldots, w_{i_t}$ randomly and running

$$\mathrm{SIM}(k, T = i_1, \ldots, i_t, x, (w_{i_1}, \ldots, w_{i_t}), 1).$$

We now show that the simulation M^* is perfect. First fix (x, w) such that $R(x, w) = 1$. Let r_{V^*} denote the randomness of the algorithm V^* that determines the distribution from which V^* chooses the random values $i_1, \ldots, i_t$. Then r_{V^*} is distributed identically whether it is considered in the actual execution of Π_R or is called by the simulator. Hence it suffices to show that M^* is perfect given any choice of r_{V^*}. Let $i_1, \ldots, i_t$ be the Verifier's selected indices determined by r_{V^*}. Then $w_{i_1}, \ldots, w_{i_t}$ chosen by the simulator M^* are uniform and independent, since this is the case in the honest execution as well. Now, conditioned on this choice of $r_{V^*}, w_{i_1}, \ldots, w_{i_t}$, since SIM is a perfect t-private simulator for Π_f, the views produced by M^* are identically distributed to those received from an honest prover in Π_R. Thus, the simulation M^* is indeed perfect.

Finally, if Π_f is only statistically t-private, then we know that the views produced by SIM are only statistically close to the output of an honest Prover. This in turn means that the output of M^* is only statistically close to the view of V^*, as desired. ∎

Zero-knowledge proofs derived from secure MPC protocols are widely used in cryptography [3, 7, 10, 33, 64], and the security of these protocols usually relies on a variant of some well-studied cryptographic primitive believed to be difficult to solve. This hard cryptographic problem becomes the basis for an NP-relation, in which witnesses are the solution to the problem, which defines a function f for an $(n-1)$-robust MPC protocol, from which we can construct a zero-knowledge proof as demonstrated above.

Note that in this case, $t = n - 1$, which does not satisfy the hypothesis of Theorem 4.1.8. In fact, the probability $\frac{1}{\binom{n}{t}}$ from the soundness proof above is not negligible, since it is equal to $\frac{1}{n}$. To ensure that the Prover cannot cheat by corrupting $n-1$ parties in this case, they will be required to repeat the protocol several times with new initial input each time. We discuss this repetition of the protocol further in the next section.

In applications, the inputs to the MPC protocol will often be in a general finite field $\mathbb{F}_q$, rather than just the binary field and the parties will possess an *additive sharing*

of the witness $w \in \mathbb{F}_q$. We define such a sharing as a vector in $\mathbb{F}_q^n$,

$$[[w]] = ([[w]]_1, \ldots, [[w]]_n),$$

such that

$$w = \sum_{i=1}^{n} [[w]]_i.$$

The secret input to party P_i is thus the i-th coordinate of this vector, $[[w]]_i$. Note that if we take any other $v \in \mathbb{F}_q$, the vector $[[w]] + [[v]]$ is an additive sharing of $w + v$ and the vector $v \cdot [[w]]$ is an additive sharing of vw. In the case that the MPC requires the parties to collectively compute a sharing of $w + v$ where only the whole value of v, rather than a sharing of it, is known, the convention is to have party P_1 add v to $[[w]]_1$ and for the rest of the parties to leave their coordinates of $[[w]]$ as they are.

4.1.4 Multiparty Computation in-the-Head

The use of Multi-Party Computation to develop signature schemes has become popular since the 2017 submission of Picnic [64] to the NIST standardization process. In order to transform the zero-knowledge proof constructed from an $(n-1)$-robust MPC protocol into a signature scheme, we must remove the interaction between the Prover and Verifier so that the Signer can execute the proof "in their head". We will use a tool called the *Fiat-Shamir Transform* to accomplish this, and the resulting non-interactive protocol is called *Multi-Party Computation in the Head* (MPCitH).

The *Fiat-Shamir Transform* is a general heuristic which allows for zero-knowledge proofs to be transformed into secure digital signature schemes. It was originally presented in [27] using a scheme based on the hardness of factoring, which is not secure in the post-quantum context, but the techniques can be used in conjunction with any "hard" cryptographic primitive. We base our description on the one given in [3].

We will denote by V_{ZK} the Verifier in the zero-knowledge proof protocol and by V_{sig} the Verifier in the digital signature scheme. Given an NP-relation R, the public key will be a statement x and the private key will be a corresponding witness w. The Signer will play the role of the Prover in a zero-knowledge proof protocol and will pseudo-randomly generate the challenges that they would have received from V_{ZK} in order to remove the interactive aspects of the proof protocol.

The Signer begins as the Prover would in a zero-knowledge proof, by committing to the initial auxiliary information. For MPCitH, this initial information consists of the initial views of the parties, which include the public input x and local private inputs $w_1, \ldots, w_n$. This information is generated using a random salt. They compute a first hash value h_1 by using these commitments as input to a hash function.

They then use an XOF (Extendable Output Function) evaluated on h_1 to pseudo-randomly generate the first challenge of the zero-knowledge proof protocol, which would have been received from V_{ZK} in the interactive context. This challenge usually consists of some random values that the Signer will have to use during their computations. For MPCitH, this consists of the local random inputs to the parties $r_1, \ldots, r_n$.

The Signer then carries out the computations required for the zero-knowledge proof, and uses the information computed in this step as input to a hash function to generate a second hash value h_2. This hash value will also be used as input to an XOF to generate the second challenge for the zero-knowledge proof protocol, which would have usually been sent by V_{ZK} in the interactive context. This usually consists of a set of indices which indicates what information should be revealed to V_{ZK} in the final response of the proof protocol. For MPCitH, this is the subset of parties whose views will be revealed to V_{ZK}.

The signature then consists of the random salt which was used to generate the initial information, the two hash values h_1 and h_2 and the final response that would be sent to V_{ZK} in the zero-knowledge proof. This allows V_{sig} to reconstruct some parts of the Signer's proof and check that they get the same hash values h_1 and h_2 when they assume that the proof is consistent and correct. From the salt, the Verifier can recompute the initial information for all revealed parties, and from this information they can run the MPC protocol for these parties. They then recompute h_1 and h_2 by determining what the output of the hidden parties *should* have been if all views were consistent to check if they get the same values as the Signer sent them. Since the Signer must compute h_2 before they discover the subset of views that will be revealed, a malicious Signer cannot change their strategy to guarantee that those revealed views are consistent. Thus, the probability that a Verifier accepts an invalid signature is negligible.

Figure 4.3 illustrates how a signature scheme derived from a zero-knowledge proof based on an MPC protocol works using the Fiat-Shamir transform. In this figure, s denotes the secret key of the Signer.

To further guarantee the security of an MPCitH digital signature scheme, the Signer will repeat the zero-knowledge proof according to the Fiat-Shamir transform τ times, with different random input for each iteration.

4.1.5 Kales-Zaverucha Forgery Attack on MPCitH

In the case of an MPCitH-based signature scheme for an n-party protocol, the well-known Kales-Zaverucha forgery attack [34] is generally considered the fastest way to forge a signature. For this attack, the adversary chooses an integer partition $\tau = \tau_1 + \tau_2$

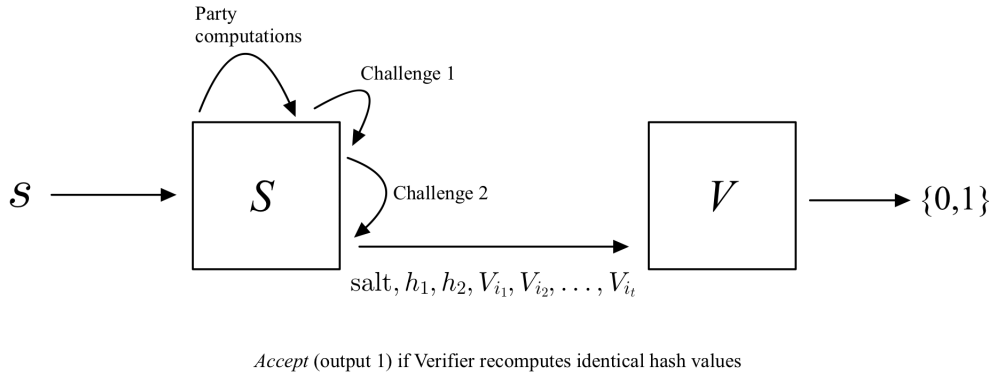

Figure 4.3: A signature scheme derived from the MPCitH construction using the Fiat-Shamir transform with Signer S and Verifier V.

of the number of times the Signer repeats the MPC protocol and a "witness" w that does not really satisfy the relation, and proceeds as follows [15]:

1. Choose a random salt and run the signing algorithm as an honest Signer would up until the second challenge would be generated (i.e., commit to auxiliary information, generate the first hash value h_1, generate the first challenge, and perform the MPC protocol).

2. Set $R := \{1 \leq i \leq \tau |$ all n parties accept w in the i-th iteration of the MPC protocol$\}$

3. If $|R| < \tau_1$, return to Step 1. Here, the attacker wishes to ensure that their fake witness is accepted in at least τ_1 iterations of the MPC protocol.

4. Randomly guess the second challenge, i.e., the index of the single party whose view will not be revealed to the Verifier, for all rounds *not* in R.

5. Produce a forgery by running the rest of the signing algorithm as an honest Signer would for all rounds in R, and using the guess of the second challenge for all rounds *not* in R to ensure all inconsistencies occur in the concealed party's computations.

6. If this is not successful, return to Step 4.

The complexity of this attack relies on choosing τ_1 and τ_2 that minimize the number

of steps required. Essentially, the attacker is guessing the first challenge for τ_1 of the iterations and guessing the second challenge for τ_2 of the iterations. Let p_1 be the false positivity rate of the MPC protocol and $p_2 = \frac{1}{n}$. Then the Kales-Zaverucha attack requires

$$\mathrm{KZ}_\tau(p_1, p_2) = \min_{\{\tau_1,\tau_2|\tau_1+\tau_2=\tau\}} \left\{ \frac{1}{\sum_{i=\tau_1}^{\tau} \binom{\tau}{i}(1-p_1)^{\tau-i}} + \frac{1}{p_2^{\tau_2}} \right\} \tag{4.1.1}$$

calls to hash functions.

4.2 NIST Submission: Biscuit

One of the proposed post-quantum digital signature schemes that we studied was Biscuit, which was submitted to the NIST standardization process in June 2023 [10]. Biscuit uses the MPCitH paradigm to construct a signature scheme based on an underlying hard problem that is a structured variant of the MQ problem. Note that since their NIST submission, the authors of Biscuit have also posted a pre-print that makes some small changes and addresses an attack that was discovered [9], but we base this section on the scheme as it was submitted to NIST.

We state the underlying hard problem of Biscuit, called PowAff2(q, m, n), in Section 4.2.1 and provide the secure MPC protocol based on it in Section 4.2.2. We then provide a high-level description of the Biscuit algorithms in Sections 4.2.3 and 4.2.4, with the full pseudocode algorithms located in Appendix B. We go through the security analysis of Biscuit in Section 4.2.5. We first provide the analysis presented in [10], expanding on the proof of one of the key theorems. We then also outline an attack that was posted to the NIST forum for official comments in 2023 [14] and later in a pre-print [15]. In Section 4.2.6, we discuss some other interesting algebraic properties of the polynomials used in Biscuit that were observed in [15]. Finally, we discuss our conclusions in Section 4.2.7.

4.2.1 Underlying Problem: PowAff2(q, m, n)

The hard problem underlying the Biscuit signature scheme is one that the authors call PowAff2. It is a structured variant of the MQ problem defined as follows.

Definition 4.2.1. *Given* $\mathbf{t} = (t_1, \ldots, t_m) \in \mathbb{F}_q^m$ *and affine forms* $A_{k,j}(x_1, \ldots, x_n) \in \mathbb{F}_q[x_1, \ldots, x_n]$ *with* $k \in [m]$ *and* $0 \leq j \leq 2$*, i.e.,*

$$A_{k,j} = a_0^{(k,j)} + \sum_{i=1}^{n} a_i^{(k,j)} x_i \text{ with } a_0^{(k,j)}, \ldots, a_n^{(k,j)} \in \mathbb{F}_q,$$

define $f_k = A_{k,0} + A_{k,1}A_{k,2}$. *The* **PowAff2($q$,$m$,$n$) problem** *asks to find (if any) a vector* $(s_1, \ldots, s_n) \in \mathbb{F}_q^n$ *such that*

$$f_k(s_1, \ldots, s_n) = t_k \text{ for all } k \in [m].$$

Since this is an instance of the MQ problem that uses special structured polynomials, the authors must claim that it is as hard to solve as generic instances of the MQ problem in order to use the standard MQ hardness assumptions discussed in Section 2.4. They formally prove that this is the case for known algorithms when $m \leq n$, but the case of $m > n$, which is actually used in implementation, can only be conjectured. This also leaves the possibility that some specialized attacks may be able to exploit the structure of these polynomials to solve the problem more efficiently. We highlight two such algorithms in Section 4.2.5.1. Despite the slight loss of security, the benefit of the structure of the equations in the PowAff2(q, m, n) problem is that they are more efficient to evaluate than generic MQ equations, since we need only evaluate three affine parts and then take their product and sum.

4.2.2 MPC Protocol for PowAff2(q, m, n)

We now present the MPC protocol used in Biscuit. It verifies a solution $\mathbf{s} \in \mathbb{F}_q^n$ of the PowAff2(q, m, n) problem, given $f_1, \ldots, f_m \in \mathbb{F}_q[x_1, \ldots, x_n]$ and a target vector $\mathbf{t} \in \mathbb{F}_q^m$ as in Definition 4.2.1. Given the shared secret $\mathbf{s}$, the protocol outputs **accept** if the parties are convinced that $\mathbf{t} = \mathbf{f}(\mathbf{s})$, otherwise it outputs **reject**.

For the following protocol, all parties know the public input consisting of $\mathbf{t} \in \mathbb{F}_q^m$ and the linear polynomials $A_{k,0}, A_{k,1}, A_{k,2} \in \mathbb{F}_q[x_1, \ldots, x_n]$ for $k \in [m]$ such that $\mathbf{f} = (f_1, \ldots, f_m)$, and $f_k = A_{k,0} + A_{k,1}A_{k,2}$. The i-th party also has as local private input the additive share $[[\mathbf{s}]]_i \in \mathbb{F}_q^n$ and as local random input the additive shares $[[\mathbf{a}]]_i \in \mathbb{F}_q^m$ where $\mathbf{a} = (a_1, \ldots, a_m) \xleftarrow{\$} \mathbb{F}_q^m$ and $[[\mathbf{c}]]_i \in \mathbb{F}_q^m$ where $\mathbf{c} = (c_1, \ldots, c_m)$ is given by $c_k = A_{k,2}(\mathbf{s}) \cdot a_k$. The protocol proceeds as follows.

For $k \in [m]$:

1. The parties locally compute $[[z_k]] \leftarrow t_k - A_{k,0}([[\mathbf{s}]])$, $[[x_k]] \leftarrow A_{k,1}([[\mathbf{s}]])$, and $[[y_k]] \leftarrow A_{k,2}([[\mathbf{s}]])$.

2. The parties get a random element $\varepsilon_k \xleftarrow{\$} \mathbb{F}_q$.

3. The parties locally set $[[\alpha_k]] \leftarrow ([[x_k]] \cdot \varepsilon_k + [[a_k]])$.

4. The parties open $[[\alpha_k]]$ so that they all know α_k.

5. The parties locally compute $[[v_k]] = [[y_k]] \cdot \alpha_k - [[z_k]] \cdot \varepsilon_k - [[c_k]]$.

6. The parties open $[[v_k]]$ to obtain v_k.

The parties output **accept** if $v_k = 0$ for all $k = 1, \ldots, m$ and **reject** otherwise.

This protocol is correct, as if the N parties have a true sharing of $\mathbf{s} \in \mathbb{F}_q^n$ such that $\mathbf{f}(\mathbf{s}) = \mathbf{t}$, they will indeed have, for all $k \in [m]$:

$$\begin{aligned} v_k &= y_k \cdot \alpha_k - z_k \cdot \varepsilon_k - c_k \\ &= A_{k,2}(\mathbf{s}) \cdot (x_k \cdot \varepsilon_k + a_k) - t_k \cdot \varepsilon_k + A_{k,0}(\mathbf{s}) \cdot \varepsilon_k - A_{k,2}(\mathbf{s}) \cdot a_k \\ &= A_{k,2}(\mathbf{s}) A_{k,1}(\mathbf{s}) \cdot \varepsilon_k - t_k \cdot \varepsilon_k + A_{k,0}(\mathbf{s}) \cdot \varepsilon_k \\ &= f_k(\mathbf{s}) \cdot \varepsilon_k - t_k \cdot \varepsilon_k \\ &= 0 \end{aligned}$$

and the protocol will output **accept**.

The false positivity rate of this MPC protocol is $\frac{1}{q^m}$, since if $\mathbf{s}$ is not a solution to the system of equations, we assume that v_k is a uniformly random element of $\mathbb{F}_q$ for $k \in [m]$. If the parties happen to have a solution to $m - u$ of the equations in the system, then the false positivity rate is $\frac{1}{q^u}$ (as the false probability rate of the MPC is $\frac{1}{q}$ for each of the u equations for which they do not have a solution).

The privacy of the protocol comes from the fact that the secret shares belonging to each party are masked with random elements before α_k is opened, so no information is revealed to the other parties. More precisely, α_k reveals no information about x_k because we multiply x_k by uniformly random $\varepsilon_k \in \mathbb{F}_q$ and add uniformly random $a_k \in \mathbb{F}_q$. The individual parties only know their share of a_k and not its full value, so they cannot deduce the value x_k.

Moreover, even revealing $N - 1$ of the parties' shares of the secret to the Verifer does not reveal any additional information to them. Without loss of generality, suppose the Verifier knows the shares for parties $2, \ldots, N - 1$ and the first party's share is hidden from them. Then, for all $k \in [m]$, they know that

$$\begin{aligned} t_k &= f_k(\mathbf{s}) \\ &= f_k([[\mathbf{s}]]_1 + \ldots + [[\mathbf{s}]]_N) \\ &= a_0^{(k,0)} + \sum_{i=1}^{n} a_i^{(k,0)}([[s_i]]_2 + \ldots + [[s_i]]_N) + \sum_{i=1}^{n} a_i^{(k,0)}[[s_i]]_1 \\ &+ \left(a_0^{(k,1)} + \sum_{i=1}^{n} a_i^{(k,1)}([[s_i]]_2 + \ldots + [[s_i]]_N) + \sum_{i=1}^{n} a_i^{(k,1)}[[s_i]]_1\right) \end{aligned}$$

$$\cdot \left(a_0^{(k,2)} + \sum_{i=1}^{n} a_i^{(k,2)} ([[s_i]]_2 + \ldots + [[s_i]]_N) + \sum_{i=1}^{n} a_i^{(k,2)} [[s_i]]_1 \right).$$

This is just another instance of the PowAff2(q, m, n) problem since the terms involving $[[s_i]]_2 + \cdots + [[s_i]]_N$ are now constants, and the variables are the coordinates of $[[\mathbf{s}]]_1$. The new instance is no less random than the original one if the shares for each party are random (since we then just add a random constant to each affine form).

The structure of this MPC protocol is particular to the structure of PowAff2(q, m, n) polynomials and uses m iterations of the standard MPC protocol for checking multiplicative triples [7, 35] that we discussed in Section 4.1.1. In the case of PowAff2(q, m, n), we are checking that $z_k = x_k \cdot y_k$ for all $k \in [m]$ since if this is true, then we must have $f_k(\mathbf{s}) = t_k$ for all $k \in [m]$. This protocol would thus not work for a generic instance of the MQ problem, since we would not have this nice multiplicative structure to work with.

4.2.3 Underlying Functions

The key generation, signature, and verification algorithms for Biscuit make use of several underlying functions to generate randomness, evaluate circuits, and perform various other background computations. These include several hash functions and pseudorandom functions, functions that make use of the binary tree structure which stores the views of parties during the MPC protocol, functions to sample random elements from specified sets, and functions to evaluate the values of polynomials efficiently. We provide short descriptions of each function in Appendix B.1, and their full technical specifications can be found in [10].

4.2.4 Biscuit Algorithms

The algorithms for Biscuit require the parameters τ, N, q, n, and m as in the MPC protocol for the PowAff2(q, m, n) problem, as well as an additional security parameter λ.

4.2.4.1 Key Generation Algorithm

The key generation procedure generates the public/secret key pair for Biscuit. The function $\mathbf{f}$ as in PowAff2(q, m, n) is generated with random coefficients. We then generate $\mathbf{s} \in \mathbb{F}_q^n$ at random and evaluate $\mathbf{t} = \mathbf{f}(\mathbf{s})$. The secret key consists of the random seed used to generate $\mathbf{f}$ (so the Signer and Verifier can later re-generate it themselves) as well as the values $\mathbf{s}, \mathbf{t}$, and $\mathbf{y} = (A_{1,2}(\mathbf{s}), \ldots, A_{m,2}(\mathbf{s}))$. The public key consists of the same random seed and the value $\mathbf{t}$. The pseudocode for this procedure is in Appendix B.2.

4.2.4.2 Signing Algorithm

The signing algorithm takes as input the secret key, sk, and the message to be signed, $\mu \in \{0,1\}^*$. The Signer reconstructs the function $\mathbf{f}$ using the seed contained in the secret key and produces a signature in five phases. We provide a high-level description of each phase of the algorithm here and provide pseudocode in Appendix B.3.

Phase 1: the Signer commits to the seeds and views of the parties in the MPC protocol.

During this phase, for each iteration of the MPC protocol, the Signer pseudorandomly generates the input shares for each party, using a salt and roots derived from an XOF evaluated on μ and some other random input, and corrects the shares for the first party to ensure consistency with the full values. They store these error-correcting values. We can see this as creating completely random shares for parties $2, \ldots, N$ and computing what the correct share for party 1 would then have to be. They then complete the first step of the MPC protocol for PowAff2(q, m, n) from Section 4.2.2 for each party, which is to generate shares of the values x_k, y_k, and z_k for each $k \in [m]$. They commit to the view of each party in the protocol at this stage and store this information in the commitment value σ_1.

Phase 2: the Signer generates challenges for the MPC protocol.

First, the Signer generates the hash value h_1 using salt, μ, and σ_1. Then, the Signer generates the challenge ε, as in step 2 of the MPC protocol in Section 4.2.2, for each iteration of the MPC protocol, using their first hashed value, h_1. These challenges are a part of the Fiat-Shamir transform (in the zero-knowledge proof setting, the Verifier would have provided the challenges through their interaction with the Prover). The Verifier will re-compute the value h_1 using information provided to them and information they compute themselves as part of the process of checking the consistency of the views in the MPC protocol.

Phase 3: the Signer commits to the simulation of the MPC protocol.

During this phase, the Signer executes steps 3-5 for each party in each iteration of the MPC protocol from Section 4.2.2. This involves computing shares of α_k and opening this value, followed by computing shares of v_k for each $k \in [m]$. They once again commit to the view of each party in the protocol at this stage and store this information in a commitment value σ_2.

Phase 4: the Signer generates the challenge to open the views of the MPC protocol.

The Signer generates the second hash value h_2 using salt, h_1, and σ_2. At this stage, the Signer generates, using their second hashed value, h_2, the random index $\bar{i}$ for each iteration of the MPC protocol that will not be opened to the Verifier. As with h_1, the Verifier will re-compute the value h_2 using information provided to them

and information they compute themselves as part of the process of checking the consistency of the views in the MPC protocol.

Phase 5: the Signer opens the necessary views of the MPC protocol.

Finally, the Signer generates the path in the binary tree that allows the Verifier to recover the input seed for each party except for $\bar{i}$ in each iteration of the MPC protocol. The signature which they send to the Verifier includes these paths, the salt, and the two hashed values as well as the commitment for party $\bar{i}$, the error-correcting values from Phase 1, and the shares of each α_k for party $\bar{i}$ for each iteration of the MPC protocol. Note that the commitment and the shares of α_k for party $\bar{i}$ do not reveal the private input of this party, so they can be revealed to Verifier without compromising security. In the language of MPCitH, the signature can be seen as containing the information required for the Verifier to reconstruct the views V_i for $i = 1, \ldots, N$, except for $i = \bar{i}$.

4.2.4.3 Verifying Algorithm

The verification algorithm takes as input the public key as well as the message, μ, and its signature. The Verifier reconstructs $\mathbf{f}$ from the seed in the public key and proceeds as usual for MPCitH-based signature schemes.

The Verifier checks the two hashed values by recomputing the work of an honest Signer for all parties which have been opened to them in each iteration of the MPC protocol. They then use the information provided to them by the Signer to reconstruct what should be the view of the unopened party. If this view is consistent with an honest execution of the MPC protocol, then they should recover the same hashed values h_1 and h_2 as were sent to them by the Signer, convincing them that the Signer possesses the secret key. The security of the MPC protocol for PowAff2(q, m, n) implies the security of the signature scheme, as discussed in Section 4.1. Pseudocode for this algorithm is in Appendix B.4.

4.2.5 Security Analysis

The authors claim that the security of Biscuit relies on the hardness of solving the PowAff2(q, m, n) problem, assuming the security of the general MPCitH construction.

4.2.5.1 Solving PowAff2(q, m, n)

To use the standard hardness assumptions of the MQ problem, they must claim that instances of PowAff2(q, m, n) are just as hard to solve as instances of generic MQ for known algorithms. This is formally proven for $m \leq n$, but the case of overdetermined systems used in their implementation is only conjectured [10] (its proof is equivalent

to proving the Fröberg conjecture [58]). The authors acknowledge that there may be other specialized attacks for PowAff2(q, m, n) that are more efficient than the generic algorithms that solve MQ, and they provide the outline for one such attack, which is independent of m, with complexity $\mathcal{O}\big(q^{\frac{3(n+1)}{4}}\big)$. We provide a proof of [10, Theorem 5] below as many details were omitted in its original presentation.

We will assume that we are solving the PowAff2(q, m, n) problem in the context of the digital signature scheme, that is to say that we assume a solution to the problem does indeed exist, as otherwise there could be no corresponding secret key.

We first provide some preliminary definitions and results that will serve as key tools in the attack.

Definition 4.2.2. *Let $f \in \mathbb{F}_q[x_1, \ldots, x_n]$ be a polynomial of degree d. Then we can* ***homogenize*** *f by defining $f^{(h)} \in \mathbb{F}_q[x_1, \ldots, x_n, h]$ to be the polynomial where each monomial of f is multiplied by an appropriate power of h so that every term of f has degree d.*

In particular for one of polynomials f_i defined as in PowAff2(q, m, n), we have

$$f_i^{(h)}(\mathbf{x}, h) = hA_{i,0}^{(h)}(\mathbf{x}, h) + A_{i,1}^{(h)}(\mathbf{x}, h)A_{i,2}^{(h)}(\mathbf{x}, h),$$

where

$$A_{i,j}^{(h)}(\mathbf{x}, h) = a_0^{(k,j)}h + \sum_{i=1}^{n} a_i^{(k,j)}x_i.$$

Note that $f_i^{(h)}$ is homogeneous of degree 2.

We now recall the definition of the polar form of a homogeneous degree 2 polynomial $g \in \mathbb{F}_q[x_1, \ldots, x_n, h]$ from Section 3.2.1.2. It is defined by a matrix $\mathbf{A}$ such that

$$\mathbf{v}^\top \mathbf{A}\mathbf{w} = g(\mathbf{v} + \mathbf{w}) - g(\mathbf{v}) - g(\mathbf{w})$$

for all $\mathbf{v}, \mathbf{w} \in \mathbb{F}_q^{n+1}$.

The polar form of the PowAff2(q, m, n) polynomials will be useful to us because it allows us to work with the polynomials when evaluated at the sums of particular vectors in $\mathbb{F}_q^{n+1}$. In particular, if we take $\mathbf{v}$ in both the kernel of $f \in \mathbb{F}_q[x_1, \ldots, x_n, h]$ and the kernel of the corresponding matrix $\mathbf{A}$ as defined above, then for any $\mathbf{w} \in \mathbb{F}_q^{n+1}$, we have

$$f(\mathbf{v} + \mathbf{w}) = f(\mathbf{w}).$$

The nice matrix representation of the polar form allows us to use as much linear algebra as possible during the attack, which is of course more efficient than working directly with quadratic polynomials.

We now state the key mathematical tool that will allow us to develop the specialized attack for PowAff2(q, m, n).

Theorem 4.2.3. *Let $f_i(\mathbf{x}) = t_i$ for $i \in [m]$ be an instance of the PowAff2(q, m, n) problem. Then there exists a subspace W of $\mathbb{F}_q^{n+1}$ such that, if $W \cap W^\perp = \emptyset$, there exists $(\mathbf{x}, h_x) \in W^\perp$ for which $f_i^{(h)}(\mathbf{x}, h_x) = t_i$ for all $i \in [m]$ such that $t_i \neq 0$. Moreover, given this partial solution $(\mathbf{x}, h_x)$, there exists a solution to the entire homogenized PowAff2(q, m, n) instance of the form $(\mathbf{x}, h_x) + (\mathbf{y}, h_y)$ where $(\mathbf{y}, h_y) \in W$.*

Proof: Assume that $\mathbf{t} = (t_1, \ldots, t_m)$ has $k < n$ non-zero coordinates and that they are, without loss of generality, $t_1, \ldots, t_k$.

For every $i \in [k]$, let $\mathbf{A}_i$ be the matrix representing the polar form of $f_i^{(h)}$. Define the following two subspaces of $\mathbb{F}_q^{n+1}$:

$$O_i = \{(\mathbf{x}, h) \in \mathbb{F}_q^{n+1} | A_{i,0}^{(h)}(\mathbf{x}, h) = A_{i,1}^{(h)}(\mathbf{x}, h) = 0\};$$
$$K_i = \{(\mathbf{x}, h) \in \mathbb{F}_q^{n+1} \mid \mathbf{A}_i(\mathbf{x}, h)^\top = \mathbf{0}\}.$$

We then let $W_i = O_i \cap K_i$ and consider the subspace

$$W = W_1 \cap \ldots \cap W_k.$$

Note that if $\mathbf{w} \in W$, then for all $i \in [k]$,

$$\begin{aligned} f_i^{(h)}(\mathbf{w}) &= w_1 A_{i,0}^{(h)}(\mathbf{w}) + A_{i,1}^{(h)}(\mathbf{w}) A_{i,2}^{(h)}(\mathbf{w}) \\ &= w_1 \cdot 0 + 0 \cdot A_{i,2}^{(h)}(\mathbf{w}) \text{ (since } \mathbf{w} \in O_i) \\ &= 0. \end{aligned}$$

Thus, since $t_i \neq 0$ for all $i \in [k]$, a solution to the first k homogenized equations must lie in the (set-theoretic) complement of W. Moreover, assuming that $W \cap W^\perp = \emptyset$, we have $\mathbb{F}_q^{n+1} = W^\perp \oplus W$ (where $W^\perp$ is the orthogonal complement of W in $\mathbb{F}_q^{n+1}$). The authors of [10] verified experimentally that this assumption holds with high probability. Thus, we can write such a solution as $(\mathbf{x}, h_x) + (\mathbf{y}, h_y)$ for some $(\mathbf{x}, h_x) \in W^\perp$ and $(\mathbf{y}, h_y) \in W$. Now we use our definition of W to make the following observation for all $i \in [k]$:

$$\begin{aligned} f_i^{(h)}((\mathbf{x}, h_x) + (\mathbf{y}, h_y)) &= (\mathbf{x}, h_x)^\top \mathbf{A}_i(\mathbf{y}, h_y) + f_i^{(h)}(\mathbf{x}, h_x) + f_i^{(h)}(\mathbf{y}, h_y) \text{ (by definition)} \\ &= f_i^{(h)}(\mathbf{x}, h_x) \text{ (since } (\mathbf{y}, h_y) \in W). \end{aligned}$$

Let $\mathcal{P} = \{(\mathbf{x}, h_x) \in W^\perp \mid \text{for all } i \in [k], f_i^{(h)}(\mathbf{x}, h_x) = t_i\}$. By the preceding, every solution to the first k homogeneous equations is of the form $(\mathbf{x}, h_x) + (\mathbf{y}, h_y)$ with

$(\mathbf{x}, h_x) \in \mathcal{P}$ and $(\mathbf{y}, h_y) \in W$. Now, we know that a solution to the entire homogenized PowAff2$(q, m.n)$ system must exist in $\mathcal{P} + W$, since such a solution must, in particular, satisfy the first k equations. ∎

We now use this theorem to develop an attack that recovers a solution to an instance of PowAff2(q, m, n).

Theorem 4.2.4. *[10, Theorem 5] There exists an algorithm to solve PowAff2(q, m, n) with complexity $\mathcal{O}\left(q^{\frac{3(n+1)}{4}}\right)$ that succeeds with high probability.*

Proof: Consider the PowAff2(q, m, n) system as defined in the previous theorem and recall the notation used for all relevant subspaces and solutions. We will also assume that $W \cap W^{\perp} = \emptyset$ to ensure the success of our attack, which as mentioned in the proof of Theorem 4.2.3, is the case with high probability.

The only further observation that we make in order to solve the PowAff2(q, m, n) instance is that we must restrict ourselves to finding a solution $(\mathbf{x}, h_x) + (\mathbf{y}, h_y)$ in $\mathcal{P} + W$ that satisfies $h_x + h_y = 1$ in order to have found a solution to the original system of equations, rather than just the homogenized one. There must exist such a solution since we know that a solution to the original system of equations exists.

The formal protocol to find a solution based on our existence statement of the previous theorem is given in Figure 4.4.

Note that if $\mathbf{x} + \mathbf{y}$ is returned by the algorithm, then for all $i = 1, \ldots, k$,

$$\begin{aligned} f_i(\mathbf{x} + \mathbf{y}) &= f_i^{(h)}(\mathbf{x} + \mathbf{y}, 1) \\ &= f_i^{(h)}((\mathbf{x}, h_x) + (\mathbf{y}, h_y)) \\ &= (\mathbf{x}, h_x)^{\top} \mathbf{A}_i (\mathbf{y}, h_y) + f_i^{(h)}(\mathbf{x}, h_x) + f_i^{(h)}(\mathbf{y}, h_y) \text{ (by definition of } \mathbf{A}_i) \\ &= 0 + f_i^{(h)}(\mathbf{x}, h_x) + 0 \text{ (since } (\mathbf{y}, h_y) \in W) \\ &= t_i \text{ (since } (\mathbf{x}, h_x) \in \mathcal{P}). \end{aligned}$$

and for all $i = k + 1, \ldots, m$,

$$\begin{aligned} f_i(\mathbf{x} + \mathbf{y}) &= f_i^{(h)}((\mathbf{x}, h_x) + (\mathbf{y}, h_y)) \\ &= t_i \text{ (by the exhaustive search in Step 6).} \end{aligned}$$

So when the algorithm outputs $\mathbf{x} + \mathbf{y}$, it is indeed a solution to the original system of equations.

They provide a rough complexity estimate for this attack in [10] which they measure by the number of evaluations of vectors of polynomials (i.e., evaluations in Steps 5 and 6). The number of such evaluations is thus $|W^{\perp}| + |\mathcal{P}| \cdot |W|$.

1. For each $i \in [m]$, compute the homogenized polynomial

$$f_i^{(h)}(\mathbf{x}, h) = hA_{i,0}^{(h)}(\mathbf{x}, h) + A_{i,1}^{(h)}(\mathbf{x}, h)A_{i,2}^{(h)}(\mathbf{x}, h).$$

   Note that taking $h = 1$ in the above equations gives our original system of polynomials.

2. For each $i \in [m]$, define:
   (a) $O_i = \{(\mathbf{x}, h) \in \mathbb{F}_q^{n+1} | A_{i,0}^{(h)}(\mathbf{x}, h) = A_{i,1}^{(h)}(\mathbf{x}, h) = 0\}$,

   (b) $K_i = \text{RightKernel}(\mathbf{A}_i)$ where $\mathbf{A}_i \in \mathbb{F}_q^{(n+1)\times(n+1)}$ is the matrix representing the polar form of $f_i^{(h)}$,

   (c) $W_i = O_i \cap K_i$.
3. Define $W = W_1 \cap \ldots \cap W_k$.

4. Define $\mathbf{f}_k = (f_1^{(h)}, \ldots, f_k^{(h)})$ and $\mathbf{t}_k = (t_1, \ldots, t_k)$. Set $W^\perp$ to be the orthogonal complement of W in $\mathbb{F}_q^{n+1}$.

5. Define $\mathcal{P} = \{(\mathbf{x}, h_x) \in W^\perp \mid \mathbf{f}_k(\mathbf{x}, h_x) = \mathbf{t}_k\}$.

6. Exhaustively search $(\mathbf{x}, h_x) \in \mathcal{P}$ and $(\mathbf{y}, h_y) \in W$ such that $h_x + h_y = 1$ and

$$f_i^{(h)}((\mathbf{x}, h_x) + (\mathbf{y}, h_y)) = t_i \text{ for all } i = k+1, \ldots, m$$

   and return $\mathbf{x} + \mathbf{y}$. Return $\perp$ if no such vector is found.

Figure 4.4: Algorithm for finding a solution of PowAff2(q, m, n) using Theorem 4.2.3.

They note that with high probability, each W_i for $i = 1, \ldots, k$ has dimension $n+1-3$, so that $\dim(W) = n + 1 - 3k$. From this, we can conclude that $|W| = q^{n+1-3k}$ and $|W^{\perp}| = q^{3k}$.

Since every polynomial $f_i(\mathbf{x})$ in our system has random coefficients, we expect $f_i^{(h)}(\mathbf{x}, 1)$ to be random for randomly chosen $\mathbf{x} \in \mathbb{F}_q^n$. Hence, since elements of $\mathcal{P}$ are solutions to a random quadratic system with k equations in a vector subspace of dimension $3k$, we expect $|\mathcal{P}| = |W^{\perp}|/q^k = q^{2k}$.

Thus, there are $q^{3k} + q^{2k} \cdot q^{n+1-3k} = q^{3k} + q^{n+1-k}$ vector evaluations. This value is minimized (equal to $q^{\frac{3(n+1)}{4}} + q^{\frac{4n+4-n-1}{4}} = 2q^{\frac{3(n+1)}{4}}$) when $k = \frac{n+1}{4}$, so the minimum complexity estimate for this algorithm is $\mathcal{O}(q^{\frac{3(n+1)}{4}})$. ∎

The ideas behind this attack are reminiscent of the structure of many attacks against UOV, in which we use kernels of the polar forms of the public polynomials to find vectors in the "oil subspace", as discussed in Section 3.2.1. While there is no "oil subspace" in PowAff2(q, m, n), we can still find the useful subspace W on which we know something about the behaviour of our polynomials in order to solve the system more easily. However, for this attack on PowAff2(q, m, n), there is still an element of brute force search because finding a vector in the subspace we defined is not a trapdoor to the private key as it is for UOV.

Another way to use the special structure of the PowAff2(q, m, n) system of equations to develop an attack was proposed on the NIST PQC forum [14] and later in a preprint [15]. We begin with the following observation.

Lemma 4.2.5. *Let $f_k(\mathbf{x}) = t_k$ for $k \in [m]$ be an instance of the PowAff2(q, m, n) problem. Then, with high probability, there exists an invertible matrix $\mathbf{M} \in \mathbb{F}_q^{n \times n}$ such that the first $n < m$ equations satisfy the special form*

$$f_k(\mathbf{Mx}) = a_0^{(k,0)} + \sum_{i=1}^{n} a_i^{(k,0)} (\mathbf{Mx})_i + \left(a_0^{(k,1)} + \sum_{i=1}^{n} a_i^{(k,1)} (\mathbf{Mx})_i\right)(a_0^{(k,2)} + x_k),$$

which, for $k \in [n]$, only contains quadratic terms that involve the variable x_k.

Proof: We consider the affine form

$$\begin{aligned} A_{k,2}(\mathbf{x}) &= a_0^{(k,2)} + \sum_{i=1}^{n} a_i^{(k,2)} x_i \\ &= a_0^{(k,2)} + \mathbf{w}_k^{\top} \mathbf{x} \end{aligned}$$

where $\mathbf{w}_k^{\top} = [a_1^{(k,2)}, \ldots, a_n^{(k,2)}]$.

Recall that $m > n$ and consider the set of vectors $\{\mathbf{w}_1, \ldots, \mathbf{w}_n\} \subseteq \mathbb{F}_q^n$. Suppose it forms a basis for $\mathbb{F}_q^n$, so that there exists some change of basis matrix $\mathbf{M} \in \mathbb{F}_q^{n \times n}$ such that $\mathbf{w}_k^\top \mathbf{M} = \mathbf{e}_k^\top$ (where $\mathbf{e}_k$ is the k-th standard basis vector).

Then, we see that

$$\begin{aligned} A_{k,2}(\mathbf{Mx}) &= a_0^{(k,2)} + \mathbf{w}_k^\top \mathbf{Mx} \\ &= a_0^{(k,2)} + \mathbf{e}_k^\top \mathbf{x} \\ &= a_0^{(k,2)} + x_k. \end{aligned}$$

Now for $k = 1, \ldots, n$, we have

$$f_k(\mathbf{Mx}) = a_0^{(k,0)} + \sum_{i=1}^{n} a_i^{(k,0)} (\mathbf{Mx})_i + \Big(a_0^{(k,1)} + \sum_{i=1}^{n} a_i^{(k,1)} (\mathbf{Mx})_i\Big)(a_0^{(k,2)} + x_k),$$

which only contains quadratic terms that involve the variable x_k.

Note that the probability that $\{\mathbf{w}_1, \ldots, \mathbf{w}_n\} \subseteq \mathbb{F}_q^n$ forms a basis for $\mathbb{F}_q^n$ is

$$\Big(1 - \frac{1}{q^n}\Big)\Big(1 - \frac{1}{q^{n-1}}\Big) \ldots \Big(1 - \frac{1}{q}\Big).$$

Thus, such a matrix $\mathbf{M}$ exists with high probability. Moreover, we may, by reordering the equations, choose any n-element subset of the vectors $\{\mathbf{w}_1 \ldots, \mathbf{w}_m\}$ as our attempt at a basis, further increasing our probability of success. ∎

Theorem 4.2.6. *Under the assumption of Lemma 4.2.5, there is an algorithm that solves PowAff2(q, m, n) with worst-case complexity* $\mathcal{O}\big(\big(\frac{n}{2}\big)^\omega q^{\frac{1}{2}n}\big)$.

Proof: With high probability, we can find a change of basis matrix such that for $k = 1, \ldots, n$, the equation f_k only contains quadratic terms involving the variable x_k, as in the previous lemma.

Now, we can guess the values of the first $\frac{n}{2}$ variables $x_1, \ldots, x_{\frac{n}{2}}$, so that $\{f_k(\mathbf{Mx})\}_{k=1,\ldots,\frac{n}{2}}$ becomes a linear system in the variables $x_{\frac{n}{2}}, \ldots, x_n$, which can be solved using Gaussian elimination. If a solution exists, we then plug it into each equation in $\{f_k(\mathbf{Mx})\}_{k=\frac{n}{2}+1,\ldots,m}$ to see if it is a solution to the entire system of equations. If not, we make a new guess of the first $\frac{n}{2}$ variables and repeat the process. The worst-case complexity of this approach is $\mathcal{O}\big(\big(\frac{n}{2}\big)^\omega q^{\frac{1}{2}n}\big)$ since we must try every possible guess for the values of the first $\frac{n}{2}$ variables and perform Gaussian elimination to solve for the remaining $\frac{n}{2}$ variables. ∎

4.2.5.2 Forgery Attacks

For the security of Biscuit against forgery, the authors cite the well-known Kales-Zaverucha forgery attack [34] on MPCitH-based signature schemes with τ repetitions of a secure MPC protocol, which we discussed in Section 4.1.5. Such an attack requires $\mathrm{KZ}_\tau(p_1, p_2)$, as defined in Equation 4.1.1, calls to hash functions where p_1 is the false probability rate of the MPC protocol and $p_2 = \frac{1}{N}$ is the probability of guessing some of the views of the parties that remain unopened.

Assume that an attacker has managed to find a solution solution to $m - u$ of the equations in the PowAff2(q, m, n) instance. Then $p_1 = \frac{1}{q^u}$, as discussed in Section 4.2.2, and the number of steps required for the forgery attack is

$$\mathrm{KZ}_\tau\left(\frac{1}{q^u}, \frac{1}{N}\right) = \min_{\{\tau_1,\tau_2|\tau_1+\tau_2=\tau\}}\left\{\frac{1}{\sum_{i=\tau_1}^{\tau}\binom{\tau}{i}(1-\frac{1}{q^u})^{\tau-i}} + N^{\tau_2}\right\}.$$

4.2.6 Other Algebraic Properties of PowAff2

We now discuss two other interesting properties arising from the special structure of the polynomials used in Biscuit, that while not obviously applicable to attacks, offer more insight into how they differ from generic MQ polynomials. These properties were first observed in [15].

4.2.6.1 Immediate Computation of a Gröbner Basis when $n = 2m$

In [15], the authors demonstrate how in a particular underdetermined instance of PowAff2(q, m, n) with $n = 2m$, a Gröbner basis can be immediately computed via a linear change of variables with high probability. Note that this construction is not applicable for a cryptographic attack to Biscuit, since the scheme always uses the overdetermined case with $m > n$.

Given an instance of PowAff2(q, m, n), we will define the following values and vectors for ease of notation. For all $k \in [m]$, we define

$$c_k = a_0^{(k,0)} + a_0^{(k,1)}a_0^{(k,2)}, \mathbf{u}_k = \begin{bmatrix} a_1^{(k,0)} + a_0^{(k,1)}a_1^{(k,2)} + a_0^{(k,2)}a_1^{(k,1)} \\ a_2^{(k,0)} + a_0^{(k,1)}a_2^{(k,2)} + a_0^{(k,2)}a_2^{(k,1)} \\ \vdots \\ a_n^{(k,0)} + a_0^{(k,1)}a_n^{(k,2)} + a_0^{(k,2)}a_n^{(k,1)} \end{bmatrix},$$

$$\mathbf{v}_k = \begin{bmatrix} a_1^{(k,1)} \\ a_2^{(k,1)} \\ \vdots \\ a_n^{(k,1)} \end{bmatrix}, \text{ and } \mathbf{w}_k = \begin{bmatrix} a_1^{(k,2)} \\ a_2^{(k,2)} \\ \vdots \\ a_n^{(k,2)} \end{bmatrix}. \tag{4.2.1}$$

This allows us to write each polynomial in the PowAff2(q, m, n) instance more compactly as

$$f_k(\mathbf{x}) = c_k + \mathbf{u}_k^\top \mathbf{x} + (\mathbf{v}_k^\top \mathbf{x}) \cdot (\mathbf{w}_k^\top \mathbf{x}).$$

Now suppose that $n = 2m$ and that $\mathbf{v}_1, \ldots, \mathbf{v}_m, \mathbf{w}_1, \ldots, \mathbf{w}_m$ are linearly independent which, as discussed in the proof of Lemma 4.2.5, is the case with high probability since they are a set of n random vectors in $\mathbb{F}_q^n$. Then, define new variables $y_k := \mathbf{v}_k^\top \mathbf{x}$ and $z_k := \mathbf{w}_k^\top \mathbf{x}$ and define the vectors $\mathbf{y} := (y_1, \ldots, y_m)$ and $\mathbf{z} := (z_1, \ldots, z_m)$. Then we can rewrite f_k as

$$f'_k(\mathbf{y}, \mathbf{z}) = y_k z_k + \mathbf{u}_k'^\top \mathbf{y} + \mathbf{v}_k'^\top \mathbf{z} + c_k$$

where $\mathbf{u}'_k \in \mathbb{F}_q^m$ and $\mathbf{v}'_k \in \mathbb{F}_q^m$ are vectors of coefficients for the new variables, which exist since $\mathbf{u}_k$ can be written as a linear combination of the basis vectors $\mathbf{v}_1, \ldots, \mathbf{v}_m, \mathbf{w}_1, \ldots, \mathbf{w}_m$. Note that since $n = 2m$, the polynomial f'_k is still quadratic in n variables, but now has only one quadratic term, namely $y_k z_k$.

The following lemma demonstrates that $\{f'_k\}$ is a Gröbner basis for the ideal $\mathcal{I} = \langle f'_1, \ldots, f'_m \rangle$.

Lemma 4.2.7. *[15, Lemma 2] The set $G = \{f'_k\}_{k \in [m]}$ is a Gröbner basis with respect to any graded monomial order.*

Proof: In a graded order, the leading monomial of f'_k is $y_k z_k$ for any $k \in [m]$, since it is the only quadratic term. Thus, the leading monomials of distinct polynomials in $G = \{f'_k\}_{k \in [m]}$ have no common factor, so the S-polynomial of any two $f'_i, f'_j \in G$ is

$$\begin{aligned} S(f'_i, f'_j) &= y_i z_i f'_j - y_j z_j f'_i \\ &= y_i z_i (f'_j - y_j z_j) - y_j z_j (f'_i - y_i z_i). \end{aligned}$$

We wish to show that dividing $S(f'_i, f'_j)$ by G (according to the multivariate division algorithm) gives a remainder of zero. We will first divide the term $g := y_i z_i (f'_j - y_j z_j)$ by G. All highest degree monomials in g have the form $y_i z_i y_k$ or $y_i z_i z_k$, so the only leading monomial of G by which they are divisible is $y_i z_i$ (the leading monomial of f'_i). Thus, dividing g by G amounts to dividing it by f'_i and gives

$$g = f'_i (f'_j - y_j z_j) + (y_i z_i - f'_i)(f'_j - y_j z_j).$$

Thus, since $S(f'_i, f'_j)$ is comprised of two terms of this form, when we divide it by G, we obtain

$$\begin{aligned} S(f'_i, f'_j) &= f'_i (f'_j - y_j z_j) + (y_i z_i - f'_i)(f'_j - y_j z_j) - f'_j (f'_i - y_i z_i) - (y_j z_j - f'_j)(f'_i - y_i z_i) \\ &= f'_i (f'_j - y_j z_j) - f'_j (f'_i - y_i z_i) \end{aligned}$$

which has a remainder of zero. Therefore, by Buchberger's criterion, G is a Gröbner basis. ∎

Thus, in the particular underdetermined case where $n = 2m$, we are able to avoid costly algorithms for computing Gröbner bases and use the special structure of PowAff2(q, m, n) to obtain one efficiently. This allows us to solve for $\mathbf{y}$ and $\mathbf{z}$ efficiently, which in turn allows us to find $\mathbf{x}$.

Note that the Gröbner basis we obtain in Lemma 4.2.7 is with respect to a graded order, so it would have to be transformed into one with respect to the lexicographic order to be used for solving the system of equations through back substitution. As discussed in Section 2.4.2.1, there exist efficient algorithms for executing this change of ordering.

4.2.6.2 Reducing the Number of Variables by One Third

The authors of [15] describe a simple way to again use linear transformations on the polynomials of a PowAff2(q, m, n) to exploit their special structure, this time obtaining a system of equations in which one third of the variables have been eliminated.

We assume that n is a multiple of 3 so we can define $l = n/3$. We further impose that $m \geq l$. Now suppose that the vectors $\mathbf{u}_1, \ldots, \mathbf{u}_l, \mathbf{v}_1, \ldots, \mathbf{v}_l, \mathbf{w}_1, \ldots, \mathbf{w}_l$, defined in (4.2.1), are linearly independent, which is the case with high probability. Then, define new variables $w_k := \mathbf{u}_k^\top \mathbf{x}$, $y_k := \mathbf{v}_k^\top \mathbf{x}$, and $z_k := \mathbf{w}_k^\top \mathbf{x}$. Then, the first l polynomials in the system become

$$f'_k := y_k z_k + w_k + c_k.$$

Thus, in the remaining $m - l$ equations of the system, we can replace the variable w_k with an equation in y_k and z_k. This eliminates the variables w_k, of which there are $l = n/3$, so the new system has $m - \frac{n}{3}$ polynomials (as we only consider $f'_{l+1}, \ldots, f'_m$) of degree up to four in $\frac{2n}{3}$ variables.

Note that since the reduction in the number of variables and equations comes at the cost of the increased degree of the polynomials, the authors of [15] note that the resulting system is no easier to solve than the original system. However, this transformation is still of interest because it once again points to the non-generic nature of the polynomials in a PowAff2(q, m, n) instance, since such an efficient transformation is not known to exist for a generic MQ system.

4.2.7 Conclusions

The Biscuit digital signature scheme is an interesting case-study in combining ideas from multiple areas of cryptography, in this case, zero-knowledge proofs and multivariate cryptography, to produce a unique protocol based on various security assumptions.

Though MPCitH-based signature schemes generally feature slower signing and verification algorithms than other types of signature schemes (as they require repetitive computations for all parties in the protocol over several iterations), the introduction of PowAff2(q, m, n) polynomials by the authors of Biscuit provides a promising way to increase the efficiency of these algorithms [10].

With the introduction of a new hard cryptographic problem such as PowAff2(q, m, n), there is always room for new attacks to surface. This was the case with the special structure of the polynomials used in PowAff2(q, m, n), as demonstrated in Section 4.2.5. These attacks point out the many ways in which the polynomials in Biscuit do not behave generically, which is always a concern in cryptographic protocols, as we would like the polynomials to appear as random as possible. While the authors of Biscuit have been able to increase their parameters slightly in their more recent version [9] to avoid the attack of [15] outlined in Theorem 4.2.6, we cannot yet be sure whether there are even more efficient attacks on PowAff2(q, m, n). The new polynomial structures used in Biscuit, while needing rigorous study before the signature scheme can be considered for standardization, provide a rich new set of problems to explore.

Chapter 5

MPPK/DS Signature Scheme

In this chapter, we study a proposed post-quantum digital signature scheme, Multivariate Polynomial Public Key Digital Signature (MPPK/DS), which was created with the intention to submit to the NIST standardization process in 2023, but was eventually not submitted. We study MPPK/DS in its original form [41]. The content of this chapter is adapted from a paper that we published in La Matematica in 2024, entitled "A Classically Efficient Forgery of MPPK/DS Signatures" [44].

In Section 5.1, we first give the key generation, signature, and verification algorithms for MPPK/DS and discuss the correctness of the scheme. We then present some slight modifications that are required for the signing algorithm in Section 5.2.1. In Proposition 5.2.4, we present an original efficient attack of complexity $\mathcal{O}(8^m)$ on the scheme using a small cyclic subgroup of $\mathbb{Z}_p^*$ (the group over which the signer calculates the signature) and describe in Proposition 5.2.5 an optimization using ideas from another algebraic attack of polynomial complexity proposed in [30]. These two attacks demonstrate that the MPPK/DS protocol is not secure as it is presented in [41], with its most fatal flaw being that the signature consists of several powers of some random base $g \in \mathbb{Z}_p^*$. In Section 5.3, we then provide several experimental results to verify the validity and efficiency of our attacks. We conclude in Section 5.4 with remarks on the consequences of our attacks on MPPK/DS in the post-quantum context and conjecture that the fundamental structure of the scheme renders it insecure.

5.1 The MPPK/DS Scheme

In this section, we present the MPPK/DS algorithms with some minor modifications. We also demonstrate the correctness of the scheme and discuss the security claims made by the authors [41].

The MPPK/DS signature scheme is adapted from MPPK KEM, a key encapsulation mechanism, created by the same authors [42]. The public key in MPPK/DS consists of a collection of four multivariate polynomials and the private key consists of two noise constants and four univariate polynomials (to be evaluated at the value of the message by the signer). Particular algebraic relations between the polynomials in the public and private keys are required for correctness of the signature scheme. An MPPK/DS signature consists of five powers of a random base generated by the Signer, and this base is required to remain unknown to the Verifier to ensure that they cannot use the quantum discrete logarithm to solve for the exponents generated by the Signer.

5.1.1 Key Generation Algorithm

The following security parameters are agreed upon between the Signer and the Verifier before communication:

- $p = 2^x q + 1$ a generalized safe prime (meaning that p and q are both prime and x is a positive integer): this determines the domains $\mathbb{Z}_p$ and $\mathbb{Z}_{\varphi(p)} = \mathbb{Z}_{p-1}$ for all coefficients and variables;

- m a positive integer - this specifies the number of noise variables that will be used;

- n a positive integer - this specifies the degree of the message variable x_0 in the base polynomial β that will be constructed;

- λ a positive integer - this specifies the degree of two univariate polynomials to be constructed; and

- $l_1, \ldots, l_m$ positive integers - these determine the degrees of each of the m noise variables in the base polynomial β.

Recommended values for the above parameters are given in [41]. The highest such recommendations, to achieve NIST Level V security based on the attacks presented in [41] are

$$(\log_2 q, x, \log_2 p, m, n, \lambda) = (32, 32, 64, 2, 6, 3).$$

All polynomials generated for the keys are in the ring $\mathbb{Z}_{p-1}[x_0, x_1, \ldots, x_m]$, sometimes lying in specific subsets embedded in this ring. During the signing and verification procedures, these polynomials will be evaluated at the point $(\mu, r_1, \ldots, r_m)$ where μ is the (possibly hashed) message being signed and $r_1, \ldots, r_m$ are random noise variables generated by the Verifier. Note that the Signer needs only to evaluate univariate

polynomials in x_0 at μ, so they do not require access to $r_1, \ldots, r_m$.

For any polynomial $F \in \mathbb{Z}_{p-1}[x_0, \ldots, x_m]$, we denote by F_i the coefficient of x_0^i, so in general $F_i \in \mathbb{Z}_{p-1}[x_1, \ldots, x_m]$. Note that when $F \in \mathbb{Z}_{p-1}[x_0]$ is a univariate polynomial, we have $F_i \in \mathbb{Z}_{p-1}$.

First, the Signer creates the following objects for the private key:

1. Two polynomials $f, h \in \mathbb{Z}_{p-1}[x_0]$ of degree λ with random coefficients;
2. Even integers R_0, R_n chosen randomly from $\mathbb{Z}_{p-1}$;
3. Two polynomials $E_\phi, E_\psi \in \mathbb{Z}_{p-1}[x_0]$ of degree $n + \lambda - 1$ with 0 constant term and otherwise random coefficients.

The private key used for signing is then

$$s = (f, h, R_0, R_n, E_\phi, E_\psi). \tag{5.1.1}$$

Then, the Signer creates the following intermediate objects:

1. A base polynomial $\beta \in \mathbb{Z}_{p-1}[x_0, x_1, \ldots, x_m]$, of degree n in x_0 and degree l_i in x_i for $i = 1, \ldots, m$, with random coefficients;
2. Two product polynomials

   $$\phi = f\beta \qquad \text{and} \qquad \psi = h\beta$$

   so that for every $i \in \{0, \ldots, \lambda\}$ and $j \in \{0, \ldots, n\}$ their coefficients of x_0^k are

   $$\phi_k = \sum_{k=i+j} f_i \beta_j \qquad \text{and} \qquad \psi_k = \sum_{k=i+j} h_i \beta_j;$$

3. Two modified polynomials obtained from ϕ and ψ respectively by removing their constant and leading terms:

   $$\Phi = \sum_{k=1}^{n+\lambda-1} \phi_k x_0^k \qquad \text{and} \qquad \Psi = \sum_{k=1}^{n+\lambda-1} \psi_k x_0^k.$$

Finally, the Signer creates the following objects for the public key:

1. Two polynomials: $P = R_0(\Phi - E_\phi)$ and $Q = R_n(\Psi - E_\psi)$;
2. Two noise polynomials: $N_0 = R_0\beta_0$ and $N_n = R_n\beta_n x_0^{n+\lambda}$.

The public key used for verifying signatures is then

$$v = (P, Q, N_0, N_n). \tag{5.1.2}$$

5.1.2 Signing Algorithm

The signing algorithm takes as input a (possibly hashed) message $\mu \in \mathbb{Z}_{p-1}$ [2] and private key s as in (5.1.1). A base $g \neq 1$ is chosen randomly from the multiplicative group $\mathbb{Z}_p^*$.

The Signer computes the following:

1. $A = g^a \bmod p$ where $a = R_0 f(\mu)$;
2. $B = g^b \bmod p$ where $b = R_n h(\mu)$;
3. $C = g^c \bmod p$ where $c = s_0(\mu) = R_n[h(\mu)f_0 - f(\mu)h_0]$;
4. $D = g^d \bmod p$ where $d = s_n(\mu) = R_0[h(\mu)f_\lambda - f(\mu)h_\lambda]$;
5. $E = g^e \bmod p$ where $e = t(\mu) = R_0 R_n[h(\mu)E_\phi(\mu) - f(\mu)E_\psi(\mu)]$.

The digital signature corresponding to μ is then the tuple (A, B, C, D, E), and the output of the signing algorithm [3] is $S_s(\mu) = (\mu, (A, B, C, D, E))$.

5.1.3 Verifying Algorithm

The verifying algorithm takes as input a message μ, a digital signature $s' = (A, B, C, D, E)$ and the public key v as in (5.1.2).

The Verifier chooses non-zero values $r_1, \ldots, r_m$ randomly from $\mathbb{Z}_{p-1}$. They then evaluate their four polynomials on the tuple $(\mu, r_1, \ldots, r_m)$, to obtain the following elements of $\mathbb{Z}_{p-1}$:

$$\overline{P} = P(\mu, r_1, \ldots, r_m), \qquad \overline{Q} = Q(\mu, r_1, \ldots, r_m),$$
$$\overline{N_0} = N_0(r_1, \ldots, r_m), \qquad \overline{N_n} = N_n(\mu, r_1, \ldots, r_m).$$

The output of the algorithm is

$$V_v(\mu, s') = \begin{cases} (\mu, \text{VALID}) & \text{if } A^{\overline{Q}} = B^{\overline{P}} C^{\overline{N_0}} D^{\overline{N_n}} E \bmod p, \\ (\mu, \text{INVALID}) & \text{otherwise.} \end{cases} \tag{5.1.3}$$

5.1.4 Correctness

We now show how the correctness of MPPK/DS arises from the algebraic relations between the polynomials in the public and private keys.

[2] The message space was stated to be $\mathbb{Z}_p$ in [41], but evaluating a polynomial in $\mathbb{Z}_{p-1}[x_0]$ at a value in $\mathbb{Z}_p$ is not well-defined. We thus change it to $\mathbb{Z}_{p-1}$ and assume that arithmetic is done mod $p-1$ when evaluating the polynomials. Note that the notation $GF(p-1)$ was used in [41] to denote $\mathbb{Z}_{p-1}$, but we change it here since this ring is not a field.

[3] The signing algorithm in [41] also stipulated that none of A, B, C, D or E should be equal to 1 and that if one was, a new base g should be chosen. We discuss in Section 5.2.1 why this condition must be removed.

Proposition 5.1.1. *MPPK/DS is a correct digital signature scheme, that is for any message $\mu \in \mathbb{Z}_{p-1}$, public key v, and corresponding private key s, we have $V_v(S_s(\mu)) = (\mu,\ VALID)$.*

Proof: The correctness of the algorithm relies on the fact that since p is prime, if $a \equiv b \bmod \varphi(p)$, then $g^a \equiv g^b \bmod p$, for any $g \in \mathbb{Z}_p$, by Fermat's Little Theorem.

Recall that the private key is $(f, h, R_0, R_n, E_\phi, E_\psi)$ and the public key is (P, Q, N_0, N_n). We first claim that

$$h(R_n f_0 N_0 + R_0 f_\lambda N_n + R_n P + R_0 R_n E_\phi) = f(R_n h_0 N_0 + R_0 h_\lambda N_n + R_0 Q + R_0 R_n E_\psi). \tag{5.1.4}$$

To see why, recall the two polynomials $\phi = f\beta$ and $\psi = h\beta$. We then have

$$R_0 R_n h\phi = R_0 R_n f\psi$$

where

$$\begin{aligned} R_0 R_n \phi &= R_0 R_n f_0 \beta_0 + R_0 R_n f_\lambda \beta_n x_0^{n+\lambda} + R_0 R_n \Phi \\ &= R_n f_0 R_0 \beta_0 + R_0 f_\lambda R_n \beta_n x_0^{n+\lambda} + R_n R_0 (\Phi - E_\phi) + R_0 R_n E_\phi \\ &= R_n f_0 N_0 + R_0 f_\lambda N_n + R_n P + R_0 R_n E_\phi \end{aligned}$$

and similarly

$$R_0 R_n \psi = R_n h_0 N_0 + R_0 h_\lambda N_n + R_0 Q + R_0 R_n E_\psi.$$

which gives the equality as claimed after multiplying by h and f as required.

Rearranging (5.1.4) to isolate the term containing Q gives

$$\begin{aligned} f R_0 Q &= h R_n P + (R_n f_0 h - R_n h_0 f) N_0 + (R_0 f_\lambda h - R_0 h_\lambda f) N_n + R_0 R_n h E_\phi - R_0 R_n f E_\psi \\ &= h R_n P + s_0 N_0 + s_n N_n + t \end{aligned} \tag{5.1.5}$$

with

$$s_0 = R_n(h f_0 - f h_0),\ s_n = R_0(h f_\lambda - f h_\lambda),\ \text{and}\ t = R_0 R_n (h E_\phi - f E_\psi)$$

as defined in the signing algorithm.

Now suppose the Signer has produced the valid signature (A, B, C, D, E) on a message $\mu \in \mathbb{Z}_{p-1}$ using the randomly generated base $g \in \mathbb{Z}_p^*$. The Verifier then chooses random noise variables $r_1, \ldots, r_m \in \mathbb{Z}_{p-1}$ and computes the values $\overline{Q}, \overline{P}, \overline{N_0}, \overline{N_n}$ of their public key polynomials on $(\mu, r_1, \ldots, r_m)$.

By (5.1.5),

$$f(\mu) R_0 \overline{Q} = h(\mu) R_n \overline{P} + \overline{N_0} s_0(\mu) + \overline{N_n} s_n(\mu) + t(\mu) \in \mathbb{Z}_{p-1},$$

so by Fermat's Little Theorem,

$$g^{f(\mu)R_0\overline{Q}} = g^{h(\mu)R_n\overline{P}+\overline{N_0}s_0(\mu)+\overline{N_n}s_n(\mu)+t(\mu)} \bmod p.$$

Then, by definition of a, b, c, d, and e computed by the Signer,

$$g^{a\overline{Q}} = g^{b\overline{P}+c\overline{N_0}+d\overline{N_n}+e} \bmod p.$$

Thus, since $(A, B, C, D, E) = (g^a, g^b, g^c, g^d, g^e)$, the Verifier using (5.1.3) will indeed output (μ, VALID). ∎

5.1.5 Security Claims

The security proofs in [41] fall into three categories: using the public key to find the private key, using signatures to find the private key, and forging signatures. The authors outline particular attacks in each case and show that the complexity of their attacks is exponential, both with known classical and quantum algorithms.

The attack to find the private key given the public key follows a brute force approach using systems of equations derived from the relations between the polynomials in the two keys. Since the coefficients of these polynomials are in $\mathbb{Z}_{p-1}$ where $p - 1 = 2^x q$, an attacker is required to solve the equations mod q first and "lift" the solutions to the correct coefficients mod $p - 1$. This lifting step is complex, so that coupled with the number of equations to be solved, the attack has exponential complexity.

The attack outlined to find the private key given some collection of signatures also follows a brute force approach, as the authors prove that there is no algebraic way to express the components of a signature in terms of each other, so the five components of a signature must be treated separately. The attack uses discrete logarithms relative to some generator $\bar{g}$ of $\mathbb{Z}_p$; in general $\bar{g} \neq g$ since the base g used for signing was not necessarily a generator and is not known to the attacker. The number of equations to be solved and the same lifting technique as in the previous attack require exponential time to complete.

Finally, the forgery attack on a given message μ is a brute force search through $\mathbb{Z}_p$ for elements A, B, C, D and E that will pass the verification algorithm for any choice of random noise variables $r_1, \ldots, r_m$ that a Verifier could make. This has exponential complexity since, in the worst-case, all of $\mathbb{Z}_p$ must be searched, and all random choices of noise variables in $\mathbb{Z}_{p-1}$ must be checked. It is this attack that we will improve upon in the following sections.

5.2 Cryptanalysis of MPPK/DS

In this section, we present our cryptanalysis of MPPK/DS, which includes observations on the signing algorithm as it was presented in [41], as well as two original, efficient attacks, one of which combines our ideas with those from Guo's attack [30] for increased efficiency.

5.2.1 Observations on Signing Algorithm

In [41], the authors assert that if one of the components of a signature (A, B, C, D, E) is equal to 1, then a new base g should be chosen and the signature should be recomputed. This seems like a reasonable assertion at first glance: if the chosen base g does not have maximal order, this order could divide one or more of the computed values a, b, c, d or e, which could reveal information to the Verifier.

Our first observation is that there are some messages for which certain signature components must be equal to 1 (*i.e.*, a new base g cannot be chosen to avoid the appearance of a 1).

Lemma 5.2.1. *The signature (A, B, C, D, E) produced by the signing algorithm for the message $\mu = 2^{x-1}q \in \mathbb{Z}_{p-1}$ must have $C = E = 1$ for every choice of base $g \in \mathbb{Z}_p^*$.*

Proof: Recall that $p - 1 = 2^x q$. Since hf_0 and fh_0 both have constant term $f_0 h_0$, we can factor $h(\mu)f_0 - f(\mu)h_0 = \mu M$ for some $M \in \mathbb{Z}_{p-1}$. Let $\mu = 2^{x-1}q$; since R_n is even, $R_n\mu \equiv 0 \bmod p-1$. Using the notation from the signing algorithm, we compute

$$c = s_0(\mu) = R_n[h(\mu)f_0 - f(\mu)h_0] = R_n\mu M \equiv 0 \bmod p - 1.$$

Thus, regardless of the choice of base g, we have $C = g^c = 1 \in \mathbb{Z}_p$. Similarly, since E_ϕ and E_ψ do not have constant terms and R_0 and R_n are even, we have

$$e = t(\mu) = R_0 R_n[h(\mu)E_\phi(\mu) - f(\mu)E_\psi(\mu)] \equiv 0 \bmod p - 1,$$

implying $E = 1$ for any choice of base g. ∎

Of course, $\mu = 2^{x-1}q$ is not the only message that will produce a signature that will always have a 1. It follows directly from the proof that any message μ for which $R_0\mu$ or $R_n\mu$ is equal to 0 mod $p-1$ gives $C = 1$, and if $R_0 R_n \mu \equiv 0 \bmod p-1$ then $E = 1$. So if R_0 or R_n is divisible by a higher power of 2 or is divisible by q, there will be many more messages μ that yield valid signatures containing 1.

Another important instance is the following.

Proposition 5.2.2. *If f, respectively h, has a root μ, then the signature (A, B, C, D, E) on message μ will have A, respectively B, equal to 1 for every choice of base $g \in \mathbb{Z}_p^*$.*

Proof: Let the message $\mu \in \mathbb{Z}_{p-1}$ be a root of f. Then with the notation from this algorithm, we have $a = R_0 f(\mu) \equiv 0 \bmod p-1$. Thus for any choice of base $g \in \mathbb{Z}_p^*$, we have $A = g^a = 1$. The case of h and b is identical. ∎

In general, we see that the Verifier will have to accept signatures with 1s as valid if they pass the verification algorithm, since they could indeed be honest signatures. However, Proposition 5.2.2 shows an instance where doing so reveals some information about the private key.

5.2.2 An Original Attack on MPPK/DS

We now present an original efficient forgery attack on MPPK/DS. Our attack is modelled after the one presented in [41, Proposition 4.15], which has complexity $\mathcal{O}(p^{4+m})$, but ours uses a significantly smaller search space, reducing the complexity to $\mathcal{O}(8^m)$ as m approaches infinity. In practice, $m \leq 3$ is a small constant, so our attack can be implemented in constant time.

We begin with a simple lemma.

Lemma 5.2.3. *When p is a generalized safe prime there exists an efficient algorithm to identify a generator of $\mathbb{Z}_p^*$.*

Proof: The attacker can find a generator g by randomly selecting an element $g \in \mathbb{Z}_p^*$ and checking that it satisfies

$$g^{\frac{p-1}{2}} \neq 1 \qquad \text{and} \qquad g^{\frac{p-1}{q}} \neq 1$$

since 2 and $q \geq 3$ are the only prime factors of $p-1$. We now estimate the expected number of iterations before a generator is found. First note that the number of generators in $\mathbb{Z}_p^*$ is

$$\varphi(p-1) = \varphi(2^x q) = (2^x - 2^{x-1})(q-1) = 2^{x-1}(q-1).$$

Thus the probability that a randomly selected element of $\mathbb{Z}_p^*$ is a generator is

$$\frac{2^{x-1}(q-1)}{p-1} = \frac{1}{2} - \frac{1}{2q}.$$

Thus, the probability that a randomly selected element is not a generator is $\frac{1}{2} + \frac{1}{2q}$. So after selecting k random elements of $\mathbb{Z}_p^*$, the probability that at least one generator

was selected is

$$1 - \left(\frac{1}{2} + \frac{1}{2q}\right)^k > 1 - \left(\frac{2}{3}\right)^k.$$

Thus we can model this as a Bernoulli trial with probability of success at least $p_0 = 1/3$, yielding an expected number of trials of $\mathcal{E} = 1/p_0 = 3$. ∎

Alternately, one can use the technique of the proof to estimate that to find a generator with probability at least P, one needs to choose $k > \log(1-P)/(1-\log(3))$ random elements g.

Our forgery attack exploits the fact that an honest Signer chooses a random base $g \in \mathbb{Z}_p^*$ and sends the Verifier a signature consisting of several powers of this base. This base g may not have maximal order, and since the Verifier does not have access to g, they cannot determine whether a base of small order was used. This allows an attacker to efficiently produce a signature using a base of small order which a Verifier will accept.

Proposition 5.2.4. *An attacker can forge a valid signature for MPPK/DS on an arbitrary message μ efficiently on a classical computer, with complexity $\mathcal{O}(8^m)$.*

Proof: For the sake of concreteness, we suppose $x \geq 3$ so that $p-1$ is divisible by 8 and choose a base of order 8 for this attack. The attacker chooses a generator $g' \in \mathbb{Z}_p^*$ and sets $g = (g')^{\frac{p-1}{8}}$, which is an element of order 8. The attacker then computes the cyclic subgroup $\langle g \rangle = \{1, g, g^2, \ldots, g^7\}$. This takes a constant number of steps.

Let $\mu \in \mathbb{Z}_{p-1}$ be the message on which the attacker wishes to forge a signature. Then there must be a valid signature (A, B, C, D, E) for μ with all entries in $\langle g \rangle$, since an honest signer could have chosen g as their random base in the signing algorithm.

To forge a signature on μ, the attacker first randomly chooses two sets of noise variables $\xi_1 = (r_1, \ldots, r_m)$ and $\xi_2 = (r'_1, \ldots, r'_m)$ in $\mathbb{Z}_8^m$; let $\overline{Q}_i, \overline{P}_i, \overline{N_{0i}}, \overline{N_{ni}}$ for $i = 1, 2$ be the corresponding values of the public key polynomials, which are now in $\mathbb{Z}_8$. The attacker now searches for $A, B, C, D \in \langle g \rangle$ such that

$$A^{\overline{Q}_1} B^{-\overline{P}_1} C^{-\overline{N_{01}}} D^{-\overline{N_{n1}}} \equiv A^{\overline{Q}_2} B^{-\overline{P}_2} C^{-\overline{N_{02}}} D^{-\overline{N_{n2}}} \mod p. \tag{5.2.1}$$

This is well-defined since the orders of A, B, C, D divide 8. When this is the case, call this value E. This takes at most $|g|^4 = 8^4$ steps. Now, with the candidates found above, the attacker checks that $A^{\overline{Q}} \equiv B^{\overline{P}} C^{\overline{N_0}} D^{\overline{N_n}} E \mod p$ for all choices of random noise variables in $\mathbb{Z}_8^m$; this takes 8^m steps. This will succeed with high probability since a solution is known to exist; if this fails, the attacker chooses another solution to (5.2.1) and repeats.

We claim that if the attack succeeds with $s' = (A, B, C, D, E)$ then the pair (μ, s') will pass the verification algorithm in all cases. Namely, let $\xi = (r_1, \ldots, r_m)$ be any vector of noise variables from $\mathbb{Z}_{p-1}^m$, and let $\overline{Q}, \overline{P}, \overline{N_0}$, and $\overline{N_n}$ be the corresponding evaluations of the public key polynomials. Since $A, B, C, D \in \langle g \rangle$ and g has order 8, the values of $A^{\overline{Q}}$, $B^{\overline{P}}$, $C^{\overline{N_0}}$ and $D^{\overline{N_n}}$ depend only on the values of the exponents mod 8. The properties of polynomials imply in turn that these values mod 8 depend only on the equivalence classes of $r_1, \ldots, r_m$ mod 8, and the attack algorithm ensures that the Verification algorithm passes on each of these equivalence classes.

Thus, the total number of steps for the attack is a constant plus 8^{4+m}. ∎

The only parameter on which the complexity of our attack relies is m, which is recommended to be at most 3 in [41]. In order to render our attack inefficient, the value of m would have to be increased significantly.

A base of order 8 was chosen so that the search space was small, but large enough so that with reasonable likelihood, not all components of the signature would be 1. (By the remarks in the previous section, even such a trivial signature can be valid.) However, the argument can be adapted for any $g \in \mathbb{Z}_p^*$ of non-maximal order.

5.2.3 Another Attack on MPPK/DS

As we developed the attack in Proposition 5.2.4, Hao Guo published a different attack on MPPK/DS in [30]. This attack also constructs a forged signature, but with a different approach. Instead of directly constructing A, B, C, D and E that will pass verification, the attacker instead solves for a, b, c, d and e using the public key polynomials (working separately mod q and mod 2^x, and combining the results using the Chinese Remainder Theorem). The attacker then chooses a base g in $\mathbb{Z}_p^*$ and creates the signature $(g^a, g^b, g^c, g^d, g^e)$. Setting

$$L = (l_1 + 1) \ldots (l_m + 1),$$

the attack in [30] has complexity $\mathcal{O}(5^{\omega-1}L + 4^{\omega-1}xL + \min(x, \log q))$ where $2 \leq \omega \leq 3$ [56], which is in $\mathcal{O}(xL)$. Note that this complexity was determined based on the number of equations and variables in the linear systems that must be solved during the attack.

We can combine ideas from the previous section with methods from [30] to improve our attack as follows.

Proposition 5.2.5. *An attacker can forge a signature for MPPK/DS on an arbitrary message μ on a classical computer in time $\mathcal{O}(L)$.*

Proof: Let $\mu \in \mathbb{Z}_{p-1}$ be a message. As observed in [30], in order to pass the verification algorithm to sign μ, the attacker must find $(a, b, c, d, e) \in \mathbb{Z}_{p-1}^5$ that satisfies the following equality of polynomials in $\mathbb{Z}_{p-1}[x_1, \ldots, x_m]$:

$$\begin{aligned} &aQ(\mu, x_1, \ldots, x_m) \\ =&bP(\mu, x_1, \ldots, x_m) + cN_0(x_1, \ldots, x_m) + dN_n(\mu, x_1, \ldots, x_m) + e \bmod p-1, \end{aligned} \tag{5.2.2}$$

obtained by equating the exponents of either side of the verification equality (5.1.3). Since l_i is the degree of the noise variable x_i, (5.2.2) can be separated into $L = (l_1+1)(l_2+1)\ldots(l_m+1)$ linear equations in a, b, c, d and e by equating the coefficients of the corresponding monomials.

As in the proof of Proposition 5.2.4, the attacker may choose an element $g \in \mathbb{Z}_p^*$ of order 8 in constant time to use as a base for the signature. Therefore it suffices for the attacker to solve (5.2.2) for a, b, c, d, and e mod 8 and use $(g^a, g^b, g^c, g^d, g^e)$ as a signature on μ.

A general algorithm for solving a system of n linear equations in m variables modulo any positive (composite) integer is presented in [56] and has complexity $\mathcal{O}(nm^{\omega-1})$ where $2 < \omega < 3$ is the linear algebra constant. Since we have L equations in the 5 variables a, b, c, d, e, the asymptotic complexity of this attack is simply $\mathcal{O}(L)$. ∎

As the highest recommended value of m in [41] is 3 and $l_i = 1$ for all $i = 1, \ldots, m$ in their reference implementation [5], the above attack can also be implemented in constant time in practice. Moreover, increasing the values of m or l_i to counter the attack will increase the complexity of the Verifier's work correspondingly: the number of nonzero equations in this linear system is equal to the number of terms that the honest Verifier must compute on their random noise variables.

5.3 Experimental Results

In this section, we present several experimental results to demonstrate the efficiency and relevance of our attacks presented in Section 5.2.

5.3.1 Existence of Nontrival Messages with Trivial Signature Elements

In Proposition 5.2.2, we proved that roots of the univariate polynomials f and h will be messages that cannot be signed without the occurrence of 1 as an element of the signature. It remains to discuss whether it is likely that f (or equivalently, h) will indeed have a root in $\mathbb{Z}_{p-1}$.

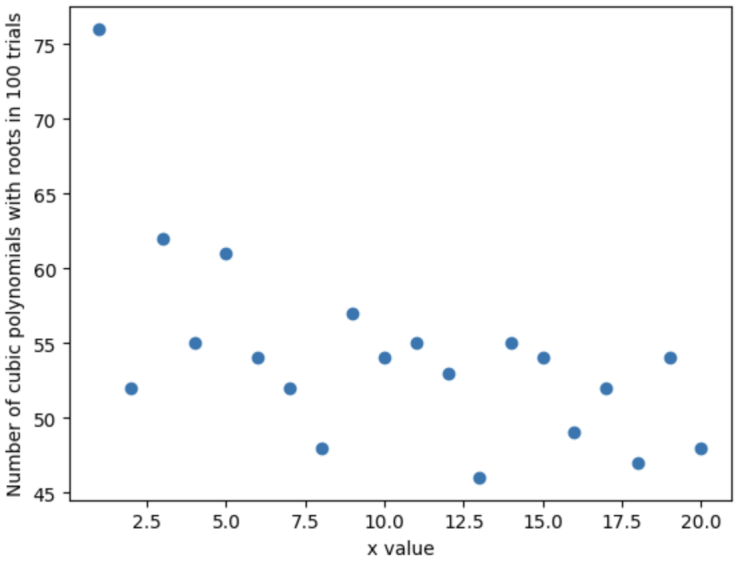

Figure 5.1: Experimental evidence of how many cubic polynomials out of 100 randomly generated ones have a root in $\mathbb{Z}_{2^x}$.

By the Chinese Remainder Theorem, we know that f has a root mod $p-1 = 2^x q$ if and only if it has a root mod q and a root mod 2^x. We will consider f to be of degree 3 (as per the highest recommended security parameters) and assume that, without loss of generality, it is monic.

In $\mathbb{Z}_q[x_0]$, f has no roots if and only if it is irreducible (since q is prime). In $\mathbb{Z}_q[x_0]$, there are $\frac{1}{3}(q^3 - q)$ irreducible monic polynomials of degree 3 [40], so there are

$$q^3 - \frac{1}{3}(q^3 - q) \approx \frac{2}{3}q^3$$

monic polynomials of degree 3 with a root (assuming q is large). Thus, given random $f \in \mathbb{Z}_{p-1}[x_0]$, the probability that it has a root mod q is approximately $\frac{2}{3}$.

Since $\mathbb{Z}_{2^x}$ is not a field, it is more difficult to count how many polynomials in $\mathbb{Z}_{2^x}[x_0]$ have a root, so we turn to experimental evidence to provide an estimate. We generated 100 random cubic polynomials in $\mathbb{Z}_{2^x}[x_0]$ for $x = 1$ to $x = 20$ and counted how many had a root. The graph in Figure 5.1 demonstrates these findings.

We can see that, especially as x grows larger, the number of cubic polynomials with roots in 100 trials is usually between 45 and 55. We will thus estimate that for $x \approx 32$ (the value recommended for highest security), the probability that a cubic polynomial has a root mod 2^x is $\frac{1}{2}$.

Thus, by the Chinese Remainder Theorem, the probability that a random cubic

p	Average Number of Steps	Sample Generator	Element of Order 8
31127341089062649857	2.07	13300341839071835659	1836263857679436181
24728833639095205889	2.07	19793539959513251370	7364368891953175294
29743968054717448193	1.93	4072920019819459351	15878382446559672244
23512868671782387713	1.97	4581142981800872171	22007163001879428406

Table 5.1: Sample generators and corresponding elements of order 8 in $\mathbb{Z}_p^*$ for randomly generated 64-bit generalized safe primes. The average number of steps to find a generator was taken over 100 trials.

polynomial $f \in \mathbb{Z}_{p-1}[x_0]$ has roots is approximately $\frac{2}{3} \cdot \frac{1}{2} = \frac{1}{3}$, which is certainly not negligible, and thus Proposition 5.2.2 is applicable in a practical setting.

5.3.2 Efficiency of Algorithm to Find a Generator

In Lemma 5.2.3, we presented an algorithm to efficiently identify a generator of $\mathbb{Z}_p^*$, as the first step of the attack proposed in Proposition 5.2.4.

We experimentally validated that for the recommended MPPK/DS parameters from [41] (that is, for p a 64-bit generalized safe prime), our algorithm to find a generator is indeed efficient. Table 5.1 demonstrates these experimental results for four distinct values of p; it shows that the average number of iterations needed to identify a generator is approximately two.

5.3.3 An Example of a Forgery Attack

Using the highest recommended security parameters for MPPK/DS (Level V), we were able to successfully produce a forgery using our attack in Proposition 5.2.5. Using $q = 5832672127$ and $p = 25051136033755758593$, we produced both honest and forged signatures for the message $\mu = 10080703823023296604$. As our attack builds upon the work in [30], we modified the code provided by the author to implement our attack.

For an honest signer, we generated the random base $g = 10058362435088020699$, which produced the following signature:

$$\begin{aligned}(A, B, C, D, E) = (&18667305689104477559, 3550055564693041622,\\ &18125679575807015338, 13884945533026050335,\\ &18364756192344131419).\end{aligned}$$

Using our attack, we found an element of order 8 in $\mathbb{Z}_p^*$, $g = 1242402777496096371$ (using the generator $g' = 23358972522162257998$), and solved the system of equations

mod 8 to obtain the signature:

$$(A, B, C, D, E) = (1242402777496096371, 1242402777496096371, \\ 25051136033755758592, 1726782908526691326, \\ 25051136033755758592).$$

We can see that our forged signature is valid. Moreover, it does not contain any 1s, so would pass even the validity checks of the original paper. An alert Verifier may notice the equality of some of the elements of the signature tuple (namely the first and second as well as the third and fifth), which is the only immediate signal that this signature may not be as random as it should be; this is a side effect of choosing 5 values from among the 8 elements of order dividing 8 in $\mathbb{Z}_p^*$. If desired, the forger may avoid such coincidences by working with a base of order 16 or 32, for example. Using the same generator, we found an element in $\mathbb{Z}_p^*$ of order 32, $g = 11385258226623274447$, to generate a different forged signature containing no duplicate elements:

$$(A, B, C, D, E) = (20624269404271038251, 25051136033755758592, \\ 250095367582303466, 4426866629484720342, \\ 1242402777496096371).$$

5.4 Conclusions

We have presented an attack against MPPK/DS that proves that its security does not rely exclusively on the hardness of solving a system of multivariate polynomials over a finite ring. Our attack exploits the very feature that was intended to ensure that MPPK/DS would be secure in the post-quantum context: a variable and unknown base, so that the quantum discrete logarithm algorithm could not be applied by an attacker with quantum capabilities. Note that since the signature elements are, up to scaling, the values of the single-variable secret key polynomials on known inputs (the messages being signed), an attacker who obtains sufficiently many signatures could determine the secret key polynomials by interpolation.

It remains to consider to what extent the scheme can be modified to protect itself against such forgery attacks.

The attack by Guo [30] on MPPK/DS exploited a weakness in the verification algorithm that allowed the process of forging a signature to be reduced from a system of multivariate polynomials to a linear system over $\mathbb{Z}_{p-1}$, which was particularly efficient to solve using the Chinese Remainder Theorem. One might make this attack more difficult by instead choosing p in such a way that the complete factorization of $p-1 = 2^x\tilde{q}$ is not known, rather than specifying p to be a generalized safe prime. Such

modification does not inhibit our attack, however: even without the full factorization of $p-1$, one can create an element of small 2-power order by starting with a random element $h \in \mathbb{Z}_p^*$ and setting $g = h^{(p-1)/2^k}$ for some small k.

A method to thwart our attack would be to modify the scheme more significantly: require the Verifier to additionally compute the order of each signature element, and reject the signature if these were all found to be too small. As discussed in Sections 5.2.1 and 5.3, since this does occur on valid messages, the Signer must then also verify the order of their signature elements, which could be small even if they chose a generator for $\mathbb{Z}_p$ as their base, and in some cases would be required to choose a different private key/public key pair in order to successfully generate a valid signature. This would thus require the Signer and Verifier to share multiple private key/public key pairs, which would increase the storage costs associated to the signature scheme.

Chapter 6

Conclusions and Future Work

In the quickly evolving area of post-quantum cryptography, multivariate cryptography has remained a viable secure option for constructing digital signature schemes.

While it is fundamentally vulnerable to attack, the MPPK/DS signature scheme does highlight the possibility of working with multivariate polynomials over a ring rather than a field. One of the UOV-based submissions to the NIST standardization process in 2023, SNOVA, demonstrated another way in which we could move away from fields by considering UOV over a non-commutative ring of matrices [61].

UOV-based signature schemes tend to suffer from very large public keys, and in comparing TUOV to Biscuit, one can see that the keys for TUOV are much larger than those for Biscuit and that key generation for TUOV is much slower. Indeed, one of the most important lines of work in multivariate cryptography is to attempt to modify the UOV scheme to accommodate smaller key sizes. Some attempts at this have been made, such as the 2023 submission to NIST, MAYO [18], but it remains to be seen whether these modified schemes will hold up under specialized attacks. Similarly, the question remains open as to whether there are specialized attacks on TUOV that exploit its distinct signing process beyond its similarities to UOV.

On the other hand, Biscuit has significantly larger signatures than TUOV, as well as much slower signing and verification algorithms. This is a common feature of MPCitH-based schemes, as the Signer and Verifier must perform a significant number of party computations. As advances in the techniques used in MPCitH, such as [26], continue to improve the efficiency of these types of schemes, it will be interesting to see if the performance of Biscuit can be improved in any significant way. The special polynomials used in Biscuit also present a rich area of open problems. A promising line of inquiry will be to consider if the guessing technique of Theorem 4.2.6 can be improved to create an even more efficient attack on Biscuit.

Appendix A

TUOV Signing Algorithm

This appendix provides more detailed specifications of the TUOV signing algorithm. We continue with the notation introduced in Section 3.3.2.

In order to find a pre-image under the TUOV central map, we must construct a suitable triangular map as in Equation 3.3.1 that will dictate how we "randomly" select our vinegar variables and subsequently solve for our triangular variables.

Since we wish to incorporate an element of randomness into the selection of vinegar variables, as we do for the UOV signature scheme, the (n, m)-OV polynomial f that we use in this triangular map will be a random linear combination of the polynomials f_k of the TUOV central for $k \in [m_1]$. Moreover, the $(n, m, k - m_1)$-TOV polynomials for $k \in [m_2] \setminus [m_1]$ that we use in the triangular map will be f_k from the TUOV central map, with random linear combinations of the (n, m)-OV polynomials added to them. The selection of polynomials for the triangular map can be seen in Steps 3.1 to 3.3 of the pseudocode.

Plugging in a choice $\mathbf{v}$ for the vinegar variables into h gives a linear polynomial in the remaining variables. We require $\mathbf{v}$ so that this polynomial only includes the variable x_{n-m+1}. Then similarly, for all $k \in [m_2] \setminus [m_1]$, we require $\mathbf{v}$ such that only the variables $x_{n-m+1}, \ldots, x_{n-m+k-m_1+1}$ appear in the polynomial formed by h_k. This gives a linear system to solve for $\mathbf{v}$, which appears in Step 3.4 of the pseudocode, and now an iterative procedure lets us solve for the triangular variables $\mathbf{u}^{(1)}$ in Steps 3.5 and 3.6.

Finally, once $\mathbf{v}$ and $\mathbf{u}^{(1)}$ have been found, the resulting system is linear in the oil variables (as in the standard UOV case), and we can solve for $\mathbf{u}^{(2)} \in \mathbb{F}_q^{m-m_2+m_1-1}$

such that

$$\mathcal{F}\left(\begin{bmatrix}\mathbf{v}\\ \mathbf{u}^{(1)}\\ \mathbf{u}^{(2)}\end{bmatrix}\right) = \mathbf{y}.$$

Step 3.8 of the pseudocode demonstrates how we solve for the oil variables.

Recall that for each polynomial f_k in the TUOV central map $\mathcal{F} = (f_1, \ldots, f_m)^\top$, we can write

$$f_k(\mathbf{x}) = \mathbf{x}^\top \mathbf{A}_k \mathbf{x} + \mathbf{b}_k^\top \mathbf{x} + \gamma_k$$

where the upper triangular matrix $\mathbf{A}_k$ has a special form depending on whether it is an OV-polynomial or a TOV-polynomial. We will use the notation from both of these definitions when discussing particular blocks of the matrix $\mathbf{A}_k$.

The pseudocode for the TUOV signing algorithm is as follows.

Sign(params,sk=$(\mathcal{S},\mathcal{F},\mathcal{T}),\mu\in\{0,1\}^*$):

1. Hash the message to obtain $\mathbf{z}\leftarrow\mathrm{Hash}(\mu)$

2. Compute $\mathbf{y}:=\mathcal{S}^{-1}(\mathbf{z})$

3. Until a solution is found, do steps 3.1 through 3.8:

   3.1. Sample $[\lambda_1,\ldots,\lambda_{m_1}]^\top\xleftarrow{\$}\mathbb{F}_q^{m_1}$

   3.2. Set $\tilde{\mathbf{A}}:=\sum_{k=1}^{m_1}\lambda_k\mathbf{A}_k$, $\tilde{\mathbf{b}}:=\sum_{k=1}^{m_1}\lambda_k\mathbf{b}_k$, $[\tilde{\gamma}\ \tilde{y}]:=\sum_{k=1}^{m_1}\lambda_k[\gamma_k\ y_k]$

   3.3. For $l\in[m_2]\setminus[m_1]$ do:

   (i) Set $\tilde{\mathbf{A}}_l:=\mathbf{A}_l$, $\tilde{\mathbf{b}}_l:=\mathbf{b}_l$, $[\tilde{\gamma}_l\ \tilde{y}_l]:=[\gamma_l\ y_l]$

   (ii) For $k\in[m_1]$ do:

   Sample $\lambda_{lk}\xleftarrow{\$}\mathbb{F}_q$

   Set $\tilde{\mathbf{A}}_l=\tilde{\mathbf{A}}_l+\lambda_{lk}\mathbf{A}_k$, $\tilde{\mathbf{b}}_l=\tilde{\mathbf{b}}_l+\lambda_{lk}\mathbf{b}_k$, $[\tilde{\gamma}\ \tilde{y}]=[\tilde{\gamma}\ \tilde{y}]+\lambda_{lk}[\gamma_k\ y_k]$

   3.4. Find a solution $\mathbf{v}\in\mathbb{F}_q^{n-m}$ for the vinegar variables of the following system of equations:

$$\begin{cases}\tilde{\mathbf{A}}^{(2)^\top}\mathbf{v}+\tilde{\mathbf{b}}^{(2)}=\begin{bmatrix}1\\\mathbf{0}_{m-1}\end{bmatrix}\\ \tilde{\mathbf{A}}_l^{(3,l-m_1)^\top}\mathbf{v}+\tilde{\mathbf{b}}_l^{(3,l-m_1)}=\begin{bmatrix}1\\\mathbf{0}_{m-l+m_1-1}\end{bmatrix}\ \text{for } l\in[m_2]\setminus[m_1]\end{cases}$$

   3.5. Set the first triangular variable to $\mathbf{u}^{(1)}:=\tilde{y}-(\mathbf{v}^\top\tilde{\mathbf{A}}^{(1)}+\tilde{\mathbf{b}}^{(1)^\top})\mathbf{v}-\tilde{\gamma}$

   3.6. For $k\in[m_2]\setminus[m_1]$ do:

   (i) Set $d:=k-m_1$

   (ii) Set the next triangular variable

$$u:=\tilde{y}_k-\begin{bmatrix}\mathbf{v}\\\mathbf{u}^{(1)}\end{bmatrix}^\top\begin{bmatrix}\tilde{\mathbf{A}}_k^{(1)}&\tilde{\mathbf{A}}_k^{(2,d)}\\\mathbf{0}&\tilde{\mathbf{A}}_k^{(4,d)}\end{bmatrix}\begin{bmatrix}\mathbf{v}\\\mathbf{u}^{(1)}\end{bmatrix}-\begin{bmatrix}\tilde{\mathbf{b}}_k^{(1)}\\\tilde{\mathbf{b}}_k^{(2,d)}\end{bmatrix}^\top\begin{bmatrix}\mathbf{v}\\\mathbf{u}^{(1)}\end{bmatrix}-\gamma_k$$

   (iii) Update the vector of triangular variables $\mathbf{u}^{(1)}:=\begin{bmatrix}\mathbf{u}^{(1)}\\u\end{bmatrix}$

3.7. Set $d := m_2 - m_1 + 1$

3.8. Solve for the oil variables $\mathbf{u}^{(2)} \in \mathbb{F}_q^{m-m_2+m_1-1}$ using the following system of equations:

$$\begin{cases} (\mathbf{v}^\top \mathbf{A}_k^{(3,d)} + \mathbf{b}_k^{(3,d)^\top})\mathbf{u}^{(2)} \\ = y_k - \gamma_k - \left(\mathbf{v}^\top \begin{bmatrix} \mathbf{A}^{(1)} & \mathbf{A}^{(2,d)} \end{bmatrix} + \begin{bmatrix} \mathbf{b}_k^{(1)} \\ \mathbf{b}_k^{(2,d)} \end{bmatrix}^\top \right) \begin{bmatrix} \mathbf{v} \\ \mathbf{u}^{(1)} \end{bmatrix} & \text{for } k \in [m_1] \\ (\mathbf{v}^\top \mathbf{A}_k^{(3,d)} + \mathbf{b}_k^{(3,d)^\top})\mathbf{u}^{(2)} \\ = y_k - \left(\begin{bmatrix} \mathbf{v} \\ \mathbf{u}^{(1)} \end{bmatrix}^\top \begin{bmatrix} \mathbf{A}_k^{(1)} & \mathbf{A}_k^{(2,d)} \\ \mathbf{0} & \mathbf{A}_k^{(4,d)} \end{bmatrix} + \begin{bmatrix} \mathbf{b}_k^{(1)} \\ \mathbf{b}_k^{(2,d)} \end{bmatrix}^\top \right) \begin{bmatrix} \mathbf{v} \\ \mathbf{u}^{(1)} \end{bmatrix} & \text{for } k \in [m_2] \setminus [m_1] \end{cases}$$

4. Return the signature $\sigma = \mathcal{T}^{-1}\left(\begin{bmatrix} \mathbf{v} \\ \mathbf{u}^{(1)} \\ \mathbf{u}^{(2)} \end{bmatrix} \right)$

Appendix B

Biscuit Algorithms

This appendix provides more detailed specifications of the functions and algorithms used in the Biscuit Digital Signature Scheme.

B.1 Underlying Functions

First, Biscuit uses several hash functions and pseudorandom functions, all of which are instances of the function SHAKE256:

- Commit(salt, e, i, seed), which generates the Signer's commitment to the view of each party in the MPC protocol, as required in a zero-knowledge proof for an MPC protocol;

- H1(salt, μ, σ_1), which will output a value h_1 that is expanded to generate pseudorandom challenges as required for the Fiat Shamir transform. It is required to be a hash function so that the Signer cannot predict or fix what the challenges for the protocol will be. The Verifier checks that the value of h_1 they receive from the Signer is consistent with their computations;

- H2(salt, h_1, σ_1), which will output a value h_2 that is expanded to generate the pseudorandom index that indicates which party's view will be withheld from the Verifier, as required for the Fiat-Shamir transform. Once again, this is a hash function so that its output cannot be predicted or fixed, and the Verifier will check the consistency of h_2 with the information sent from the Signer;

- ChildSeeds(salt, k, j, seed), which is used in the construction of a binary tree of seeds from a single root. Such a tree creates an efficient way to generate the randomness required to create the shares for each party;

- ExpandTape(salt, e, i, seed), which generates a tape, using seeds from the binary tree, from which the random shares for a party in the MPC protocol can be sampled; and

- PRF(input, k), which pseudorandomly generates the initial salt and roots from which each party's randomness (i.e., seeds) will be derived.

During the MPC protocol, the random input shares for the N parties are derived from N seeds. These seeds are derived from a single root using a binary tree structure. The signature scheme contains three functions that make use of this tree structure:

- GetSeeds(root, salt, e, N): the Signer obtains all N seeds from a single root. Each seed will be used to generate the randomness for one of the parties;

- GetPath(root, salt, $e, \bar{i}, N$): the Signer obtains a path in the binary tree so the Verifier can recover all seeds except the $\bar{i}$-th one. This allows the Verifier to recreate the randomness for all parties except for the $\bar{i}$-th one as they verify the consistency of the views revealed to them;

- GetPathSeeds(path, salt, $e, \bar{i}, N$): the Verifier uses the path from the Signer to recover all seeds except the $\bar{i}$-th one.

Three functions are used throughout when the Signer or Verifier needs to generate some values pseudorandomly from a specific set:

- Sample(tape, $\mathcal{E}, n$): pseudorandomly extracts one vector of n values in a set $\mathcal{E}$ from a tape;

- Expand(seed, $k, \mathcal{E}, n$): pseudorandomly extracts k vectors of n values each in a set $\mathcal{E}$ from a pseudorandomly generated tape;

- ExpandCircuit(seedF, $\mathbb{F}_q, n, m$): pseudorandomly generates the coefficients of the polynomial system required for an instance of the PowAff2(q, m, n) problem.

Finally, two functions are used to evaluate the circuit used in the MPC protocol:

- LinearCircuit($\mathbf{s}, \mathbf{t}$, idx, $\mathbf{f}$): allows the Signer or Verifier to evaluate shares of $\mathbf{x}, \mathbf{y}$, and $\mathbf{z}$ as defined in the MPC protocol, given a share of $\mathbf{s}$;

- EvalCircuit($\mathbf{s}, \mathbf{f}$): used during key generation to evaluate $\mathbf{t} = \mathbf{f}(\mathbf{s})$ and $\mathbf{y}$ as defined in the MPC protocol.

B.2 Key Generation Algorithm

1. Sample a seed, seedF, uniformly at random from $\{0,1\}^\lambda$.
2. Generate the function $\mathbf{f}$, as in the PowAff2(q,m,n) problem: $\mathbf{f} \leftarrow$ ExpandCircuit(seedF, $\mathbb{F}_q, n, m$).
3. Sample a seed, seedS, uniformly at random from $\{0,1\}^\lambda$.
4. Generate the secret: $\mathbf{s} \leftarrow$ Expand(seedS, $1, \mathbb{F}_q, n$).
5. Evaluate the circuit to generate $\mathbf{y}, \mathbf{t} \leftarrow$ EvalCircuit$(\mathbf{s}, \mathbf{f})$.
6. The secret key, sk, then consists of seedF, $\mathbf{s}, \mathbf{t}$, and $\mathbf{y}$. The public key, pk, consists of seedF and $\mathbf{t}$.

B.3 Signing Algorithm

Phase 1: the Signer commits to the seeds and views of the parties in the MPC protocol.

1. Sample an element, rnd, uniformly at random from $\{0,1\}^{2\lambda}$ (rnd is set equal to 0 in the case where signing should be deterministic).

2. Generate a salt and roots for each iteration of the MPC protocol: salt, $\text{root}^{(1)}, \ldots, \text{root}^{(\tau)} \leftarrow \text{PRF}(\text{rnd} \| \text{sk} \| \ \mu, 2\lambda + \tau \cdot \lambda)$.

3. For $e \in [\tau]$, i.e., each iteration of the MPC protocol:
   (a) Generate seeds for each party in the MPC protocol: $\text{seed}^{(e,1)}, \ldots, \text{seed}^{(e,N)} \leftarrow \text{GetSeeds}(\text{root}^{(e)}, \text{salt}, e, N)$.

   (b) For $i \in [N]$, i.e., each party in the MPC protocol:
      i. Generate a commitment and tape from the seed: $\text{com}^{(e,i)} \leftarrow \text{Commit}(\text{salt}, e, i, \text{seed}^{(e,i)})$, $\text{tape}^{(e,i)} \leftarrow \text{ExpandTape}(\text{salt}, e, i, \text{seed}^{(e,i)})$.

      ii. Sample the required random shares for the MPC protocol from the tape: $[[\mathbf{s}]]_i^{(e)} \leftarrow \text{Sample}(\text{tape}^{(e,i)}, \mathbb{F}_q, n)$, $[[\mathbf{a}]]_i^{(e)}, [[\mathbf{c}]]_i^{(e)} \leftarrow \text{Sample}(\text{tape}^{(e,i)}, \mathbb{F}_q, m)$.

   (c) Generate correcting values so that the shares of $\mathbf{s}$ and $\mathbf{c}$ can be adjusted to their real values: $\Delta\mathbf{s}^{(e)} \leftarrow \mathbf{s} - \sum_{i=1}^{N} [[\mathbf{s}]]_i^{(e)}$, $\Delta\mathbf{c} \leftarrow \mathbf{y} \cdot \sum_{i=1}^{N} [[\mathbf{a}]]_i^{(e)} - \sum_{i=1}^{N} [[\mathbf{c}]]_i^{(e)}$.

   (d) Correct the first shares of $\mathbf{s}$ and $\mathbf{c}$ so that the sums of all N shares are indeed equal to their real values: $[[\mathbf{s}]]_1^{(e)} \leftarrow [[\mathbf{s}]]_1^{(e)} + \Delta\mathbf{s}^{(e)}$, $[[\mathbf{c}]]_1^{(e)} \leftarrow [[\mathbf{c}]]_1^{(e)} + \Delta\mathbf{c}^{(e)}$.

   (e) For $i \in [N]$, compute shares of $\mathbf{x}, \mathbf{y}$, and $\mathbf{z}$ as in the MPC protocol: $[[\mathbf{x}]]_i^{(e)}, [[\mathbf{y}]]_i^{(e)}, [[\mathbf{z}]]_i^{(e)} \leftarrow \text{LinearCircuit}([[\mathbf{s}]]_i^{(e)}, i, \mathbf{t}, \mathbf{f})$.

4. Generate a committed value σ_1 that consists of $\text{com}^{(e,i)}, \Delta\mathbf{s}^{(e)}$, and $\Delta\mathbf{c}^{(e)}$ for all $i \in [N]$ and $e \in [\tau]$.

Phase 2: the Signer generates challenges for the MPC protocol.

1. Generate a first hashed value: $h_1 \leftarrow \text{H1}(\text{salt}, \mu, \sigma_1)$.

2. Generate the challenges $\varepsilon^{(1)}, \ldots, \varepsilon^{(\tau)} \leftarrow \text{Expand}(h_1, \tau, \mathbb{F}_q, m)$.

Phase 3: the Signer commits to the simulation of the MPC protocol.

1. For $e \in [\tau]$:
   (a) For $i \in [N]$, compute $[[\boldsymbol{\alpha}]]_i^{(e)} \leftarrow [[\mathbf{x}]]_i^{(e)} \cdot \varepsilon^{(e)} + [[\mathbf{a}]]_i^{(e)}$.
   (b) Broadcast the shares to compute $\boldsymbol{\alpha}^{(e)} \leftarrow \sum_{i=1}^{N} [[\boldsymbol{\alpha}]]_i^{(e)}$
   (c) For $i \in [N]$, compute $[[\mathbf{v}]]_i^{(e)} \leftarrow [[\mathbf{y}]]_i^{(e)} \cdot \boldsymbol{\alpha}^{(e)} - [[\mathbf{z}]]_i^{(e)} \cdot \varepsilon^{(e)} - [[\mathbf{c}]]_i^{(e)}$.
2. Generate a committed value σ_2 that consists of $[[\boldsymbol{\alpha}]]_i^{(e)}$ and $[[\mathbf{v}]]_i^{(e)}$ for all $i \in [N]$ and $e \in [\tau]$.

Phase 4: the Signer generates the challenge to open the views of the MPC protocol.

1. Generate a second hashed value: $h_2 \leftarrow \mathrm{H2}(\mathrm{salt}, h_1, \sigma_2)$.
2. Generate the indices which will not be opened to the Verifier for each iteration of the MPC protocol: $\bar{i}_1, \ldots, \bar{i}_\tau \leftarrow \mathrm{Expand}(h_2, \tau, [N], 1)$.

Phase 5: the Signer opens the necessary views of the MPC protocol.

1. For $e \in [\tau]$, generate the path that allows the Verifier to extract all seeds except for $\mathrm{seed}^{(e,\bar{i}_e)}$: $\mathrm{path}^{(e)} \leftarrow \mathrm{GetPath}(\mathrm{root}^{(e)}, \mathrm{salt}, e, \bar{i}_e, N)$.
2. Generate the signature σ that consists of salt, $h_1, h_2, \mathrm{path}^{(e)}, \mathrm{com}^{(e,\bar{i}_e)}, \Delta\mathbf{s}^{(e)}, \Delta\mathbf{c}^{(e)}$, and $[[\boldsymbol{\alpha}]]_{\bar{i}_e}^{(e)}$ for all $e \in [\tau]$.

B.4 Verifying Algorithm

1. Regenerate the challenges for the MPC protocol: $\varepsilon^{(1)}, \ldots, \varepsilon^{(\tau)} \leftarrow \text{Expand}(h_1, \tau, \mathbb{F}_q, m)$, $\bar{i}_1, \ldots, \bar{i}_\tau \leftarrow \text{Expand}(h_2, \tau, [N], 1)$.

2. For $e \in [\tau]$, i.e., each iteration of the MPC protocol:
   (a) Generate all seeds except for $\text{seed}^{(e,\bar{i}_e)}$ from the path provided in the signature: $\text{seed}^{(e,1)}, \ldots, \text{seed}^{(e,N)} \leftarrow \text{GetPathSeeds}(\text{path}^{(e)}, \text{salt}, e, \bar{i}_e, N)$.

   (b) For $i \in [N] \setminus \{\bar{i}_e\}$, i.e., all but the one hidden party in the MPC protocol:
      i. Regenerate the commitment and tape: $\text{com}^{(e,i)} \leftarrow \text{Commit}(\text{salt}, e, i, \text{seed}^{(e,i)})$, $\text{tape}^{(e,i)} \leftarrow \text{ExpandTape}(\text{salt}, e, i, \text{seed}^{(e,i)})$.

      ii. Re-sample the shares from the tape: $[[\mathbf{s}]]_i^{(e)} \leftarrow \text{Sample}(\text{tape}^{(e,i)}, \mathbb{F}_q, n)$, $[[\mathbf{a}]]_i^{(e)}, [[\mathbf{c}]]_i^{(e)} \leftarrow \text{Sample}(\text{tape}^{(e,i)}, \mathbb{F}_q, m)$.

      iii. If $i = 1$, then correct the shares for $\mathbf{s}$ and $\mathbf{c}$ using the correcting values provided in the signature: $[[\mathbf{s}]]_1^{(e)} \leftarrow [[\mathbf{s}]]_1^{(e)} + \Delta\mathbf{s}^{(e)}$, $[[\mathbf{c}]]_1^{(e)} \leftarrow [[\mathbf{c}]]_1^{(e)} + \Delta\mathbf{c}^{(e)}$.

      iv. Re-compute shares of $\mathbf{x}, \mathbf{y}$, and $\mathbf{z}$: $[[\mathbf{x}]]_i^{(e)}, [[\mathbf{y}]]_i^{(e)}, [[\mathbf{z}]]_i^{(e)} \leftarrow \text{LinearCircuit}([[\mathbf{s}]]_i^{(e)}, i, \mathbf{t}, \mathbf{f})$.

      v. Re-compute $[[\boldsymbol{\alpha}]]_i^{(e)} \leftarrow [[\mathbf{x}]]_i^{(e)} \cdot \varepsilon^{(e)} + [[\mathbf{a}]]_i^{(e)}$.

   (c) Using $[[\boldsymbol{\alpha}]]_{\bar{i}_e}^{(e)}$ from the signature, compute $\boldsymbol{\alpha}^{(e)} \leftarrow \sum_{i=1}^{N} [[\boldsymbol{\alpha}]]_i^{(e)}$.

   (d) For $i \in [N] \setminus \{\bar{i}_e\}$, re-compute $[[\mathbf{v}]]_i^{(e)} \leftarrow [[\mathbf{y}]]_i^{(e)} \cdot \boldsymbol{\alpha}^{(e)} - [[\mathbf{z}]]_i^{(e)} \cdot \varepsilon^{(e)} - [[\mathbf{c}]]_i^{(e)}$.

   (e) Set the value for the missing share of $\mathbf{v}$: $[[\mathbf{v}]]_{\bar{i}_e}^{(e)} \leftarrow - \sum_{i=1, i \neq \bar{i}_e}^{N} [[\mathbf{v}]]_i^{(e)}$.

3. Generate the committed value σ_1 consisting of $\text{com}^{(e,i)}, \Delta\mathbf{s}^{(e)}$, and $\Delta\mathbf{c}^{(e)}$ for all $i \in [N]$ and $e \in [\tau]$.

4. Generate a first hashed value $h'_1 \leftarrow \text{H1}(\text{salt}, \mu, \sigma_1)$.

5. Generate the committed value σ_2 consisting of $[[\boldsymbol{\alpha}]]_i^{(e)}$ and $[[\mathbf{v}]]_i^{(e)}$ for all $i \in [N]$ and $e \in [\tau]$.

6. Generate a second hashed value $h'_2 \leftarrow \text{H2}(\text{salt}, h_1, \sigma_2)$.

7. If $h_1 = h'_1$ and $h_2 = h'_2$, then return Valid. Otherwise, return Invalid.

Bibliography

[1] Gora Adj, Luis Rivera-Zamarripa, Javier Verbel, Emanuele Bellini, Stefano Barbero, Andre Esser, Carlo Sanna, and Floyd Zweydinger. MiRitH (MinRank in the Head), 2023. https://csrc.nist.gov/projects/pqc-dig-sig/round-1-additional-signatures.

[2] Nicolas Aragon, Magali Bardet, Loïc Bidoux, Jesús-Javier Chi-Domínguez, Victor Dyseryn, Thibauld Feneuil, Philippe Gaborit, Romaric Neveu, Matthieu Rivain, and Jean-Pierre Tillich. MIRA Specifications, 2023. https://csrc.nist.gov/projects/pqc-dig-sig/round-1-additional-signatures.

[3] Nicolas Aragon, Loïc Bidoux, Jesús-Javier Chi-Domínguez, Thibauld Feneuil, Philippe Gaborit, Romaric Neveu, and Matthieu Rivain. MIRA: a Digital Signature Scheme based on the MinRank problem and the MPC-in-the-Head paradigm. arXIV, Paper 2307.08575, 2023. https://arxiv.org/abs/2307.08575.

[4] Gwénolé Ars, Jean-Charles Faugère, Hideki Imai, Mitsuru Kawazoe, and Makoto Sugita. Comparison Between XL and Gröbner Basis Algorithms. In Pil Joong Lee, editor, *Advances in Cryptology - ASIACRYPT 2004*, pages 338–353, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[5] Michel Barbeau. A Teeny-Tiny Implementation of Multivariate Polynomial Public Key Digital Signature (MPPK/DS), 2022. https://github.com/michelbarbeau/Multivariate-Polynomial-Public-Key-Digital-Signature.

[6] Magali Bardet, Jean-Charles Faugère, and Bruno Salvy. On the complexity of the F5 Gröbner basis algorithm. *Journal of Symbolic Computation*, 70:49–70, 2015.

[7] Carsten Baum and Ariel Nof. Concretely-Efficient Zero-Knowledge Arguments for Arithmetic Circuits and Their Application to Lattice-Based Cryptography. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors,

Public-Key Cryptography – PKC 2020, pages 495–526, Cham, 2020. Springer International Publishing.

[8] Emanuele Bellini, Rusydi H. Makarim, Carlo Sanna, and Javier Verbel. An Estimator for the Hardness of the MQ Problem. In Lejla Batina and Daemen Joan, editors, *Progress in Cryptology - AFRICACRYPT 2022: 13th International Conference on Cryptology in Africa, AFRICACRYPT 2022, Fes, Morocco, July 18-22, 2022, Proceedings*, pages 323–347. Springer Nature Switzerland, 2022.

[9] Luk Bettale, Delaram Kahrobaei, Ludovic Perret, and Javier Verbel. Biscuit: New MPCitH Signature Scheme from Structured Multivariate Polynomials. Cryptology ePrint Archive, Paper 2023/1760, 2023. https://eprint.iacr.org/2023/1760.

[10] Luk Bettale, Delaram Kahrobaei, Ludovic Perret, and Javier Verbel. Biscuit specifications, 2023. https://csrc.nist.gov/projects/pqc-dig-sig/round-1-additional-signatures.

[11] Ward Beullens. Improved Cryptanalysis of UOV and Rainbow. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021*, pages 348–373, Cham, 2021. Springer International Publishing.

[12] Ward Beullens. Breaking Rainbow Takes a Weekend on a Laptop. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022*, pages 464–479, Cham, 2022. Springer Nature Switzerland.

[13] 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: Unbalanced Oil and Vinegar, 2023. https://csrc.nist.gov/projects/pqc-dig-sig/round-1-additional-signatures.

[14] Charles Bouillaguet. Round 1 (additional signatures) official comment: Biscuit. Post on NIST Post-Quantum Cryptography Forum, 2023. https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/sw8NueiNek0/m/2sa_emjABQAJ.

[15] Charles Bouillaguet and Julia Sauvage. Preliminary Cryptanalysis of the Biscuit Signature Scheme. Cryptology ePrint Archive, Paper 2024/148, 2024. https://eprint.iacr.org/2024/148.

[16] Anne Broadbent. Lecture notes for MAT4199 Mathematical Cryptography, Fall 2022.

[17] Anne Broadbent. Lecture notes for MAT5341 Quantum Computing, Fall 2023.

[18] Ward Buellens, Fabio Campos, Sofía Celi, Basil Hess, and Matthias J. Kannwischer. MAYO specifications, 2023. https://csrc.nist.gov/projects/pqc-dig-sig/round-1-additional-signatures.

[19] Alessio Caminata and Elisa Gorla. Solving degree, last fall degree, and related invariants. *Journal of Symbolic Computation*, 114:322–335, 2023.

[20] David A. Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms*. Springer International Publishing Switzerland, 2018.

[21] Jintai Ding, Ming-Shing Chen, Matthias Kannwischer, Jacques Patarin, Albrecht Petzoldt, Dieter Schmidt, and Bo-Yin Yang. Rainbow - Algorithm Specification and Documentation, 2020. https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[22] Jintai Ding, Boru Gong, Hao Guo, Xiaoou He, Yi Jin, Yuansheng Pan, Dieter Schmidt, Chengdong Tao, Danli Xie, Bo-Yin Yang, and Ziyu Zhao. TUOV: Triangular Unbalanced Oil and Vinegar, 2023. https://csrc.nist.gov/projects/pqc-dig-sig/round-1-additional-signatures.

[23] Jintai Ding and Dieter Schmidt. *Solving Degree and Degree of Regularity for Polynomial Systems over a Finite Fields*, pages 34–49. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[24] Jean-Charles Faugére. A new efficient algorithm for computing Gröbner bases (F4). *Journal of Pure and Applied Algebra*, 139(1):61–88, 1999.

[25] Thibauld Feneuil and Matthieu Rivain. MQOM: MQ on my Mind: Algorithm Specifications and Supporting Documentation, 2023. https://csrc.nist.gov/projects/pqc-dig-sig/round-1-additional-signatures.

[26] Thibauld Feneuil and Matthieu Rivain. Threshold Computation in the Head: Improved Framework for Post-Quantum Signatures and Zero-Knowledge Arguments. Cryptology ePrint Archive, Paper 2023/1573, 2023. https://eprint.iacr.org/2023/1573.

[27] 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*, pages 186–194. Springer, 1986.

[28] Hiroki Furue and Momonari Kudo. Polynomial XL: A Variant of the XL Algorithm Using Macaulay Matrices over Polynomial Rings. Cryptology ePrint Archive, Paper 2021/1609, 2021. https://eprint.iacr.org/2021/1609.

[29] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to Construct Random Functions. *Journal of the Association for Computing Machinery*, 33(4):792–807, 1986.

[30] Hao Guo. An algebraic attack for forging signatures of MPPK/DS. Cryptology ePrint Archive, Paper 2023/453, 2023. https://eprint.iacr.org/2023/453.

[31] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. *An Introduction to Mathematical Cryptography*. Springer New York, NY, 2008.

[32] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A Pseudorandom Generator from any One-Way Function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[33] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 21–30. ACM, 2007.

[34] Daniel Kales and Greg Zaverucha. An Attack on Some Signature Schemes Constructed From Five-Pass Identification Schemes. Cryptology ePrint Archive, Paper 2020/837, 2020. https://eprint.iacr.org/2020/837.

[35] Daniel Kales and Greg Zaverucha. Efficient Lifting for Shorter Zero-Knowledge Proofs and Post-Quantum Signatures. Cryptology ePrint Archive, Paper 2022/588, 2022. https://eprint.iacr.org/2022/588.

[36] Burt Kaliski. Pseudorandom function. In Henk C. A. van Tilborg, editor, *Encyclopedia of Cryptography and Security*, pages 485–485. Springer US, Boston, MA, 2005.

[37] Burt Kaliski. O-Notation. In Henk C. A. van Tilborg and Sushil Jajodia, editors, *Encyclopedia of Cryptography and Security*, pages 881–882. Springer US, Boston, MA, 2011.

[38] Aviad Kipnis, Jacques Patarin, and Louis Goubin. Unbalanced Oil and Vinegar Signature Schemes. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, pages 206–222, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[39] Aviad Kipnis and Adi Shamir. Cryptanalysis of the oil and vinegar signature scheme. In Hugo Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, pages 257–266, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[40] Arnold Knopfmacher and John Knopfmacher. Counting irreducible factors of polynomials over a finite field. *Discrete Mathematics*, 112(1):103–118, 1993.

[41] Randy Kuang, Maria Perepechaenko, and Michel Barbeau. A new quantum-safe multivariate polynomial public key digital signature algorithm. *Scientific Reports*, 2022.

[42] Randy Kuang, Maria Perepechaenko, and Michel Barbeau. A new post-quantum multivariate polynomial public key encapsulation algorithm. *Quantum Inf. Process.*, 21(10):Paper No. 360, 25, 2022.

[43] Yehuda Lindell and Ariel Nof. A Framework for Constructing Fast MPC over Arithmetic Circuits with Malicious Adversaries and an Honest-Majority. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 259–276, New York, NY, USA, 2017. Association for Computing Machinery.

[44] Laura Maddison and Monica Nevins. A Classically Efficient Forgery of MPPK/DS Signatures. *La Matematica*, 2024. https://doi.org/10.1007/s44007-024-00095-0.

[45] Tsutomu Matsumoto and Hideki Imai. Public Quadratic Polynominal-Tuples for Efficient Signature-Verification and Message-Encryption. In *Advances in Cryptology - EUROCRYPT '88, Workshop on the Theory and Application of of Cryptographic Techniques, Davos, Switzerland, May 25-27, 1988, Proceedings*, volume 330 of *Lecture Notes in Computer Science*, pages 419–453. Springer, 1988.

[46] J. Milnor and D. Husemoller. *Symmetric Bilinear Forms*. Ergebnisse der Mathematik und ihrer Grenzgebiete. 2. Folge. Springer Berlin Heidelberg, 2013.

[47] Monica Nevins. Lecture notes for MAT3143 Rings and Modules, Fall 2021.

[48] National Institute of Standards and Technology. Post-Quantum Cryptography: Digital Signature Schemes. https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures. Accessed 2023-09-24.

[49] National Institute of Standards and Technology. Post-Quantum Cryptography Security (Evaluation Criteria). https://csrc.nist.gov/Projects/post-quantum-cryptography. Accessed 2024-04-02.

[50] National Institute of Standards and Technology. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. Technical Report Federal Information Processing Standards Publications (FIPS PUBS) 202, U.S. Department of Commerce, Washington, D.C., 2015.

[51] National Institute of Standards and Technology. Digital Signature Standard. Technical Report Federal Information Processing Standards Publications (FIPS PUBS) 186-5, U.S. Department of Commerce, Washington, D.C., 2023.

[52] National Institute of Standards and Technology. Post-Quantum Cryptography: Digital Signature Schemes. Standardization of Additional Digital Signature Schemes. https://csrc.nist.gov/Projects/pqc-dig-sig/standardization, 2023. Accessed: August 2023.

[53] Jacques Patarin. Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms. In Ueli Maurer, editor, *Advances in Cryptology — EUROCRYPT '96*, pages 33–48, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[54] Jacques Patarin. The Oil and Vinegar signature scheme, 1997. Presented at the Dagstuhl Workshop on Cryptography.

[55] Peter Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Journal on Computing*, 114(5):1484–1509, 1997.

[56] Arne Storjohann and Thom Mulders. Fast Algorithms for Linear Algebra Modulo N. In Gianfranco Bilardi, Giuseppe F. Italiano, Andrea Pietracaprina, and Geppino Pucci, editors, *Algorithms — ESA' 98*, pages 139–150, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[57] Salil Vadhan. Computational complexity. In Henk C. A. van Tilborg and Sushil Jajodia, editors, *Encyclopedia of Cryptography and Security*, pages 235–240. Springer US, Boston, MA, 2011.

[58] Duc Trung Van. Fröberg's Conjecture and the initial ideal of generic sequences. arXIV, Paper 1803.04997, 2018. https://arxiv.org/abs/1803.04997.

[59] Henk C. A. van Tilborg and Sushil Jajodia, editors. *Encyclopedia of Cryptography and Security*. Springer New York, NY, 2014.

[60] Marion Videau. Distinguishing attacks. In Henk C. A. van Tilborg and Sushil Jajodia, editors, *Encyclopedia of Cryptography and Security*, pages 358–359. Springer US, Boston, MA, 2011.

[61] Lih-Chung Wang, Po-En Tseng, Yen-Liang Kuan, and Chun-Yen Chou. A Simple Noncommutative UOV Scheme. Cryptology ePrint Archive, Paper 2022/1742, 2022. https://eprint.iacr.org/2022/1742.

[62] Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. New Bounds for Matrix Multiplication: from Alpha to Omega. arXIV, Paper 2307.07970, 2023. https://arxiv.org/abs/2307.07970.

[63] Takanori Yasuda, Xavier Dahan, Yun-Ju Huang, Tsuyoshi Takagi, and Kouichi Sakurai. MQ Challenge: Hardness Evaluation of Solving Multivariate Quadratic Problems. *IACR Cryptol. ePrint Arch.*, page 275, 2015.

[64] Greg Zaverucha, Melissa Chase, David Derler, Steven Goldfeder, Daniel Kales, Jonathan Katz, Vladimir Kolesnikov, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Xiao Wang. Picnic specifications, 2020. https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.