Contents

Appendices

Abstract

Multivariate cryptography (MVQC) is currently one of the most promising families of seemingly quantum-safe cryptographical schemes. However, it often suffers from extremely large key sizes or ad-hoc security assumptions. In this thesis we deal with a MVQC scheme, *Unbalanced Oil and Vinegar* (UOV), leveraging a reformulation of it recently given by Beullens ([6]) with the aim of exploring how much one can reduce the size of secret keys while retaining practicality. Moreover, we focus on two simplifications commonly applied to UOV — these are justified simply by seeing that one can apply the same hardness assumptions that buttress standard UOV. We show that (with some concessions), it can be proven that they retain security just from the assumption that standard UOV is secure.

Resumen

La *criptografía multivariante* (MVQC) es, actualmente, una de las familias de esquemas criptográficos más prometedoras en el ámbito post-cuántico. Aun así, con frecuencia sufre de llaves de tamaño excesivamente grande o suposiciones de seguridad hechas a medida. En esta tesis, tratamos un esquema de MVQC, *Unbalanced Oil and Vinegar* (UOV), utilizando una reformulación de éste dada recientemente por Beullens ([6]), con el objetivo de explorar cuánto es posible reducir el tamaño de las llaves privadas sin comprometer la practicalidad. También nos centramos en dos simplificaciones comúnmente aplicadas a UOV — habitualmente estas se justifican viendo que uno puede aplicarles las suposiciones de seguridad que forman la base de UOV. Demostramos que (con algunas concesiones), se puede demostrar directamente que su seguridad sigue de la seguridad de UOV tradicional.

Resum

La *criptografia multivariant* (MVQC) és, actualment, una de les famílies d'esquemes criptogràfics més prometedores en l'àmbit postquàntic. Tanmateix, amb freqüència pateix de claus d'una mida excessiva o suposicions de seguretat fetes a mida. En aquesta tesi, tractarem un esquema de MVQC, el *Unbalanced Oil and Vinegar* (UOV), emprant una reformulació d'aquest donada recentment per Beullens ([6]), amb l'objectiu d'explorar fins a quin punt és possible reduir la mida de les claus privades sense afectar a la practicalitat. També ens centrem en dues simplificacions freqüentment aplicades a UOV — habitualment aquestes es justifiquen veient que un pot aplicar-hi les suposicions de seguretat que formen la base de UOV. Demostrem que (amb algunes concessions), es pot demostrar directament que la seva seguretat segueix de la seguretat de UOV tradicional.

Chapter 1

Introduction

This text deals with the *Unbalanced Oil and Vinegar* (UOV) scheme from Multivariate cryptography (MVQC), a presumed quantum-safe cryptographical family based primarily on the assumed hardness of solving multivariate quadratic systems, and the assumed possibility to effectively hide trapdoors within these systems. As is common in MVQC, UOV suffers from exceedingly large public key sizes, as these have to hold a description of a multivariate quadratic map with sufficiently large numbers of variables and components. Numerous schemes derived from UOV have been proposed to mitigate this issue, from the newly broken NIST finalist *Rainbow* ([8]), to the recently proposed *MAYO* ([7]). In this text, we focus instead on the *secret* key of UOV, leveraging a reformulation of it given by Beullens in [6] to remove a significant amount of redundant information from these keys — allowing us to shave off up to a third of the key's size for safe parameters.

Being more specific, UOV is composed of two maps $\mathcal{F}$ and $\mathcal{T}$ — $\mathcal{F}$ being a multivariate quadratic of a form that makes it readily invertible, and $\mathcal{T}$ being a linear masking map such that $\mathcal{F} \circ \mathcal{T}$ looks like any random multivariate quadratic. Two simplifications are usually applied to UOV — namely, that $\mathcal{F}$ and $\mathcal{T}$ can be assumed to be (quadratically and linearly, respectively) homogenous. These simplifications are commonly justified by the fact that the security assumptions that buttress UOV also support the simplified versions — however, we verify that in most settings these assumptions are not necessary, and it can be proven directly that the security of these simplified versions follows from that of standard UOV.

The text is structured as follows:

Chapter 2 deals with preliminary theory. It contains a very minimal introduction to coding theory; as well as a more fleshed section about cryptography, meant to both serve as a brief introduction to the field and lay out the concepts that will be used during the text.

Chapter 3 deals with multivariate cryptography. In particular, §3.1 is about the (assumed-hard) problems underlying multivariate cryptography, giving a number of definitions and basic results necessary to deal with them for their application in MVQC. §3.2, on the other

hand, gives the two main definitions of UOV, and consists primarily of results regarding the security of simplified versions of UOV with respect to the security of standard UOV.

Chapter 4 attempts to find an optimal encoding for subspaces of finite fields, giving both theoretical bounds for the size of these encodings and then showing that these bounds can be met by efficiently computable codes. Moreover, it discusses how this can be applied to UOV.

1.1 Notation & convention

Linear algebra. All vectors are column vectors unless otherwise specified. The transpose of a matrix M is denoted $M^\top$. The orthogonal complement of a subspace V is denoted $V^\perp$. All vectors and matrices are 1-indexed unless otherwise specified. If $\mathbf{v}$ is a vector, $diag(\mathbf{v})$ denotes the matrix with $\mathbf{v}$ along the diagonal and zeros elsewhere. Conversely, if M is a matrix, $diag(M)$ denotes the vector that has as its elements the diagonal of M. $\mathcal{M}(\mathbb{K}, n, m)$ denotes the set of $n \times m$ matrices over a field $\mathbb{K}$, and $GL(\mathbb{K}, n)$ the set of invertible $n \times n$ matrices over this same field. Id denotes the identity matrix and $\vec{0}$ denotes the zero vector, both with dimension depending on the context. $Gr(k, V)$ denotes the k-th grassmannian of V, the set of k-dimensional subspaces of V. Unless otherwise specified, "subspace" refers to linear subspaces, although we will also talk about affine and projective subspaces at times.

Arithmetic. The degree of the zero polynomial is $-\infty$. $\mathbb{F}_q$ denotes the finite field of cardinality q. $\mathbb{F}_q^*$ denotes the set of invertible (i.e. nonzero) elements of $\mathbb{F}_q$.

Other. The smallest number in $\mathbb{N}$ is 1. $[a, b]$ denotes the set of positive integers between a and b, inclusive. $[n]$ is defined to be $[1, n]$. If $\mathbf{u}$ and $\mathbf{v}$ are vectors, $\mathbf{u}\|\mathbf{v}$ denotes their concatenation. We occasionally use "w.r.t." and "s.t." to abbreviate "with respect to" and "such that," respectively.

Chapter 2

Preliminary Theory

2.1 Noiseless coding

Coding deals with representation of objects from a finite set S through a finite alphabet Σ, often with the intent of finding representations that are resistant to noise (i.e. probably still decipherable if some bounded amount of symbols are corrupted), or that minimize expected word length w.r.t. some probability distribution on S. This will not be the case for us, so we limit ourselves to the necessary elementary notions.

We start by defining $T^* = \cup_{i\in\mathbb{N}} T^i$. T^* can be endowed with the operation of concatenation (such that T^* becomes the free semigroup over T), which we will denote by juxtaposition. With this, a code (or representation) of S with alphabet T is an injective map $\phi\colon S \to T^*$. Often we will denote $\phi(s)$ by $\langle s\rangle$ — we will call this the encoding (or representation) of s.

Because S^* is free over S, ϕ extends uniquely to a morphism $\overline{\phi}\colon S^* \to T^*$. We say that the coding ϕ is **uniquely decodable** if $\overline{\phi}$ is a monomorphism. We will be using a more combinatoric condition for unique decodability: a code $\phi\colon S \to T^*$ is a **prefix code** if there are no two s_1, s_2 from S such that $\phi(s_1)$ is a prefix of $\phi(s_2)$, that is, $\phi(s_2) = \phi(s_1)\omega$, for some $\omega \in T^*$. Any prefix code is uniquely decodable. We state two more useful results:

Theorem 2.1. *(Kraft's inequality) Let ϕ be a code of S with alphabet T. Denote $q = \#T$, and for any $s \in S$, let ℓ_s denote the length of $\phi(s)$. Then, if ϕ is a prefix code, it must hold that*

$$\sum_{s\in S} q^{-\ell_s} \leq 1.$$

The two claims we have made so far are rather intuitive — nonetheless, proofs can be found in [21][1]. This short paper also provides a less obvious generalization:

Theorem 2.2. *(McMillan's inequality) Let ϕ be a code of S with alphabet T. Denote $q = \#T$, and for any $s \in S$, let ℓ_s denote the length of $\phi(s)$. If ϕ is a uniquely decodable code, it must hold that*

$$\sum_{s\in S} q^{-\ell_s} \leq 1.$$

[1] McMillan attributes Theorem 2.1 to an oral remark by a colleague — but in fact, this result had been published by Kraft in [19], their MSc thesis.

2.2 Introduction to Cryptography

Cryptography is, at least in a vague sense, the study of the secure transferral of knowledge in presence of adversarial agents. The practice of cryptography often consists of the design of protocols to be followed by cooperating parties, such that this secure transferral of knowledge can be achieved. The mathematical specifications of these protocols distill exactly what one means by "secure" and "transferral of knowledge"— we give an example:

Definition 2.3. *A **Shannon Cipher** with message space $\mathcal{M}$, ciphertext space $\mathcal{C}$, and key space $\mathcal{K}$, is a tuple $\mathcal{E} = (E, D)$, with $E\colon \mathcal{M} \times \mathcal{K} \to \mathcal{C}$ and $D\colon \mathcal{C} \times \mathcal{K} \to \mathcal{M}$, satisfying the **correctness property**, i.e. that $D(E(m,k),k) = m$ holds for any $k \in \mathcal{K}$, $m \in \mathcal{M}$.*

We will say that $\mathcal{E}$ is defined over $(\mathcal{M}, \mathcal{C}, \mathcal{K})$, and we will call E, D the encryption and decryption functions, respectively. The correctness property ensures that $\mathcal{E}$ can be used to transfer information: two parties, say Alice and Bob, share knowledge of a key $k \in \mathcal{K}$, and Alice wants to send some $m \in \mathcal{M}$ to Bob — "in presence of adversarial agents", meaning that whatever she sends may be eavesdropped on. The approach would be to send $c := E(m,k)$, such that Bob can recover $m = D(c,k)$. In this context, the "transferral of knowledge" is the choice of $m \in \mathcal{M}$, and that this transferral is "secure" should mean, informally, that an eavesdropper without knowledge of k should not be able to gain knowledge about m just from hearing c.

There are a number of ways to interpret this definition of "secure". The gold standard (for single-use keys) is aptly known as *perfect security*:

Definition 2.4. *A Shannon cipher (E, D) is **perfectly secure** iff, letting k be a random element with uniform[2] distribution over $\mathcal{K}$ and $m \in \mathcal{M}$, then $E(m, \mathsf{k})$ is independent of the choice of m.*

The reason that this is the strongest notion of security in this context is somewhat akin to the reason that rock-paper-scissors has no winning strategy — indeed, as long as your opponent truly picks their move randomly, then the symmetry of the game ensures that you can have no knowledge of the outcome. Similarly, as long as Alice and Bob picked $k \in \mathcal{K}$ at random, then an eavesdropper cannot gain any information about m from hearing c, as c comes from the distribution $E(m, \mathsf{k})$, which is independent of m.

Remark. For the past three paragraphs we have assumed that not only Alice and Bob, but also any adversarial agents, have complete knowledge of the definition of $\mathcal{E}$. This is common practice, and often known as **Kerckhoff's principle** (or **Shannon's maxim**): the choice of k should be the only thing that one needs to keep secret when using $\mathcal{E}$.

Both of these definitions (that of Shannon Cipher and Perfect Security) come from Claude Shannon's seminal paper [25], *Communication Theory of Secrecy Systems*[3]. This paper also contains a very important result about perfect security:

[2] the usage of "uniform", and treating $E(m, K)$ as random elements (i.e. $E(m, \cdot)$ as measurable) rest on the unspoken assumption that $\mathcal{K}$ is finite. In practice, this is always the case.

[3] notably, much of the same material appears in an earlier report, classified at time of publication: *A Mathematical Theory of Cryptography* — this report also predates Shannon's other seminal paper, *A Mathematical Theory of Communication*.

Theorem 2.5. *(Shannon's theorem of perfect secrecy). Let $\mathcal{E}$ be a Shannon cipher with perfect security defined over $(\mathcal{M}, \mathcal{C}, \mathcal{K})$. Then, $\#\mathcal{K} \geq \#\mathcal{M}$.*

This is rather bad news. If $\#\mathcal{K} \geq \#\mathcal{M}$, then $\log_2 \#\mathcal{K} \geq \log_2 \#\mathcal{M}$, meaning the encryption of any file will incur the creation of a separate item, a key, which will take up at least as much space as the file itself. This isn't extremely outlandish for sensitive communication, but if one wishes to have, say, an encrypted video-conference, doubling the amount of data that the computer has to process doesn't seem like good practice.

So we look for ways to relax our definition of perfect security — by looking for ways in which the definition of perfect security is a bit *too* paranoid. A possibility is this: perfect security asks that $E(m, \mathsf{k})$ be independent of m, i.e. that $E(m_1, \mathsf{k}) = E(m_2, \mathsf{k})$ for any m_1, m_2 from $\mathcal{M}$. However, since our adversaries do not often have the capacity of evaluating arbitrary mathematical functions, we might simply ask that $E(m_1, \mathsf{k})$ is indistinguishable from $E(m_2, \mathsf{k})$ *in practice*, i.e. that no efficient algorithm can tell the two distributions apart.

Until now, we have dealt with Shannon Ciphers, which are purely functional objects. The prior paragraph introduces a new computational flavor to the concept of ciphers, and we call the result a **computational cipher** — essentially, a Shannon Cipher that comes with efficient algorithms for computing E and D, as well as sampling keys from $\mathcal{K}$.

2.2.1 Relevant elements of cryptography

In this section, we aim to introduce the less elementary concepts of cryptography that will be necessary during this text — though, evidently, the field of Cryptography is richer than what can be condensed in a two-page summary[4]. As seen before with the introduction of computational ciphers, we will need to deal with some concepts rooted in elementary complexity theory. For the sake of brevity, we will assume the reader is familiar with the concepts of Turing Machines (TMs), efficiency, computational problems, and hardness. We refer to [2] for a formal treatment of these, and [9, §2.3] for their role in cryptography — here we simply outline the conventions we will be following.

Conventions regarding computational notions

Algorithms. We will often refer to TMs as simply **algorithms** and deal with them through a layer of abstraction. This is primarily in the sense that we will write their input/output as mathematical objects — without taking the time to specify an encoding into the TM's alphabet, or how this encoding is used to deal with the aforementioned objects (e.g. a TM may be defined as taking a "polynomial" as an input and then computing its "derivative" — it is not hard to imagine how one might actually formalize these two computational operations, but doing so is orthogonal to the topics of interest in this text).

Input/output. We may assume that a TM outputs/requests input at different times during execution (e.g. a stateful random number generator), as decided by execution itself. In

[4] the interested reader is encouraged to read [9], particularly the first chapter.

theory, this can be modeled by the TM entering a pseudo-halting state when it is waiting for additional input. At times we will have a TM `A` that takes a single input and return a single output (either of these may be a tuple of several objects) — in this case, we use $\texttt{A}(x)$ to denote the output of `A` on input x — if `A` is probabilistic, then this is a random variable.

Nondeterminism. Unless otherwise stated, we will always be talking about probabilistic algorithms. At times, probabilistic or non-deterministic algorithms may be made deterministic by feeding the randomness through an additional tape, known as a **random tape**.

Signature schemes

The primary cryptographical construction we will be dealing with in the following chapters is the so-called *Unbalanced Oil and Vinegar* scheme. In essence, the main object of interest underlying this scheme is a kind of map that is hard to invert unless one knows certain secret information. In this sense, this puts it rather close to what are usually known as *one-way trapdoor function* schemes — however, in practice it is almost always used as a *signature scheme*, and so this is the lens through which we will be looking at it:

Definition 2.6. *A **signature scheme** with message space $\mathcal{M}$, key spaces $\mathcal{K}_p$ and $\mathcal{K}_s$, and signature space $\mathcal{S}$, consists of three efficient algorithms:*

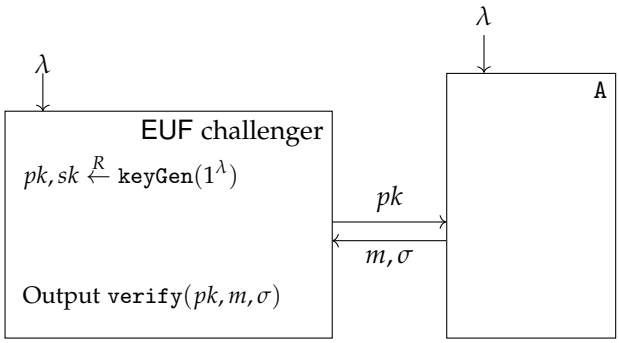
- `keyGen`*, taking a unary input 1^λ (λ being the* security parameter*), and returning a tuple $(pk, sk) \in \mathcal{K}_p \times \mathcal{K}_s$ of a public key and a private key.*

- `sign`*, with input a secret key $sk \in \mathcal{K}_s$ and message $m \in S$, and returning a signature $\sigma \in \mathcal{S}$. For convenience, `sign` may also take a public key pk as input.*

- `verify`*, taking as input a public key $pk \in \mathcal{K}_p$, a message $m \in \mathcal{M}$, and a signature $\sigma \in \mathcal{S}$, and outputting either 1 or 0.*

*satisfying the **correctness property**, i.e. that for any pk, sk returned by* `keyGen` *and any $m \in \mathcal{M}$, we have that* $\texttt{verify}(pk, m, \texttt{sign}(sk, m)) = 1$.

The correctness property essentially states that `verify` believes any legitimate signature (i.e. one produced by `sign` with the adequate secret key). For signature schemes, security will essentially be the converse of correctness, i.e. that `verify` should not believe any signature that has not been legitimately generated with knowledge of sk — i.e. that no efficient algorithm `A` can, given only 1^λ and pk, have a chance of finding m and σ such that $\texttt{verify}(pk, m, \sigma) = 1$. To be precise, since this allows `A` to pick which m it wants to forge a signature σ for, no such `A` existing would mean that the scheme is *existentially unforgeable (EUF)*. This concept is more often defined in the language of cryptographical games:



The diagram above may be interpreted as an algorithm, taking $\lambda \in \mathbb{N}$ as an input, internally running the challenger, A, and the message passing between them, and outputting whatever the challenger outputs. Let $\mathcal{E}$ be the signature scheme defined by keyGen, verify, and sign — then, we will call this algorithm $\mathsf{EUFchallenge}_{\mathcal{E}}[\mathtt{A}]$, and the **EUF advantage** of A over $\mathcal{E}$ will be

$$EUFadv_{\mathcal{E}}(\mathtt{A})\colon \lambda \mapsto Pr(\mathsf{EUFchallenge}_{\mathcal{E}}[\mathtt{A}](\lambda) = 1),$$

wherein the Pr refers to the probability space induced by the random choices of A and the challenger. Intuitively, the security parameter λ represents a choice of how difficult we want the game above to be for A — in practice, this always incurs a tradeoff, where a higher security parameter also means larger keys and longer running times[5].

We say that A **breaks the** EUF **security of** $\mathcal{E}$ if $EUFadv_{\mathcal{E}}(\mathtt{A})(\lambda)$ is a *non-negligible* function of λ, i.e. it is asymptotically greater than the inverse of some polynomial. We say that $\mathcal{E}$ has EUF **security** if no efficient algorithm A can break its EUF security[6].

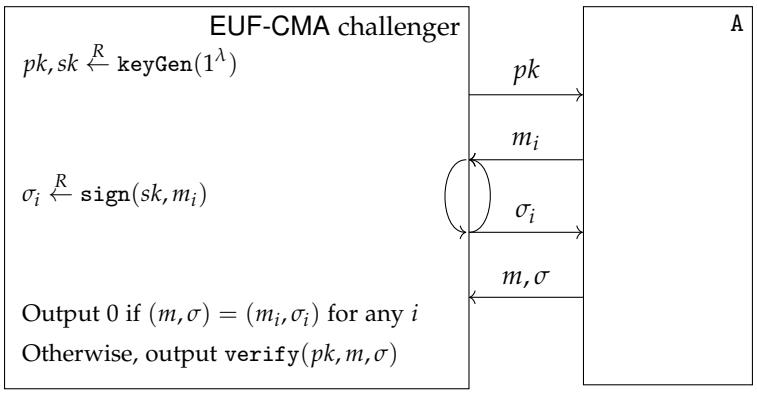
Proving $EUFadv_{\mathcal{E}}(\mathtt{A})(\lambda)$ to be negligible for *any* algorithm A seems rather daunting. In practice, this is nearly always achieved through *reductions*: showing that, if such an A existed, then another algorithm B (typically built as a wrapper around A) must also exist and efficiently solve some problem known or thought to be hard. This is analogous to the notion of computational reduction, but, crucially, here we do not need A or B to always achieve their goal — they need only work a non-negligible portion of the time.

It should be noted that there are other definitions of security for signature schemes. For instance, there exist other notions of forgeability — such as *universal unforgeability* (UUF), which means that no algorithm A should be able to succeed in a game where the challenger picks the message m that they have to forge a signature for. More importantly for us, a scheme can be shown to be secure against certain types of "attacks" — for instance, $\mathcal{E}$ is EUF-CMA if, informally, it retains EUF even when A is allowed to query for signatures of any message it chooses (so long as it outputs a forgery that is not among the queries).

[5]Note that keyGen must be efficient, i.e. with runtime bounded by a polynomial in $|1^{\lambda}| = \lambda$. By extension this also bounds the size of its output, and thus sign and verify also have runtime bounded by a polynomial in λ — so indeed, the security parameter controls how costly it is to run our scheme.

[6]Note that the negligibility here is a somewhat arbitrary requirement — in fact, all we are interested in is that for some *practical* value of λ, the advantage of any A is sufficiently low that $\mathcal{E}$ is secure in practice.

This can be succintly expressed by modifying the previous game (note that from this point onward, λ being an input will be implicit):



Analogously to before, we would define $\mathsf{EUF\text{-}CMAchallenge}_{\mathcal{E}}[\mathtt{A}]$, and $EUF\text{-}CMAadv_{\mathcal{E}}(\mathtt{A})$, and from this define EUF-CMA security.

Cryptohashes and the Random Oracle Model

As we mentioned earlier, the signature schemes that we will be dealing with are derived from constructions that are closer to *one-way functions*. This is done with an approach strongly analogous to what is known as the *Full Domain Hash* — to properly explain this, we have to introduce the notion of cryptopgraphic hash functions, or cryptohashes.

Giving a proper definition of these is not particularly enlightening, and, for reasons that will become clear shortly, we in fact do not need to properly define them at all — for this, we refer to [9, §8.1]. Informally, a cryptohash is a function for which it is (very) difficult to find collisions, i.e. different points that map to the same image. Cryptohashes are usually thought of, or at least used as, functions that seemingly have no internal structure, and thus are used as stand-inds for random functions[7]. This explains the frequent usage of the *Random Oracle Model (ROM)* — essentially, showing that some scheme that uses a hash H is secure if H is replaced by a bona-fide random function. We will be using the ROM in some reductions later on.

Informally, then, a FDH approach to constructing a signature scheme from a one-way function f (picked by `keyGen`) would sign a message m by computing $H(m)$ and finding σ such that $f(\sigma) = H(m)$ (note the signing requires knowing how to invert f), and the verification would simply involve checking that, indeed $f(\sigma) = H(m)$. It is rather easy to convince oneself that, in the ROM (i.e. assuming that H behaves like a random function), then the one-wayness of f usually begets existential unforgeability in the resulting scheme.

[7]The function itself is deterministic — "random function" here means "a random pick from the set of all relevant functions"

2.2.2 Post-quantum cryptography

Earlier in §2.2¶12, we shifted our focus from adversaries capable of computing any well-defined function, to adversaries capable of computing any efficiently computable function. Though we did not define this, what is usually meant by a function being *computable* is that it can be computed by a *Probabilistic Turing Machine* — this appears tantamount to assuming that adversaries have no technology better than Probabilistic Turing Machines. This might seem dangerous at first — what if a real-life adversary has the capability of running some other computational model?

In principle, we are protected by the **Church-Turing Hypothesis** — that all models of computation are equivalent (i.e. they can compute the same things). More precisely, since we are interested in what is *efficiently computable*, we are protected by the **Strong Church-Turing Hypothesis** — that all models of computation can simulate each other with polynomial overhead (therefore, they can efficiently compute the same things).

The advent of (the computational model of) quantum computation has put this latter version of the hypothesis in a dubious position. We cannot say for sure that the strong hypothesis is false — however, it is a definite fact that problems which are believed to be computationally hard (e.g. prime factoring), hard enough that they buttress widely-used cryptosystems (e.g. RSA), have been found to be not that hard for quantum computers.

This poses a more urgent threat than one might assume at first. Indeed, quantum computers powerful enough to attack RSA are nowhere close to existing — however, nobody is stopping adversarial agents from eavesdropping on RSA-encrypted conversations, waiting until the advent of reasonably powerful quantum computers (however many years from now), and using those computers to succesfully break the encryption of those conversations. A non-insignificant portion of the sensitive information being exchanged nowadays will certainly still be sensitive in 50 years — so, if one believes that reasonably powerful quantum computing is less than 50 years away, they should be worried about whether the cryptosystems that they're using *right now* are vulnerable to quantum adversaries.

So, like before, this forces a redefinition of safety for cryptographical protocols — we now have to find schemes that cannot be attacked by efficient *quantum* adversaries, meaning that they are based on problems which are hard for quantum computers. Several approaches exist to this — for a detailed rundown, we refer to [5]. In this text, we will be dealing with one of these approaches — that of *multivariate cryptography.*

Chapter 3

Multivariate Cryptography

Multivariate cryptography is a family of asymmetric cryptographic schemes based on trapdoor functions taking the form of multivariate quadratic maps. These maps are generally difficult to invert, but have some kind of structure baked into them which makes efficient inversion possible if one knows some secret information describing this structure. Usually, this structure is readily visible in some easy to invert "central map", and it is hidden by changes of variable in the input and/or output spaces of the maps — in general, this change of variable makes the resulting map "enough" like a random multivariate quadratic to ensure security.

The fundamental distinctions between different schemes from MVQC lie in the construction of the central map. *Big-field* multivariate cryptography, for instance, considers easy-to-invert maps with components of $\mathbb{F}_{p^r}[x]$, which are then interpreted as maps in $\mathbb{F}_p^r[x]$ — typical examples of this would be Patarin's Hidden Field Equations (HFE, [22]), or Matsumoto and Imai's C^* ([20]). In this text (in §3.2), however, we will be dealing with the *Unbalanced Oil and Vinegar* scheme (UOV) — this is the fundamental scheme in *small-field* multivariate cryptography.

The interest in UOV can be justified by the fact that *Rainbow* ([13]), a scheme derived from UOV, made it to Round 3 of the NIST competition for Quantum-secure signatures. However, very recently Beullens constructed a practical break against Rainbow in [8]. Nonetheless, this does not compromise the security of UOV — its main weakness was and remains unreasonably large public keys, and other viable approaches to remedy this exist, such as Beullens' MAYO ([7]). Nonetheless, in this text (in §3.2), we will be focusing on UOV, in particular to prove the security of two common simplifications of it, as well as delve into how a recent reformulation of it by Beullens in [6] provides an opportunity to shorten secret keys.

We note that, although we will not talk about it in this text, there is a great deal of literature focused on cryptanalizing UOV, as it is not formally known to be secure. We recommend [6] as a reference for known attacks.

3.1 Fundamental hard problems

We aim to introduce the problems upon which Multivariate Cryptography is built. Informally, one would expect to interpret this as problems that admit reductions from breaking MVQC schemes — however, in the case of MVQC, the relationship is less straightforward (and formally weaker). In this section, we will introduce the necessary concepts to define these problems, and later discuss their relationship to MVQC.

3.1.1 Single-component quadratic maps

The fundamental object of interest will be quadratic maps of several variables over finite fields — however, to start building up some helpful results, we will briefly be restricting ourselves to quadratic maps with a single component. Indeed, let $p\colon \mathbb{F}_q^n \to \mathbb{F}_q$ be a quadratic map of n variables $\mathbf{x} = (x_1, \ldots, x_n)$. Note that one can easily organize the coefficients of p:

$$p(\mathbf{x}) = \sum_{1 \leq i \leq j \leq n} A_{ij} x_i x_j + \sum_{1 \leq i \leq n} b_i x_i + c$$

At this point, if we define A_{ij} to be zero for $i > j$, A can be regarded as a matrix and b as a (column) vector. So we have that:

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

This is a **matricial representation** of p. Note that it is not unique, as adding any skew-symmetric matrix to A would not change the value of $\mathbf{x}^\perp A \mathbf{x}$ — it will be useful to characterize exactly when two different representations give rise to the same quadratic map.

Unique matricial representations

Though somewhat obtuse, it is useful to note that matricial representations are "linear". What we mean by this is that, if we introduce the notation $[A, b, c] := x \mapsto \mathbf{x}^\perp A \mathbf{x} + b^\perp \mathbf{x} + c$, then a simple calculation reveals that

$$[A_1, b_1, c_1] + [A_2, b_2, c_2] = [A_1 + A_2, b_1 + b_2, c_1 + c_2],$$

where the $+$ on the LHS denotes a sum of functions, but on the RHS of standard matrices over $\mathbb{F}_q$. So for unicity, we want to restrict the value of (A, b, c) such that it cannot happen that $(A_1, b_1, c_1) \neq (A_2, b_2, c_2)$ while $[A_1, b_1, c_1] = [A_2, b_2, c_2]$ — or put another way, $(A_1 - A_2, b_1 - b_2, c_1 - c_2) \neq 0 = [A_1 - A_2, b_1 - b_2, c_1 - c_2]$. This is tantamount to imposing that there cannot be $(A, b, c) \neq 0$ with $[A, b, c] = 0$.

It is easy to see that we can restrict A to be upper triangular — indeed, since $x_i x_j = x_j x_i$ (in $\mathbb{F}_q$, at least), any quadratic expression of n variables can be written such that all the quadratic terms $x_i x_j$ have $i \leq j$. At this point, we have to segregate the analysis according to the characteristic of the underlying field:

- If $q > 2$, we have enough to ensure uniqueness. To show this, assume that $p = [A, b, c] = 0$, and we will show that $(A, b, c) = 0$. To start, $p(0) = c = 0$. Moreover, for each $i \in [n]$, $p(e_i) = A_{ii} + b_i = 0$, and letting $x \in \mathbb{F}_q$ be such that $x^2 \neq x$, we also have that $p(xe_i) = x^2 A_{ii} + xb_i = 0$. This linear system implies that $A_{ii} = b_i = 0$, since it is homogenous and its determinant is $x^2 - x \neq 0$. Finally, for off-diagonal elements A_{ij} of A, note that $p(e_i + e_j) = A_{ij} + b_i + b_j = 0$. Since $b_i = b_j = 0$, we have that also $A_{ij} = 0$ — so every entry of A, b and c is null, like we wanted.

- The case of $q = 2$ is different, primarily owing to the fact that there is no x with x^2 in $\mathbb{F}_2$ — in particular, $x_i^2 = x_i$, and so the diagonal of A (i.e. coefficients of the x_i^2 terms) and b (i.e. coefficients of the x_i terms) are playing the exact same role. In other words, $[A, b, c] = [A + diag(b), \vec{0}, c]$ — so, in $\mathbb{F}_2$, we impose the additional restriction that $b = \vec{0}$. Proceeding similarly to before[1], we obtain that this already yields uniqueness. This motivates the following two statements:

Definition 3.1. *Let $p\colon \mathbb{F}_q^n \to \mathbb{F}_q$ be a quadratic map. If $q > 2$, a matricial representation $[A, b, c]$ of p is **standard** if A is upper triangular. If $q = 2$, a matricial representation $[A, b, c]$ of p is standard if A is upper triangular and $b = \vec{0}$.*

Proposition 3.2. *Any quadratic map $p\colon \mathbb{F}_q^n \to \mathbb{F}_q$ has a unique standard representation.*

Polar forms

Given some quadratic map $p : \mathbb{F}_q^n \to \mathbb{F}_q$, we will associate to it a **polar form**, $p^* : (\mathbb{F}_q^n)^2 \to \mathbb{F}_q$, defined by $p^*(\mathbf{x}, \mathbf{y}) = p(\mathbf{x} + \mathbf{y}) - p(\mathbf{x}) - p(\mathbf{y}) + p(\vec{0})$. Note that it is not necessary to know the definition of p to work with p^* — indeed, p^* can be computed just with access to p as an oracle. The polar form is useful, then, because it distills the (homogenously) quadratic part of a quadratic map.

For the sake of clarifying what this statement means, consider the map $\phi\colon f \mapsto f^*$ sending any function between some pair vector spaces f to its polar form f^*. Note that ϕ is clearly linear, and moreover it is easy to verify that both constant and linear functions must belong in the kernel of ϕ:

$$\begin{aligned} f \text{ constant} \implies f^*(x, y) &= f(x + y) - f(x) - f(y) + f(0) \\ &= f(0) - f(0) - f(0) + f(0) = 0, \\ f \text{ linear} \implies f^*(x, y) &= f(x + y) - f(x) - f(y) + f(0) \\ &= f(x) + f(y) - f(x) - f(y) = 0. \end{aligned}$$

[1] Obviously evaluating at $2e_i$ is not useful in $\mathbb{F}_2$, where $2 = 0$. However, this is unnecessary, since we already have that $b_i = 0$.

So if p is like before with some matricial representation $p(\mathbf{x}) = \mathbf{x}^\top A\mathbf{x} + b^\top \mathbf{x} + c$, then

$$\begin{aligned} p^* = \phi(p) &= \phi(\mathbf{x} \mapsto \mathbf{x}^\top A\mathbf{x} + b^\top \mathbf{x} + c) \\ &= \phi(\mathbf{x} \mapsto \mathbf{x}^\top A\mathbf{x}) + \phi(\mathbf{x} \mapsto b^\top \mathbf{x}) + \phi(\mathbf{x} \mapsto c) \\ &= \phi(\mathbf{x} \mapsto \mathbf{x}^\top A\mathbf{x}) \\ \implies p^*(x,y) &= (\mathbf{x}+\mathbf{y})^\top A(\mathbf{x}+\mathbf{y}) - \mathbf{x}^\top A\mathbf{x} - \mathbf{y}^\top A\mathbf{y} + \vec{0}^\top A\vec{0} \\ &= \mathbf{x}^\top A\mathbf{y} + \mathbf{y}^\top A\mathbf{x} \\ &= \mathbf{x}^\top (A + A^\top)\mathbf{y}. \end{aligned}$$

Note that p' is a bilinear form. We show the use of p' by way of example, showing how it can let us retrieve a matricial representation of p just from oracle access to p.

Retrieving representations of quadratic maps

We introduce a couple of standard results necessary for the following couple sections:

Lemma 3.3. *Let M be an skew-symmetric matrix in $\mathcal{M}(\mathbb{K}, n, n)$. Then, for any $\mathbf{x} \in \mathbb{K}^n$, it holds that $\mathbf{x}^\top M\mathbf{x} = 0$.*

Proof. $\mathbf{x}^\top M\mathbf{x} = (\mathbf{x}^\top M\mathbf{x})^\top = \mathbf{x}^\top M^\top (\mathbf{x}^\top)^\top = \mathbf{x}^\top(-M)\mathbf{x} = -\mathbf{x}^\top M\mathbf{x}$, and so $\mathbf{x}^\top M\mathbf{x} = 0$. □

Lemma 3.4. *Let A, B be two matrices from $\mathcal{M}(\mathbb{K}, n, n)$ such that $A - B$ is skew-symmetric. Then, for any $\mathbf{x} \in \mathbb{K}^n$, it holds that $\mathbf{x}^\top A\mathbf{x} = \mathbf{x}^\top B\mathbf{x}$.*

Proof. $\mathbf{x}^\top B\mathbf{x} \overset{*}{=} \mathbf{x}^\top B\mathbf{x} + \mathbf{x}^\top(A - B)\mathbf{x} = \mathbf{x}^\top B\mathbf{x}$, the marked equality from Lemma 3.3. □

Now, observe that $2A - (A + A^\top) = A - A^\top$ must be skew-symmetric, and therefore $p^*(\mathbf{x}, \mathbf{x}) = 2\mathbf{x}^\top A\mathbf{x}$, meaning that, as long as we are in $\mathbb{F}_q$ with $q \neq 2$ (i.e. we are in a field where 2 is invertible), the polar form allows us to recover a matrix M such that $\mathbf{x}^\top M\mathbf{x} = \mathbf{x}^\top A\mathbf{x}$ (by setting $(M_{ij}) = \frac{1}{2}p^*(e_i, e_j)$). With this, it is easy to consider the function

$$p(\mathbf{x}) - \frac{1}{2}p^*(\mathbf{x}, \mathbf{x}) = \mathbf{x}^\top A\mathbf{x} + b^\top \mathbf{x} + c - \mathbf{x}^\top A\mathbf{x} = b^\top \mathbf{x} + c,$$

from which we can easily extract b and c — the takeaway being that we can obtain a matricial representation of p just from querying p at certain values (in particular, with a quadratic number of queries). Moreover, let L be the strictly lower triangular part of M. Then, $L^\top - L$ is skew-symmetric, so $\mathbf{x}^\top A\mathbf{x} = \mathbf{x}^\top M\mathbf{x} = \mathbf{x}^\top(M - L + L^\top)\mathbf{x}$. Note that $M - L + L^\top$ is upper triangular, so this also allows us to recover a standard representation of p.

This is just one application. The polar form is generally more useful as a theoretical tool — however, the essence of its application is always similar to its usage above, i.e. separating out the homogenous part of a quadratic map.

Retrieving representations in $\mathbb{F}_{2^r}$

The analysis from the prior section is valid in $\mathbb{F}_2$ exactly up until the point where we talk about "$\frac{1}{2}$" — this implies that, in $\mathbb{F}_2$, for any quadratic map p, the polar map satisfies $p'(\mathbf{x}, \mathbf{x}) = 2\mathbf{x}^\top A\mathbf{x} = 0$, meaning it cannot tell us anything about the diagonal of A. So, let $p = [A, b, c]$ be a standard matricial representation, and we will have to find some other way to extract the values of A, b, c from querying p. Clearly $c = p(\vec{0})$. Now, we segregate according to r:

- If $r = 1$, that is, we are in $\mathbb{F}_2$, and so $b = \vec{0}$. Moreover, $A_{ii}^2 = p(e_i) - c$.
- If $r > 1$, again letting $x \in \mathbb{F}_{2^r}$ be such that $x \neq x^2$, we have that $p(e_i) - c = A_{ii}^2 + b_i$ and $p(xe_i) - c = x^2 A_{ii}^2 + xb_i$ — a linear system which we can invert.

In both cases, we can determine b and the diagonal of A. Now, for each $i < j$, we have $p^*(e_i, e_j) = e_i^\top (A + A^\top) e_j = A_{ij}$ — so, like before, we can retrieve a standard representation of p.

Homogeneization

Consider some quadratic map $p : \mathbb{F}_q^n \to \mathbb{F}_q$ of n variables $\mathbf{x}$, and a matricial representation of this quadratic map, $p(\mathbf{x}) = \mathbf{x}^\top A\mathbf{x} + b^\top \mathbf{x} + c$. Notice that

$$\begin{pmatrix} \mathbf{x}^\top & \big| & 1 \end{pmatrix} \left(\begin{array}{c|c} A & b \\ \hline 0 & c \end{array}\right) \left(\begin{array}{c} \mathbf{x} \\ \hline 1 \end{array}\right) = \mathbf{x}^\top A\mathbf{x} + b\mathbf{x} + c = p(\mathbf{x})$$

This is an **affine representation** of p. The same restrictions on A, b, c as before work to ensure that this representation is unique. Moreover, define the **homogeneization** of p to be the map $\overline{p} \colon \mathbb{F}_q^{n+1} \to \mathbb{F}_q$ defined by

$$p(\mathbf{v}) = \mathbf{v}^\top \left(\begin{array}{c|c} A & b \\ \hline 0 & c \end{array}\right) \mathbf{v}$$

and note that $\overline{p}(x_1, \ldots, x_n, 1) = p(x_1, \ldots, x_n)$. The homogeneization is, unsurprisingly, a (quadratically) homogenous map. Now, such a map q can be represented as $q(\mathbf{x}) = \mathbf{x}^\top M\mathbf{x}$, and clearly (or by Prop. 3.1) this representation can be made unique by imposing that M be upper triangular. Moreover, if $p = [A, b, c]$ is a standard representation, then the composite matrix above is upper triangular — in fact, it can be *any* upper triangular matrix, as all its entries will be entries of either A, b or c. Therefore:

Proposition 3.5. *The set of quadratic maps $p \colon \mathbb{F}_q^n \to \mathbb{F}_q$ and the set of homogenous quadratic maps $q \colon \mathbb{F}_q^{n+1} \to \mathbb{F}_q$ are of the same cardinality — in particular, the homogeneization map $p \mapsto \overline{p}$ is a bijection between them.*

Computational aspects

We mentioned in §2.2.1¶2 that we would treat algorithms as if their input could be pure mathematical objects, without much regard for how these would be represented. Since all our algorithms will be dealing with quadratic maps, it is worthwhile to collect some of the observations from this section regarding how algorithms may deal with quadratic maps:

- The most useful observation is that the representation of a quadratic map p is irrelevant, as long as it allows for efficient evaluation. If this is the case, then, as we saw earlier, this permits efficiently retrieving a standard matricial representation.
- Given p, one can efficiently evaluate its polar form p^*, and also retrieve a matricial representation of p^* — either directly or through a representation of p.
- Given p, one can efficiently (through finding a representation of p) find a representation of its homogeneization $\overline{p}$, and thus also evaluate $\overline{p}$.
- Given $p = [A, b, c]$, and a matrix M, one can efficiently compute a representation of the composition map $p \circ M = [M^\perp AM, bM, c]$.
- One can efficiently generate an uniformly random quadratic map p by simply generating an uniformly random standard matricial representation, i.e. filling each nonzero entry with a random element of $\mathbb{F}_q$, and likewise for homogenous maps.

During the next section, note that all of these observations extend without difficulty to the multivariate case. With this in mind, during the remainder of the text we will be able to describe algorithms that perform use the operations enumerated above — and we will do this without precisely specifying how these operations are carried out.

3.1.2 Multivariate quadratic maps

For a fixed power-of-prime modulus q, a **multivariate quadratic map** over n variables $\mathbf{x}$ with m components is a map $\mathcal{P}\colon \mathbb{F}_q^n \to \mathbb{F}_q^m$, sending $\mathbf{x}$ to $(p_1(\mathbf{x}), \ldots, p_m(\mathbf{x}))$, with each component p_i being a quadratic map of n variables. We say that a multivariate quadratic map is **homogenous** if each of its components is homogenous of degree 2. The **polar form** $\mathcal{P}^*$ of $\mathcal{P}$ is the element-wise polar form, $(p_1^*(\mathbf{x}), \ldots, p_m^*(\mathbf{x}))$, and likewise the **homogeneization** $\overline{\mathcal{P}}$ of $\mathcal{P}$ is obtained through element-wise homogeneization.

The MQ problem

The fundamental problem buttressing the safety of multivariate cryptography is the MQ problem — for fixed q, n, m, an instance of this problem consists of a multivariate quadratic map $\mathcal{P}\colon \mathbb{F}_q^n \to \mathbb{F}_q^m$ and an element $\mathbf{t} \in \mathbb{F}_q^m$, and a solution to this instance would be an element $\mathbf{x} \in \mathbb{F}_q^n$ such that $\mathcal{P}(\mathbf{x}) = \mathbf{t}$, if such an element exists. In short, the MQ problem is the problem of inverting multivariate quadratic maps over some field $\mathbb{F}_q$ — or, in perhaps more familiar terms, the problem of solving multivariate quadratic equations over this same field. This problem is known to be computationally hard (see AppendixA).

The relationship between MVQC and MQ

The fact that MQ is computationally hard does not mean that it is hard in the cryptographical sense. Indeed, the proof that this problem is computationally hard relies on some specific farfetched polynomial systems that encode other NP-hard problems — however, one can imagine a "partial" MQ solver that only solved some (non-negligible) portion of all problem instances. As far as the wisdom of computational complexity is concerned, this solver could definitely exist, as its existence does not suggest that P=NP *as long as* it fails on the particular problem instances that encode other NP-hard problems — but such a solver would definitely threaten the cryptographical security of any scheme based on the hardness of MQ, because a scheme that is secure *some of the time* (or even most of the time) is not actually secure at all.

However, it is generally believed that MQ is hard-on-average, i.e. that no such solver exists. More importantly, it is also generally believed that (the distribution of) quadratic systems that come up in specific instances of MVQC schemes is computationally indistinguishable from that of uniformly sampled quadratic systems. If this is indeed the case, then no solver can perform better on average at solving the systems that come up on MVQC, as otherwise it would serve as an algorithm to distinguish these systems from uniform ones. Indeed, the best algorithms known for solving polynomial systems (primarily XL[11] and Gröbner-base[14] algorithms) do not exhibit any kind of speedup when running on the kinds of systems used in MVQC. All in all, the current state of the art in attacks suggests that (for appropiately chosen parameters), solving the quadratic systems underpinning the security of MVQC is no easier than solving the MQ problem altogether.

The homogenousMQ problem

At some points in the text we will focus on (quadratically) homogenous maps — so it becomes necessary to justify that this does not incur a significant loss in security.

Denote by $MQM_q(n,m)$ the set of multivariate quadratic maps from $\mathbb{F}_q^n$ to $\mathbb{F}_q^m$, and likewise $HMQM_q(n,m)$ for homogenous multivariate quadratic maps. With this, $\mathsf{MQ}_q(n,m)$ is the problem of inverting random maps from $MQM_q(n,m)$ at random $\mathbf{t} \in \mathbb{F}_q^m$, and likewise for $\mathsf{homogenousMQ}_q(n,m)$. Then:

Proposition 3.6. *If there exists an efficient algorithm* A *that can solve* $\mathsf{homogenousMQ}_q(n,m)$ *with probability at least* p*, then there exists an efficient algorithm* B *that can solve* $\mathsf{MQ}_q(n,m)$ *with probability at least* p/q*.*

Proof. We directly construct B (Fig. 3.1), in the usual style of cryptographic reductions.

Before discussing the reduction, note that B contains an instruction to uniformly sample a random invertible matrix. That this is possible to do efficiently, i.e. in polynomial time (as opposed to in polynomial expected time) is not trivial. An (optimal) algorithm for this purpose is given in [24]. With that resolved[2] — evidently, if A returns an $\mathbf{x}'$ such that

[2]We could also just have B abort if it takes too long to find an invertible matrix, since this should only happen with probability $1/poly(q)$ — however, avoiding this makes the relationship between A and B cleaner.

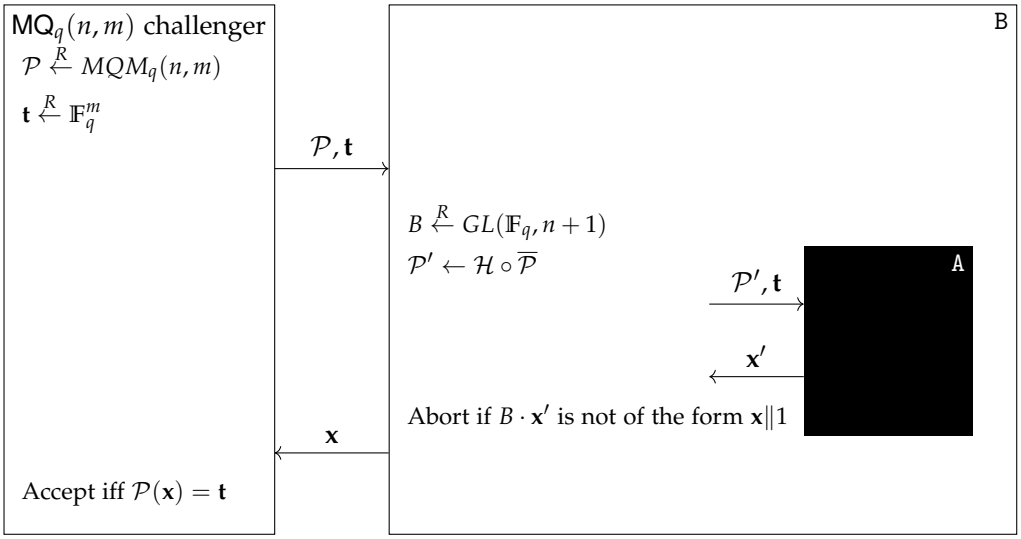


Figure 3.1: Reduction from $\mathsf{MQ}_q(n,m)$ to $\mathsf{homogenousMQ}_q(n+1,m)$

$\mathbf{t} = \mathcal{P}'(\mathbf{x}) = \mathcal{H}(B \cdot \mathbf{x}')$, and $B \cdot \mathbf{x}'$ is of the form $x\|1$, then B returns a $\mathbf{x}$ such that $\mathcal{P}(\mathbf{x}) = \mathbf{t}$, and so the challenger accepts. To complete the proof, however, we need to show two more facts:

1. $\mathcal{P}'$ is distributed uniformly over $HMQM_q(n+1,m)$. Since $\mathbf{t}$ is uniform over $\mathbb{F}_q^m$ and independent from $\mathcal{P}$ and $\mathcal{P}'$, this will imply that A is recieving inputs as in a real $\mathsf{homogenousMQ}_q(n+1,m)$ challenge, and thus will output $\mathbf{x}'$ satisfying $\mathcal{P}'(\mathbf{x}) = \mathbf{t}$ with probability p.

2. After A returns $\mathbf{x}'$, the probability that $B \cdot \mathbf{x}'$ is of the form $\mathbf{x}\|1$ is exactly $1/q$.

Combining these two facts we have that the scenario described above (i.e. B succesfully computes a preimage) happens with probability p/q, as we wanted. So, regarding (1) — note that by 3.5, $\mathcal{H}$ is uniformly distributed over $HMQM_q(n+1,m)$. Now, if $\mathcal{H}(\mathbf{x}) = x^\top M\mathbf{x}$, then $\mathcal{P}'(\mathbf{x}) = (B\mathbf{x})^\top M(B\mathbf{x}) = \mathbf{x}^\top(B^\top MB)\mathbf{x}$ — i.e., as expected $\mathcal{P}'$ is just an arbitrary change of basis of $\mathcal{H}$, and thus is also uniform over $HMQM_q(n+1,m)$.

Then, regarding (2). The idea is to show that $\mathcal{P}'$ is independent from B. Since B is clearly also independent from $\mathbf{t}$, and $\mathcal{P}', \mathbf{t}$ is the only input that A gets, this would imply that the output $\mathbf{x}'$ is also independent from B. Then, B is uniformly distributed, which implies that $B \cdot \mathbf{x}'$ is also uniformly distributed, and thus the last coordinate of $B \cdot \mathbf{x}'$ is 1 with probability $1/q$, like we want.

So it only remains to show that $\mathcal{P}'$ is indeed independent from B. Let p', b be two particular values that $\mathcal{P}'$ and B may take — then, $\mathcal{P}' = p' \wedge B = b$ if and only if $\mathcal{H} =$

$p' \circ b^{-1} \wedge B = b$. Moreover, $\mathcal{H}$ and B are independent, so

$$\begin{aligned}\Pr(\mathcal{P}' = p' \wedge B = b) &= \Pr(\mathcal{H} = p' \circ b^{-1} \wedge B = b) \\ &= \Pr(\mathcal{H} = p' \circ b^{-1}) \cdot \Pr(B = b) \\ &= \Pr(B = b)/\#HMQM_q(n+1, m) \\ &= \Pr(\mathcal{P}' = p') \cdot \Pr(B = b)\end{aligned}$$ □

Remark. Though it will not be useful to us, it is rather easy to come up with a *computational* reduction between the same two problems which does not incur this $1/q$ factor — to solve some quadratic system $\mathcal{P}(\mathbf{x}) = \mathbf{t}$, simply solve the homogeneization $\overline{\mathcal{P}}(\mathbf{x} \| x_{n+1}) = \mathbf{t}$, with the extra (homogenous) constraint that $x_{n+1}^2 = 1$. The solution will have $x_{n+1} = \pm 1$ — even if $x_{n+1} = -1$, the homogeneity of $\overline{\mathcal{P}}$ ensures that one can recover a solution of the form $\mathbf{x}\|1$, and so $\mathbf{x}$ will be a solution to our original system.

3.2 UOV

At its core, UOV is simply a rather elegant trapdoor function scheme, where the functions are multivariate quadratic maps $\mathcal{P}\colon \mathbb{F}_q^n \to \mathbb{F}_q^m$. As is usual in the literature, we will treat UOV as an FDH-like signature scheme: a message is signed by taking its hash and then using the trapdoor to find an preimage of this hash. We will give two different descriptions of UOV, according to two different ways of conceptualizing its secret key.

We note that we will not be *salting* the hashes, that is, hashing $m\|s$ and adding s to the signature of m. Salting is standard, as it ensures that the resulting signature scheme has nondeterministic signatures regardless of the underlying trapdoor scheme — nonetheless, avoiding this simplifies the analysis a great deal. We will draw attention to the distinction when it is relevant.

The origin of (the traditional description of) UOV can be traced to [23], wherein Patarin introduced the "Oil and Vinegar scheme" – importantly, this OV scheme took $n = 2m$. This was later found to be insecure by Kipnis and Shamir in [17], which prompted the introduction of "Unbalanced Oil and Vinegar" (UOV) in [16], taking $n > 2m$. In the following section we will give the original description of this signature scheme.

3.2.1 Traditional description

For convenience, we will say that a multivariate quadratic map $\mathcal{P}$ has the **OV property** if none of its components has $a_{j,k}x_jx_k$ terms with $j, k \in [n-m, n]$. The construction begins with the so-called central map $\mathcal{F}\colon \mathbb{F}_q^n \to \mathbb{F}_q^m$, which is a map with the OV property.

It will be useful to rephrase this in terms of matricial representation. For this, consider any $\mathbf{x} \in \mathbb{F}_q^n$, and split the last m components from the rest: $\mathbf{x} = (\mathbf{x}_v \mid \mathbf{x}_o)$, with $\mathbf{x}_v \in \mathbb{F}_q^{n-m}$ and $\mathbf{x}_o \in \mathbb{F}_q^m$. Note that the prior paragraph essentially states that $f_i(\mathbf{x}_v \mid \mathbf{x}_o)$ has no quadratic term in the $\mathbf{x}_o$ variables. Then, let $f_i = [A_i, b_i, c_i]$ be a standard matricial representation,

and split A_i analogously to $\mathbf{x} = (\mathbf{x}_v \mid \mathbf{x}_o)$:

$$f_i(\mathbf{x}) = f_i(\mathbf{x}_v \mid \mathbf{x}_o) = (\mathbf{x}_v \mid \mathbf{x}_o)^\top \left(\begin{array}{c|c} S_i & D_i \\ \hline \vec{0} & F_i \end{array}\right) \left(\begin{array}{c} \mathbf{x}_v \\ \hline \mathbf{x}_o \end{array}\right) + b_i^\top \mathbf{x} + c_i$$

The restriction we put earlier on $\mathcal{P}$ is equivalent to saying that the matrix F_i must be uniformly zero, as otherwise f_i would have quadratic terms in the last m variables[3]. With this in mind, we can expand the expression by multiplying out the $\mathbf{x}^\top A_i \mathbf{x}$ term:

$$f_i(\mathbf{x}) = \mathbf{x}_v^\top S_i \mathbf{x}_v + \mathbf{x}_v^\top D_i \mathbf{x}_o + b_i^\top \mathbf{x} + c_i$$

The key observation now is that, if one fixes x_v, then $f_i(\mathbf{x}_v \mid \mathbf{x}_o)$ becomes an affine function of $\mathbf{x}_o$. More importantly, the same can be said of $\mathcal{F}(\mathbf{x}_v \mid \mathbf{x}_o)$, which now becomes an affine map $\mathbb{F}_q^m \to \mathbb{F}_q^m$ — since the dimension of domain and codomain are equal, this map is easy to invert as long as it is of full rank. This gives us a procedure to invert the central map $\mathcal{F}$ — if we are seeking $\mathbf{x} = \mathcal{F}^{-1}(\mathbf{t})$:

1. Fix the first $n - m$ variables randomly.
2. Repeat the first step until the resulting map $\mathbf{x}_o \mapsto \mathcal{F}(\mathbf{x}_v \mid \mathbf{x}_o)$ is of full rank[4].
3. Inverting the resulting affine map, find $\mathbf{x}_o$ such that $\mathcal{F}(\mathbf{x}_v \mid \mathbf{x}_o) = \mathbf{t}$.
4. Output $\mathbf{x} = (\mathbf{x}_v \mid \mathbf{x}_o)$.

The core idea of *UOV* (and *OV*), then, is as follows: the secret key will be a full-rank affine map $\mathcal{T}\colon \mathbb{F}_q^n \to \mathbb{F}_q^n$ — essentially acting as a change of coordinates in $\mathbb{F}_q^n$. The public key will be $\mathcal{P} := \mathcal{F} \circ \mathcal{T}$. To someone without knowledge of $\mathcal{T}$, this should just look like an arbitrary multivariate quadratic map. However, with knowledge of $\mathcal{T}$, one can easily compute $\mathcal{F} = \mathcal{P} \circ \mathcal{T}^{-1}$, and thus $\mathcal{P}^{-1} = (\mathcal{F} \circ \mathcal{T})^{-1} = \mathcal{T}^{-1} \circ \mathcal{F}^{-1}$ — and this can be computed easily too, as $\mathcal{T}$ is an affine map and $\mathcal{F}$ can be inverted with the procedure we just described. These are all the pieces we need to define the signature scheme UOV:

- `keyGen`(1^λ) generates a random central map $\mathcal{F}\colon \mathbb{F}_q^n \to \mathbb{F}_q^m$ and affine map $\mathcal{T}\colon \mathbb{F}_q^n \to \mathbb{F}_q^n$, and returns $(pk, sk) = (\mathcal{F} \circ \mathcal{T}, \mathcal{T})$.
- `sign`(pk,sk,m) employs the procedure described above, using $\mathcal{T} = sk$ to invert $\mathcal{P} = pk$ at $H(m)$, and returns the resulting σ.
- `verify`(pk, m, σ) outputs 1 iff $\mathcal{P}(\sigma) = H(m)$, where $\mathcal{F} = pk$.

Note here that we are not specifying the value of q — in practice, these depend will depend on the security parameter λ in some way to be defined by the specification. We will not fix them, however, to be able to study UOV for any value of q, n, m — we will

[3] more precisely, F_i must be skew-symmetric, but since we constrained it to be upper triangular, this implies that it is zero.

[4] Note that the probability that this map is of full rank depends on $\mathcal{P}$, and may be as low as 0 for extremely rare (not necessarily trivial) maps. This is almost always ignored in the literature, as with overwhelming probability $\mathcal{P}$ will be "regular" enough for these restrictions to be invertible with probability $\approx 1 - q^{-1}$.

speak of $\mathsf{UOV}_q(n, m)$ when these parameters are relevant. Moreover, a full specification of UOV should also depend on what hash function H we are going to use — although this does not matter very much, on account of the fact that we will deal with H in the random oracle model. Nonetheless, *Rainbow*, the close cousin of UOV in the second round of the NIST competition for post-quantum digital signatures ([12]) specifies the usage of hash functions from the SHA2 family.

3.2.2 A secure simplification of UOV

In this section, we will intend to make two simplifications to the signature scheme described above — more for theoretical than practical convenience. In this subsection, we will show that we can assume that $\mathcal{T}$ is linear — the other simplification we will deal with in §3.2.4, as it is easier to reason about with the modern definition.

The fact that choosing $\mathcal{T}$ to be linear is safe is an observation originally made by Braeken et al. in [10, §3.1][5], essentially about a useful way to rewrite $\mathcal{P}(\mathbf{x}) = (\mathcal{F} \circ \mathcal{T})(\mathbf{x})$. Let M and $\mathbf{v}$ be such that $\mathcal{T}(\mathbf{x}) = M\mathbf{x} + \mathbf{v}$, and define $\mathcal{F}'(\mathbf{x}) = \mathcal{F}(\mathbf{x} + \mathbf{v})$ and $\mathcal{T}'(\mathbf{x}) = M\mathbf{x}$. The important observation is that

$$(\mathcal{F} \circ \mathcal{T})(\mathbf{x}) = \mathcal{F}(M\mathbf{x} + \mathbf{v}) = \mathcal{F}'(M\mathbf{x}) = (\mathcal{F}' \circ \mathcal{T}')(\mathbf{x}),$$

and so $\mathcal{F} \circ \mathcal{T} = \mathcal{F}' \circ \mathcal{T}'$, with $\mathcal{F}'$ being a multivariate quadratic map and $\mathcal{T}'$ a linear map, with the same domains and codomains as $\mathcal{F}$ and $\mathcal{T}$.

So let linearUOV be a scheme exactly like UOV, except that `keyGen` always picks $\mathcal{T}$ to be linear. We are interested in seeing that breaking UOV can be reduced to breaking linearUOV, and thus the security of the latter is implied by that of the former. For this, we need a small result:

Lemma 3.7. *Let $\mathcal{F}\colon \mathbb{F}_q^n \to \mathbb{F}_q^m$ be a multivariate quadratic map, and $\mathbf{v}$ be any element from $\mathbb{F}_q^n$. Then, $\mathcal{F}'(\mathbf{x}) := \mathcal{F}(\mathbf{x} + \mathbf{v})$ is also a multivariate quadratic, with the same quadratic terms as $\mathcal{F}$.*

Proof. We show that the condition holds for each component f_i' of $\mathcal{F}'$. Consider a matricial representation $f_i = [A_i, b_i, c_i]$. Then,

$$\begin{aligned} f_i'(\mathbf{x}) = f_i(\mathbf{x} + \mathbf{v}) &= \mathbf{x}^\top A_i \mathbf{x} && \textit{(quadratic terms)} \\ &\quad + \left(\mathbf{v}^\top (A_i + A_i^\top) + b^\top\right)\mathbf{x} && \textit{(linear terms)} \\ &\quad + \mathbf{v}^\top A_i \mathbf{v} + b^\top \mathbf{v} + c_i. && \textit{(constant terms)} \quad \square \end{aligned}$$

Corollary 3.8. *Let $\mathcal{F}$, $\mathbf{v}$ and $\mathcal{F}'$ be like above. If $\mathcal{F}$ has the OV property, then so does $\mathcal{F}'$.*

With this corollary, we are equipped to prove the result:

Proposition 3.9. *Let* `A` *be an algorithm that breaks the EUF(/-CMA) security of* linearUOV *(resp.* UOV*). Then,* `A` *also breaks the EUF(/-CMA) security of* UOV *(resp.* linearUOV*).*

[5] a complete reduction is not given (though the existence of one is suggested in §3.1¶3). The remainder of this section shows that such a reduction does indeed exist.

Proof. In fact, we show something quite a bit more fundamental: UOV and linearUOV are the same scheme, at least from the perspective of someone trying to break them. The only part where their definitions differ is in the `keyGen` — but we will show that both definitions, in fact, produce the same public keys with the same probabilities. So let $\mathcal{U}$ be the set of maps in $MQM_q^{n,m}$ with the OV property, and consider the following three methods of sampling a public key:

Method 1	Method 2	Method 3
1. $M \xleftarrow{R} \mathcal{M}(\mathbb{F}_q, n, n)$	1. $M \xleftarrow{R} \mathcal{M}(\mathbb{F}_q, n, n)$	1. $M \xleftarrow{R} \mathcal{M}(\mathbb{F}_q, n, n)$
2. $b \xleftarrow{R} \mathbb{F}_q^n$	2. $b \xleftarrow{R} \mathbb{F}_q^n$	2. $\mathcal{T} \leftarrow [\mathbf{x} \mapsto M\mathbf{x}]$
3. $\mathcal{T} \leftarrow [\mathbf{x} \mapsto M\mathbf{x} + b]$	3. $\mathcal{T}' \leftarrow [\mathbf{x} \mapsto M\mathbf{x}]$	3. $\mathcal{F} \xleftarrow{R} \mathcal{U}$
4. $\mathcal{F} \xleftarrow{R} \mathcal{U}$	4. $\mathcal{F} \xleftarrow{R} \mathcal{U}$	4. $\mathcal{P} \leftarrow \mathcal{F} \circ \mathcal{T}$
5. $\mathcal{P} \leftarrow \mathcal{F} \circ \mathcal{T}$	5. $\mathcal{F}' \leftarrow [\mathbf{x} \mapsto \mathcal{F}(\mathbf{x} + b)]$	
	6. $\mathcal{P} \leftarrow \mathcal{F}' \circ \mathcal{T}'$	

Notice that Method 1 corresponds to UOV, while Method 3 corresponds to linearUOV. Method 2 is an intermediate step, and, by the observation of Braeken et al., is equivalent to Method 1 — so, we focus on showing that Method 2 and Method 3 are equivalent. Evidently $\mathcal{T}$ and $\mathcal{T}'$ have the same distribution and are independent from $\mathcal{F}'$ and $\mathcal{F}$, respectively, so we simply have to show that $\mathcal{F}'$ in Method 2 has the same distribution as $\mathcal{F}$ in Method 3 - i.e. that $\mathcal{F}'$ in Method 2 is uniform over $\mathcal{U}$.

But this is rather straightforward: let f be some particular map from $\mathcal{U}$, and we'd like to show that $\Pr(\mathcal{F}' = f)$ does not depend on f. Note that $\mathcal{F}'(\cdot) = f(\cdot) \iff \mathcal{F}(\cdot + b) = f(\cdot) \iff \mathcal{F}(\cdot) = f(\cdot - b)$, and so $\Pr(\mathcal{F}' = f) = \Pr(\mathcal{F} = f(\cdot - b))$. Notice that $\mathcal{F}$ is uniformly distributed and independent of b, so this last probability does not depend on f — like we wanted. □

From this point on, we assume $\mathcal{T}$ to be a linear map, with the knowledge that this cannot compromise the security of UOV.

3.2.3 Modern description

What we call the "modern description" of UOV was introduced by Beullens in [6]. Similar ideas can be found in earlier key recovery attacks (i.e. ways to recover $\mathcal{T}$ from $\mathcal{P} = \mathcal{F} \circ \mathcal{T}$), where it was already noticed that with overwhelming probability one would simply recover a linear map $\mathcal{T}'$ such that $\mathcal{P} \circ \mathcal{T}'^{-1}$ had the OV property (and thus was easy to invert). This $\mathcal{T}'$ would be called an "equivalent key" (to $\mathcal{T}$). With some work, one can find that maps $\mathcal{T}'$ with this property are exactly the maps such that

$$\mathcal{T}(F) = \mathcal{T}'(F), \text{ where } F = \{0\}^{n-m} \times \mathbb{F}_q^m \subseteq \mathbb{F}_q^n.$$

If one knows $\mathcal{T}(F)$ it is not hard to construct a $\mathcal{T}'$ satisfying the property above. With this in mind, the idea underlying this new way of describing UOV is to eschew a full description of $\mathcal{T}$, and only keep the subspace $\mathcal{O} := \mathcal{T}^{-1}(F)$. Ideally, we would want to

eschew the need to talk about a map $\mathcal{T}$ at all — but, for this, we need to figure out what properties $\mathcal{O}$ needs to inherit from $\mathcal{T}$.

For this, note that if one restricts a map with the OV property (say, $\mathcal{F}$) to F, all the quadratic terms vanish, and thus $\mathcal{F}_{|F}$ is affine. Therefore, composing the two linear maps $\mathcal{F}_{|F}$ and $\mathcal{T}$ with the adequate restrictions, we obtain that $\mathcal{F}_{|F} \circ \mathcal{T}_{|\mathcal{T}^{-1}(F)} = (\mathcal{F} \circ \mathcal{T})_{|\mathcal{O}} = \mathcal{P}_{|\mathcal{O}}$ must also be affine — that is, the public key $\mathcal{P}$ must be affine when restricted to the subspace $\mathcal{O}$, which will be the private key.

We show that knowledge of the secret key $\mathcal{O}$ is indeed enough to invert $\mathcal{P}$ — suppose that, given some $\mathbf{t} \in \mathbb{F}_q^m$, we want to find $\mathbf{v} \in \mathbb{F}_q^n$ such that $\mathcal{P}(\mathbf{v}) = \mathbf{t}$. To start, pick a random $\mathbf{x} \in \mathbb{F}_q^n$, and consider, for any $\mathbf{y} \in \mathcal{O}$, $\mathcal{P}(\mathbf{x}+\mathbf{y}) = \mathcal{P}^*(\mathbf{x},\mathbf{y}) + \mathcal{P}(\mathbf{x}) + \mathcal{P}(\mathbf{y}) + \mathcal{P}(\vec{0})$. Note that, having $\mathbf{x}$ fixed and $\mathbf{y} \in \mathcal{O}$, this is an affine function of $\mathbf{y}$ — hence, it is easily invertible, and we can find $\mathbf{y}$ with $\mathcal{P}(\mathbf{x}+\mathbf{y}) = \mathbf{t}$, obtaining our inverse $\mathbf{x}+\mathbf{y}$. With this, we can describe UOV just in terms of $\mathcal{P}$ and $\mathcal{O}$:

- `keyGen` sets the secret key to a description of $\mathcal{O}$, an m-dimensional subspace of $\mathbb{F}_q^n$, and the public key to $\mathcal{P}$, a random multivariate quadratic map $\mathbb{F}_q^n \to \mathbb{F}_q^m$, such that $\mathcal{P}_{|\mathcal{O}}$ is affine.
- `sign`(pk,sk,m) employs the procedure described above, using $\mathcal{O} = sk$ to invert $\mathcal{P} = pk$ at $H(m)$, and returns the resulting σ.
- `verify`(pk, m, σ) outputs 1 iff $\mathcal{P}(\sigma) = H(m)$, where $\mathcal{P} = pk$.

Remark. We are already working with the assumption that $\mathcal{T}$ is linear — if $\mathcal{T}$ were affine, $\mathcal{O} = \mathcal{T}^{-1}(F)$ would be an affine subspace, as opposed to a linear subspace.

Relation to the traditional description

We will not prove that this modern description of UOV is equivalent to the traditional description (i.e. that either scheme reduces to each other), but it is rather intuitive to see that this should be the case. The following might be regarded as a proof sketch that these two schemes are indeed equivalent:

- `verify` works exactly the same.
- `keyGen` produces the same public keys with the same distribution — this may not be immediately obvious, but notice that the description given of what `keyGen` does, "setting the public key $\mathcal{P}$ to a uniformly random multivariate quadratic map $\mathbb{F}_q^m \to \mathbb{F}_q^n$", is rather non-specific. In practice, it is easy to see that traditional UOV's `keyGen` already selects $\mathcal{P}$ to be a uniformly random multivariate quadratic map affine on $\mathcal{T}^{-1}(F)$, and thus modern UOV's `keyGen` can be built on top of the traditional `keyGen` by simply outputting $\mathcal{O} = \mathcal{T}^{-1}(F)$ instead of $\mathcal{T}$.
- `sign` is the least obvious. To begin with, note that modern UOV's `sign` picks $\mathbf{x}$ uniformly at random from all $\mathbb{F}_q^n$, but it could just as well pick $\mathbf{x}$ from $\mathcal{O}^\top$. Since

this is a set of representatives of the cosets of $\mathcal{O}$, which are going to be where the inputs of $\mathcal{P}(\mathbf{x}+\mathbf{y})$ run over when $\mathbf{y}$ is picked from $\mathcal{O}$, this does not change the resulting $\mathbf{x}+\mathbf{y}$ unless $\mathbf{y} \mapsto \mathcal{P}(\mathbf{x}+\mathbf{y})$ is not of full rank, which only happens with negligible probability. Then, with some algebra, it is easy to see that this is equivalent to picking $\mathbf{x} \in F^{\top}$, and then inverting $F \ni \mathbf{y} \mapsto (\mathcal{P} \circ \mathcal{T}^{-1})(\mathbf{x}+\mathbf{y})$ — a moment's thought should reveal that this is exactly what traditional UOV's `sign` already does.

3.2.4 Another secure simplification of UOV

The second simplification we intend to make is assuming that $\mathcal{P}$ is quadratically homogenous. So, consider the modern description of UOV, and let homogenousUOV be a scheme exactly like UOV, except `keyGen` picks the public key $\mathcal{P}$ to be quadratically homogenous. It is worthwhile to focus for a moment on the fact that, in homogenousUOV, $\mathcal{P}$ is quadratically homogenous, but it is also expected to be affine in $\mathcal{O}$. The fact that it is homogenous clearly implies that $\mathcal{P}(\vec{0}) = 0$, and thus it must necessarily be linear in $\mathcal{O}$. Therefore, for any $\mathbf{o} \in \mathcal{O}$, we have that $\mathcal{P}(2\mathbf{o}) = 2^2\mathcal{P}(\mathbf{o})$, but also $\mathcal{P}(2\mathbf{o}) = 2\mathcal{P}(\mathbf{o})$ — meaning that either $q = 2$ or $\mathcal{P}(\mathbf{o}) = 0$.

We obtain that, in homogenousUOV with $q > 2$, we must have that the public key $\mathcal{P}$ is uniformly null on the secret key $\mathcal{O}$. In fact, this is largely also the case for $q = 2$ (except when no component of $\mathcal{P}$ has a term of the form $x_i x_j$ with $i \neq j$, which is exceedingly rare), so it is reasonable to restrict $\mathcal{P}$ to be unformly null on $\mathcal{O}$ *in general* — this restriction yields the scheme actually presented in [6]. Nonetheless, for the remainder of this subsection we will simply assume that, in homogenousUOV, $\mathcal{P}$ just simply affine on $\mathcal{O}$.

The relationship between the security of UOV and that of homogenousUOV is not straightforward. With the intent of figuring out the nature of this relationship, we make these four claims, which we will prove during the remainder of this subsection:

1. If $\mathsf{homogenousUOV}_q(n, m)$ is vulnerable to key recovery attacks, then so is $\mathsf{UOV}_q(n, m)$, with the same advantage.
2. If public keys in $\mathsf{UOV}_q(n, m)$ are indistinguishable (in distribution) from random maps from $MQM_q(n, m)$, then public keys in $\mathsf{homogenousUOV}_q(n+1, m)$ are indistinguishable (in distribution) from random maps from $HMQM_q(n+1, m)$.
3. If $\mathsf{UOV}_q(n, m)$ is EUF, then so is $\mathsf{homogenousUOV}_q(n+1, m)$ — with a loss factor of $1/q$, i.e. attacks against UOV may be q times harder than against homogenousUOV.
4. If $q > 2$ and $\mathsf{UOV}_q(n, m)$ is EUF-CMA, then so is $\mathsf{homogenousUOV}_q(n+1, m)$ — but having to introduce new hardness assumptions and/or the ROM.

Before proving these four statements, we discuss their implications. (1) and (2) are in some sense heuristic arguments for the security of this simplification. On the one hand, **(1)** essentially implies that UOV and homogenousUOV are equivalent *in practice*, since the literature is primarily focused on key-recovery attacks — and indeed, all the best-performing attacks against UOV (resp. homogenousUOV) are key-recovery attacks, and

they apply with minimal overhead to homogenousUOV (resp. UOV). On the other hand, **(2)** implies that UOV and homogenousUOV are fully equivalent in theory, *if* one accepts the widely believed assumptions that:

a. MQ is hard on average — we saw in Prop.3.6 that this implies that homogenousMQ is also hard on average; and

b. the distribution of public keys in UOV is indistinguishable from uniform, and so (2) implies that this is also the case for homogenousUOV.

If these two claims are accepted, then homogenousUOV is secure by an argument analogous to the one used in the case of UOV.

In the realm of properly defined notions of security — **(3)** and **(4)** are formal reductions. (3) is not very interesting, as it essentially amounts to a restating of Prop. 3.6 through the lens of (2). (4) is the most theoretically relevant result (as EUF-CMA is the most common notion of security for signature schemes), but it has non-insignificant loss in advantage, requires new hardness assumptions, and is performed in the ROM.

We note that we consider EUF(-CMA) security because it is the notion of security that we have defined, but we claim (without proof) that these reductions extend without much difficulty to every notion of security in {UUF/SUF/EUF}-{CMA/KMA}. As for sEUF, note that, as presented, homogenousUOV is not strongly existentially unforgeable, as the homogeneity of $\mathcal{P}$ begets the malleability $\mathcal{P}(\mathbf{x}) = \mathcal{P}(-\mathbf{x})$ — this allows us to, given any signature σ for m, obtain another valid signature $-\sigma$. It seems that this could be fixed by restricting signatures to only be valid if they are in a certain set of representatives of $\mathbb{F}_q^n/(\mathbf{x} \sim -\mathbf{x})$, but we have not considered this to simplify the analysis in this subsection.

The remainder of this subsection is dedicated to the proofs. Statements (1), (2) and (3) correspond to (or follow easily from) Prop. 3.10, Cor. 3.12, and Prop. 3.13, respectively. Statement (4) corresponds to Cor. 3.18, as well as the entire subsubsection containing this corollary.

Proposition 3.10. *If there exists an efficient algorithm* `A` *that succeeds in performing a key recovery attack against* $\mathsf{homogenousUOV}_q(n, m)$ *with some probability* p*, then there exist an efficient algorithm* `B` *that succeeds in performing a key recovery attack against* $\mathsf{UOV}_q(n, m)$ *with the same probability* p*.*

Proof. `B` is simple enough that it is easier to describe it and justify its correctness in tandem. Thus, we describe the actions of `B` on input $\mathcal{P}$. First, `B` separates out the input's homogenous and affine parts, i.e. finds maps $\mathcal{P}_1$ and $\mathcal{P}_2$ such that $\mathcal{P}(\mathbf{x}) = \mathcal{P}_1(\mathbf{x}) + \mathcal{P}_2(\mathbf{x})$, $\mathcal{P}_1$ is homogenous, and $\mathcal{P}_2$ is affine. Then, `B` internally runs `A` with input $\mathcal{P}_1$, obtaining a subspace $\mathcal{O}$ where $\mathcal{P}_1$ is affine. Since $\mathcal{P}_2$ is already affine on its entire domain, it follows that $\mathcal{P}$ is affine on $\mathcal{O}$, and thus `B` outputs $\mathcal{O}$, a valid secret key for $\mathcal{P}$. □

Note that, unlike the reductions that follow, this does not have a $1/q$ loss factor, nor does it rely on increasing the n parameter for the homogenous instance.The fact that

this reduction incurs no loss in advantage, by itself, isn't special: any polynomial loss in advantage can be overcome by a polynomial amount of repetitions. What's remarkable is that there is no loss in advantage while the only increase in runtime is due to separating the homogenous part of the public key. This increase is likely $O(n^2)$, which cannot be higher in complexity than `A`, as otherwise `A` could not read its own input. Therefore, the runtime of `B` is asymptotically equivalent to that of `A`.

Lemma 3.11. *Consider the distribution of $\mathcal{P}'$ given by running $\mathcal{P}, \mathcal{O} \leftarrow \texttt{keyGen}_{UOV(q,n,m)}(1^\lambda)$, $B \xleftarrow{R} GL(\mathbb{F}_q, n+1)$, $\mathcal{P}' \leftarrow \overline{\mathcal{P}} \circ B$. This is identical to the distribution of public keys generated by $\texttt{keyGen}_{homogenousUOV(q,n+1,m)}(1^\lambda)$.*

Proof. $\mathcal{P}$ is affine on $\mathcal{O}$, meaning that $\overline{\mathcal{P}}$ is affine on $\mathcal{O} \times \{1\}$. Moreover, $\overline{\mathcal{P}}$ is also affine on each coset of $\mathcal{O} \times \{1\}$, in particular $\mathcal{O} \times \{0\}$, which is a subspace of $\mathbb{F}_q^{n+1}$. Therefore, $\mathcal{P}'$ is affine on $\mathcal{O}' := B^{-1} \cdot (\mathcal{O} \times \{0\})$. Since B, and therefore also B^{-1}, is independent from $\mathcal{O}$ and uniformly sampled from $GL(\mathbb{F}_q, n+1)$, this means that $\mathcal{O}'$ is uniformly distributed over $Gr(m, \mathbb{F}_q^{n+1})$.

Recall from Prop. 3.5 that the homogeneization map $\mathcal{P} \mapsto \overline{\mathcal{P}}$ is a bijection[6], and thus it must have an inverse — during this proof, we will denote this by $\mathcal{P} \mapsto \underline{\mathcal{P}}$. Now, consider any given homogenous multivariate quadratic map $p' \colon \mathbb{F}_q^{n+1} \to \mathbb{F}_q^m$ that is affine on $\mathcal{O}'$ — we would like to show that $\Pr(\mathcal{P}' = p')$ does not depend on p', and thus $\mathcal{P}'$ is uniformly distributed over the set of such maps.

To start, note that since p' is affine on $\mathcal{O}' = B^{-1}(\mathcal{O} \times \{0\})$, $p' \circ B^{-1}$ is affine on $\mathcal{O} \times \{0\}$, and thus also on $\mathcal{O} \times \{1\}$, which must mean that $\underline{p' \circ B^{-1}}$ is affine on $\mathcal{O}$. With that in mind, note that $\mathcal{P}' = p' \iff \overline{\mathcal{P}} \circ B = p' \iff \mathcal{P} = \underline{p' \circ B^{-1}}$, and thus $\Pr(\mathcal{P}' = p') = \Pr(\mathcal{P} = \underline{p' \circ B^{-1}})$ — this is constant, and so it does not depend on p'. □

Corollary 3.12. *If there exists an efficient algorithm `A` that can distinguish public keys generated by $\texttt{keyGen}_{homogenousUOV(q,n+1,m)}(1^\lambda)$ from uniformly random maps from $HMQM_q(n+1, m)$ then there exists a likewise efficient algorithm `B`, with the same advantage as `A`, that can distinguish public keys generated by $\texttt{keyGen}_{UOV(q,n,m)}(1^\lambda)$ from uniformly random maps from $MQM_q(n, m)$.*

Proof. `B` would simply recieve maps $\mathcal{P}$, apply the process described in Prop. 3.11:

$$B \xleftarrow{R} GL(\mathbb{F}_q, n+1), \quad \mathcal{P}' \leftarrow \overline{\mathcal{P}} \circ B$$

and feed the resulting maps $\mathcal{P}'$ to `A`, eventually outputting whatever `A` outputs. With this:

- If `B` is recieving maps generated by the `keyGen` of $\mathsf{UOV}_q(n, m)$, Prop. 3.11 implies that the maps that `B` sends to `A` will match the distribution of those generated by the `keyGen` of $\mathsf{homogenousUOV}_q(n+1, m)$.
- If the $\mathcal{P}$ are uniformly selected from $MQM_q(n, m)$, Prop. 3.5 implies that $\overline{\mathcal{P}}$ is uniform over $HMQM_q(n+1, m)$. At this point, the argument proceeds analogously to

[6] Prop. 3.5 refers only to the single-component case, but this generalizes trivially.

before: if $p' \in HMQM_q(n+1, m)$, $\mathcal{P}' = p' \iff \overline{\mathcal{P}} = p' \circ B^{-1}$. Clearly $p' \circ B^{-1}$ is also in $HMQM_q(n+1, m)$, and so the probability of these events is constant, implying that $\mathcal{P}'$ is uniform over $HMQM_q(n+1, m)$.

We obtain that B efficiently turns the distributions it has to distinguish intro distributions that A can distinguish, and thus it works with the same advantage as A. □

Proposition 3.13. *If there exists an efficient algorithm* A *that can break the* EUF *security of* homogenousUOV *with probability at least* p*, then there exists an efficient algorithm* B *that can break the* EUF *security of* UOV *with probability at least* p/q*.*

Proof. The proof is essentially analogous to Prop. 3.6 — B will be trying to invert a map by homogenizing it, obscuring it, and passing it to A. As before, the main issue will be ensuring that the distribution of B's messages to A (in this case, only the public key) match a genuine EUF challenger of homogenousUOV.

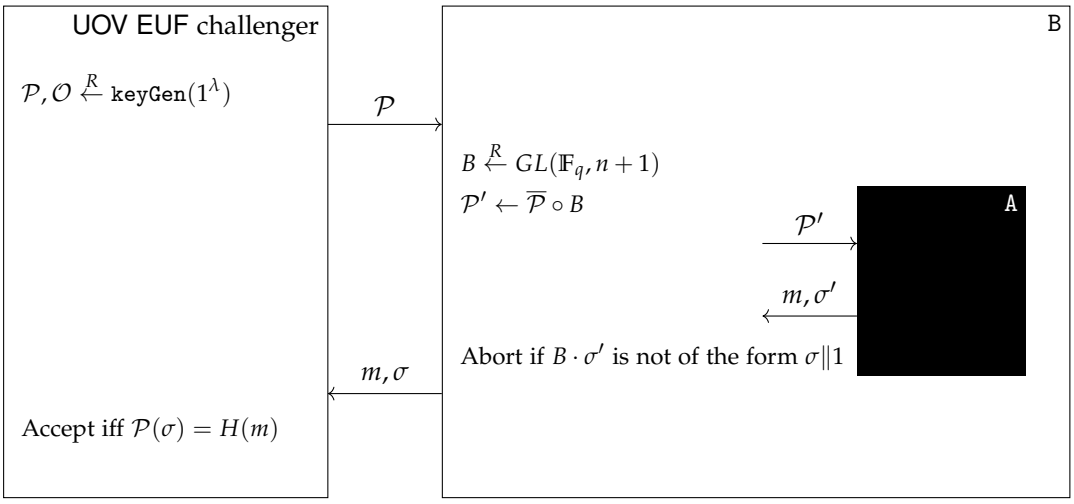


Figure 3.2: Reduction from breaking EUF security of $\mathsf{UOV}_q(n, m)$ to EUF security of $\mathsf{homogenousUOV}_q(n+1, m)$.

The reduction is given in Fig. 3.2. We will need to prove the following two statements:

1. The distribution of the $\mathcal{P}'$ that B sends to A matches that of a genuine homogenousUOV challenger, i.e. the distribution produced by homogenousUOV's keyGen.
2. The distribution of B is independent from that of $\mathcal{P}'$.

If (1) is true, then with probability P, A will output a valid forgery. If (2) is true, because $\mathcal{P}'$ is A's only input (apart from the security parameter), then B will also be independent from the distribution of m and σ' — since B is uniform, this means that $B \cdot \sigma'$ will be of the form $\sigma \| 1$ with probability $1/q$. Combining these two facts, we have that with a probability of p/q, B will output (m, σ) satisfying that $\mathcal{P}(\sigma) = \overline{\mathcal{P}}(\sigma \| 1) = \overline{\mathcal{P}}(B \cdot \sigma') = \mathcal{P}'(\sigma') = H(m)$,

that is, a valid forgery — so we focus on proving these two statements.

(1). This follows immediately from Prop. 3.11.

(2). Let p' be any element of $HMQM_q(n+1, m)$, and $b \in GL(\mathbb{F}_q, n+1)$. Now, $\Pr(\mathcal{P}' = p', B = b) = \Pr(B = b) \cdot \Pr(\mathcal{P}' = p' \mid B = b)$. Prop. 3.11 implies that the latter probability is constant, and thus does not depend on b — therefore, $\Pr(\mathcal{P}' = p', B = b) = \Pr(B = b) \Pr(\mathcal{P}' = p')$, like we wanted. □

On the EUF-CMA security of UOV and homogenousUOV

In this subsubsection, we will give or sketch several reductions. All of these are built from (and thus better understood by) taking as a starting point the naïve adaptation of Prop. 3.13 to the -CMA setting, that is, the reduction in Fig. 3.2 adapted to naïvely to handle signing queries. This is presented in Fig. 3.3 — from this point on, we will omit the challenger from the diagram to avoid clutter.

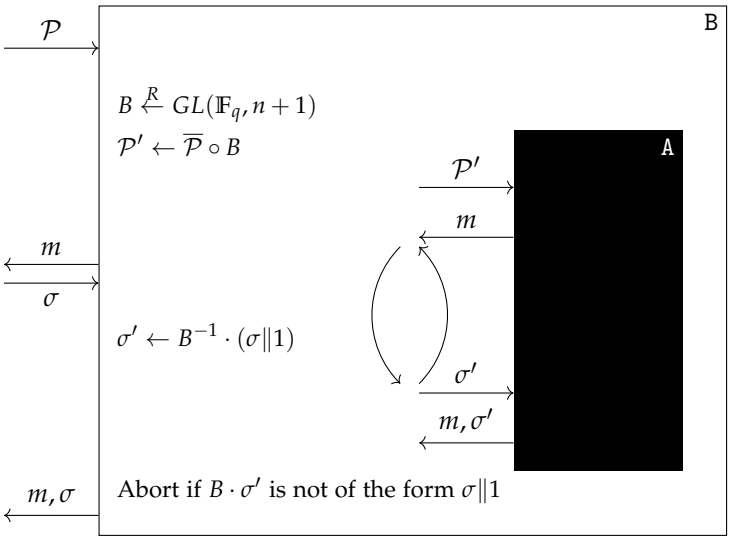


Figure 3.3: Starting point for reductions from breaking EUF-CMA security of $\mathsf{UOV}_q(n, m)$ to EUF-CMA security of $\mathsf{homogenousUOV}_q(n+1, m)$.

Note that, throughout this section, all reductions presented will have the property that B only ever sends messages to its challenger that it has recieved from A (as signing queries or forgeries). Therefore, given that if A succeeds it will do so with a message that it has not queried for a signature on, and thus the same is true of B — so we do not need to worry about checking that B does not forge signatures on messages for which it has queried a signature. Moving on, then — there are two standard "validity" properties that we will expect all reductions to satisfy (and generally, most non-contrived EUF-CMA security reductions will follow):

V1. If A outputs a valid forgery and B does not abort, then B produces a valid forgery.

V2. The signatures that B produces upon A's queries are valid (with respect to the relevant oracle, if in the ROM).

In this case, **V1** is identical to the same property we already proved in Prop. 3.13. **V2** is likewise rather easy to check — indeed, $\mathcal{P}'(\sigma') = \mathcal{P}'(B^{-1} \cdot (\sigma \| 1)) = \overline{\mathcal{P}}(\sigma \| 1) = \mathcal{P}(\sigma) = H(m)$. However, these signatures will appear very conspicuous to A, as it is very easy to detect that they do not follow the distribution of real responses to signing queries — they will all lie in the hyperplane $B^{-1} \cdot (* \| 1)$. A does not know this hyperplane — but it is easy to, after having performed sufficient signing queries, check that the signatures it recieves do not span $\mathbb{F}_q^{n+1}$, which would be overwhelmingly unlikely if it A playing against a genuine challenger.

So our focus becomes on solving this issue — that is, modifying B such that the signatures it produces are indistinguishable from those of a real challenger. From this point on we let A be an efficient algorithm that breaks the EUF-CMA security of homogenousUOV with probability at p. Our first attempt at a reduction algorithm, $\mathtt{B}_1$, is presented in Fig. 3.4.

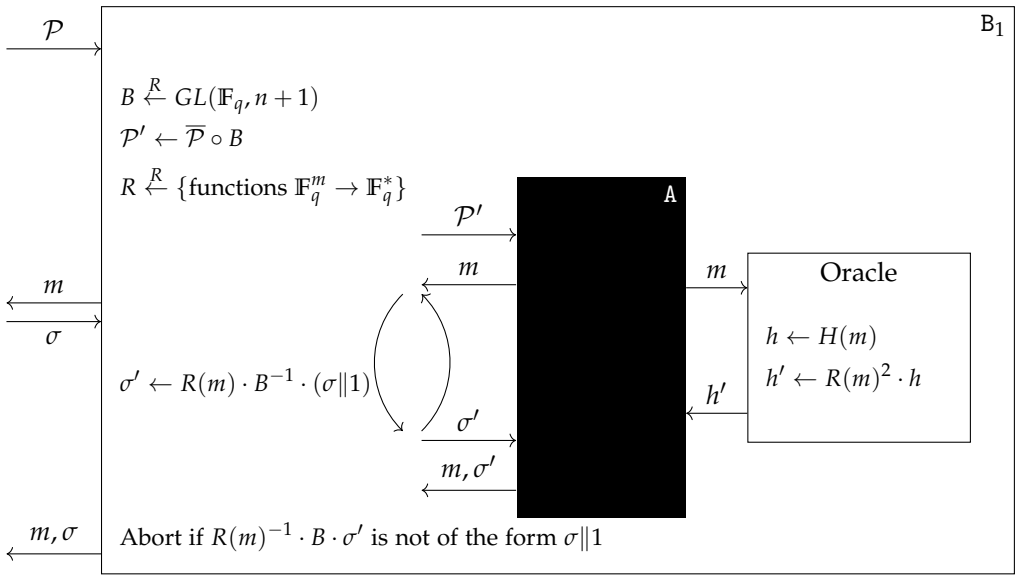


Figure 3.4: Reduction from breaking EUF-CMA security of $\mathsf{UOV}_q(n, m)$ to EUF-CMA security of $\mathsf{homogenousUOV}_q(n+1, m)$.

In essence, $\mathtt{B}_1$ takes advantage of the homogeneity of $\mathcal{P}'$ as much as possible, randomly redistributing signatures across nearly the entire input space and adjusting A's oracle such that these signatures are valid (this assumes that UOV and homogenousUOV are defined with different hash functions). Note that the function R here is used as a shorthand for generating and keeping a random value $R(m) \in \mathbb{F}_q^*$ for each message m. If one is interested in making B stateless, R can be replaced by a secure pseudo-random-function.

Like before, **V1** is identical to the previous proof, and **V2** is easy to check. Indeed, as $\mathcal{P}'$ is quadratically homogenous, this means that $\mathcal{P}'(R(m)^{-1} \cdot \sigma') = R(m)^{-2} \cdot \mathcal{P}'(\sigma') = H(m)$, that is, $\overline{\mathcal{P}}(B \cdot R(m)^{-1} \cdot \sigma') = H(m)$. If $B \cdot R(m)^{-1} \cdot \sigma'$ is of the form $\sigma \| 1$, which must be the case if $\mathtt{B}$ does not abort, we have that $\mathcal{P}(\sigma) = \overline{\mathcal{P}}(\sigma \| 1) = \overline{\mathcal{P}}(B \cdot R(m)^{-1} \cdot \sigma') = H(m)$.

However, also like before, the distribution of the signatures may arouse suspicion in $\mathtt{A}$ — indeed, since $R(m) \neq 0$, we have that all signatures will be of the form $R(m) \cdot B^{-1} \cdot (\sigma \| 1) = B^{-1} \cdot (R(m) \cdot \sigma \| R(m))$, that is, they will never be of the form $B^{-1} \cdot (* \| 0)$. We will work around this issue to tease out a hardness assumption under which $\mathtt{B}_1$ can be shown to produce a valid forgery with probability p/q.

For the sake of illustrating the issue, let us briefly act as if though genuine signatures of $\mathsf{UOV}_q(n, m)$ and $\mathsf{homogenousUOV}_q(n+1, m)$ are uniformly distributed over $\mathbb{F}_q^n$ and $\mathbb{F}_q^{n+1}$, respectively. If this is the case, then $\sigma' = R(m) \cdot B^{-1} \cdot (\sigma \| 1)$ will be uniformly distributed over $\mathbb{F}_q^{n+1} \setminus B^{-1} \cdot (* \| 0)$ — a statistical distance of $1/q$ to the distribution of true signatures, which at first might seem far too high. Crucially, however, $\mathtt{A}$ does not know B, and thus it does not know the subspace $B^{-1} \cdot (* \| 0)$ that signatures are going to be absent from — therefore, the problem that $\mathtt{A}$ faces if it wants to make the reduction invalid is to distinguish these two distributions:

- That of a genuine challenger, that is, uniform over $\mathbb{F}_q^{n+1}$.
- That of $\mathtt{B}_1$ — uniform over $\mathbb{F}_q^{n+1} \setminus H$, where $H = B^{-1} \cdot (* \| 0)$ is a hyperplane unknown to $\mathtt{A}$.

This motivates the definition of the following problem:

Definition 3.14. *The **missing hyperplane problem** in order q and dimension n, abbreviated $\mathsf{MH}_q(n)$, is the problem of distinguishing a uniform distribution over $\mathbb{F}_q^n$ from a uniform distribution over $\mathbb{F}_q^n \setminus H$, where H is a random hyperplane unknown to the distinguishing algorithm.*

The hardness of this problem is explored in Appendix B. The main takeaway pertaining to this problem is that it is easy when $q = 2$, and we assume it to be hard if $q > 2$. This is supported by the fact that the "decision version" of $\mathsf{MH}_q(n)$, i.e. deciding whether a set $P \subseteq \mathbb{F}_q^n$ is such that there exists a hyperplane H satisfying $P \subseteq \mathbb{F}_q^n \setminus H$, i.e. $P \cap H = \varnothing$, is NP-complete when $q > 2$.

It turns out that, with some work, the validity of the reduction $\mathtt{B}_1$ can be based on the hardness of $\mathsf{MH}_q(n + 1 - m)$. Note that this disagrees with our earlier reasoning, which suggested that this would be $\mathsf{MH}_q(n)$ — loosely speaking, the difference comes from taking into account that the distribution of signatures is not actually uniform on $\mathbb{F}_q^n$, but rather uniform on the cosets in $\mathbb{F}_q^{n+1}/\mathcal{O}$ where $\mathcal{P}$ is of full rank.

For the sake of readability, it is easier to first prove a lemma about B_1, before discussing some other aspects of MH that will ensure the validity of B_1:

Lemma 3.15. *Let $\mathtt{A}$ be an efficient algorithm that can, with probability p, break the EUF-CMA security of $\mathsf{homogenousUOV}_q(n+1,m)$, and $\mathtt{B}_1$ as constructed in Fig. 3.4, letting p_a be the probability that $\mathtt{A}$ outputs a valid forgery with respect to its simulated random oracle. Then, there exists an efficient algorithm $\mathtt{C}$ such that, letting p_c be its probability of solving $\mathsf{MH}_q(n+1-m)$, it holds (in the Random Oracle Model) that $p_c \geq |p_a - p|$.*

The proof of Lemma 3.15 is rather long and technical — to facilitate reading, the full proof is given in Appendix C, and here we only give a sketch so we can quickly move on to discussing the implications.

Proof Sketch. The idea is to consider the game for breaking the EUF-CMA security of $\mathsf{homogenousUOV}_q(n+1,m)$, and begin by modifying how the signature process works. In picking an $\mathbf{x}$ to consider $\mathcal{O} \ni y \to \mathcal{P}(\mathbf{x}+\mathbf{y})$, what the challenger is really doing is picking a coset of $\mathcal{O}$ in which to invert $\mathcal{P}$. The space of such cosets can be parametrized by $\mathbb{F}_q^{n-\dim(\mathcal{O})} = \mathbb{F}_q^{n-m}$, and so the challenger can be made to sample an x from $\mathbb{F}_q^{n+1-m}$ and suitably project it into some space in $\mathbb{F}_q^{n+1}$ complementary to $\mathcal{O}$.

We can reformulate this game so that the sampling x from $\mathbb{F}_q^{n+1-m}$ is provided by an external challenger for $\mathsf{MH}_q(n+1-m)$ — what remains as the adversary is $\mathtt{C}$, and if we believe $\mathsf{MH}_q(n+1-m)$ to be hard, the result of this game should be negligibly affected if x is now sampled from $\mathbb{F}_q^{n+1-m}$ except some hyperplane unknown to the adversary. We call the resulting game Game_1, while Game_2 is the one in Fig. 3.4.

The remainder of the proof deals with checking that the distribution of the messages recieved by $\mathtt{A}$ in either game is indistinguishable. The fact that the "keys" ($\mathtt{B}_1$ is not aware of its own secret key) in either game are the same follows from Prop. 3.11. Moreover, in the signing processes, both games will have a hyperplane H from where cosets of the secret key are never picked for signing — in Game_1, this hyperplane H_1 will be decided by the $\mathsf{MH}_q(n+1-m)$ challenger, while in Game_2 we have already seen that this is $H_2 = B^{-1}(*\|0)$. In both games, one can see that this hyperplane is uniformly selected from all the hyperplanes containing the secret key. Finally, we obtain that in both games, the joint distribution of the keys and this hyperplane is identical.

Finally, we note that the state of the game outside $\mathtt{A}$ does not change when responding to queries from $\mathtt{A}$, and thus we only need to show that the distribution of the answers to any given signing query is identical in either game. It is rather straightforward to check that in Game_1, the signing process uniformly selects a coset of the secret key not contained in H_1 where the public key is invertible, and then returns the unique antiimage there. In Game_2, a number of reformulations of the signing process yield that one essentially picks an affine hyperplane parallel to H_2 (in particular, $B^{-1}\cdot(*\|1)$), finds a signature there, and then returns $R(m)\cdot B^{-1}\cdot(*\|1)$, adjusting $\mathtt{A}$'s oracle so that this remains valid. Some cursory checks reveal that this is the same as uniformly picking from cosets contained in any affine hyperplane parallel to H_2 — i.e., cosets not contained in H_2, which is the same as in Game_1. □

Note that this only tells us that $\mathtt{A}$ will output (m,σ) with $\mathcal{P}'(\sigma') = R(m)^2 \cdot H(m)$, and thus

$\overline{\mathcal{P}}(B \cdot R(m)^{-1} \cdot \sigma') = H(m)$ — B will be able to turn this into a valid signature for its own challenger iff $B \cdot R(m)^{-1} \cdot \sigma'$ is of the form $\sigma \| 1$. Unlike in Props. 3.6 and 3.13, here we cannot claim that B is independent from the rest of A's input, so we cannot be sure that B will be able to produce a valid signature $1/q$ of the time.

However, whatever final forgery A outputs, it will have to be on a message m for which it has not queried a signature. This means A will not have seen $R(m)$ anywhere outside of a random oracle query, where it was multiplied by $H(m)$ — in the ROM, both of these values are randomly distributed, and thus we may assume that $H(m)$ masks $R(m)$ and $R(m)$ is uniformly distributed when A outputs its final forgery (m, σ'). This means that, if $B \cdot \sigma'$ is not of the form $(\sigma \| 0)$, then there is an exactly $1/q - 1$ chance that $B \cdot R(m)^{-1} \cdot \sigma'$ is of the form $(\sigma \| 1)$, and thus B outputs a valid forgery.

To recap, this means that B's probability of outputting a valid forgery is $p \cdot (1 - \tilde{p})/(q - 1)$, where $\tilde{p}$ is the probability that the forged signature that A outputs is not in $B^{-1} \cdot (* \| 0)$. To see that this reduction works, then, it only remains to see that $(1 - \tilde{p})$ is non-negligible. This requires introducing a new problem:

Definition 3.16. *The **missing hyperplane sampling problem** in order q and dimension n, abbreviated $\mathsf{MHS}_q(n)$, consists of, given a uniform distribution over $\mathbb{F}_q^n$, where V is some unkown (random, uniform) hyperplane in $\mathbb{F}_q^n$, finding a point $p \in V$, with $p \neq \vec{0}$.*

As with $\mathsf{MH}_q(n)$ earlier, the hardness of this problem (in particular, its relationship to $\mathsf{MH}_q(n)$) is discussed in Appendix B. In any case, the takeaway is that, if $\mathsf{MH}_q(n)$ is hard, then $\mathsf{MHS}_q(n)$ cannot be solved with probability negligibly close to 1. Knowing this,

Proposition 3.17. *Let A be an efficient algorithm that can break the EUF-CMA security of $\mathsf{homogenousUOV}_q(n+1, m)$ with probability p, and $\mathtt{B}_1$ as constructed in Fig. 3.4, letting p_a be the probability that A outputs a valid forgery with respect to its simulated random oracle, and p_b the probability that $\mathtt{B}_1$ outputs a valid forgery. Then, there exist efficient algorithms C and D such that, letting p_c and p_d be their probabilities of solving $\mathsf{MH}_q(n+1-m)$ and $\mathsf{MHS}_q(n+1-m)$, it holds (in the Random Oracle Model) that $p_c \geq |p_a - p|$ and $p_b = (1 - p_d) \cdot p_a/(q-1)$*

Proof. We let C, Game_1, and Game_2 be as in the proof of Prop. 3.15. As per the observation above, $\mathtt{B}_1$ (in Game_2) will find a valid forgery with probability $1/(q-1)$ if A outputs a valid forgery (with respect to its random oracle) that is not in $B^{-1}(* \| 0)$.

Consider Game_2, and modify it thusly: switch out the challenger of $\mathsf{MH}_q(n+1-m)$ for that of $\mathsf{MHS}_q(n+1-m)$, and modify the adversary so that it directly sends the signature of A's final forgery directly to the adversary. The resulting adversary is algorithm D, and thus with probability p_d it sends a signature σ in the hidden hyperplane. We have seen, however, that the distribution of this signature will be identical in Game_1, therefore we have that, in Game_1, with probability $p_a \cdot (1 - p_d)$, A will output a signature not in the hyperplane $B^{-1}(* \| 0)$. This completes the proof. □

Corollary 3.18. *Let A be an efficient algorithm that can break the EUF-CMA security of the scheme $\mathsf{homogenousUOV}_q(n+1, m)$ with non-negligible probability. If $\mathsf{MH}_q(n+1-m)$ is hard, then*

we can construct an efficient algorithm $\mathtt{B}_1$ *that can break the* EUF-CMA *security of* $\mathsf{UOV}_q(n,m)$ *with non-negligible probability.*

Proof. With the same definitions as Prop. 3.17, we have that p_c is negligible, and so p_a is negligibly close to p and thus non-negligible. Moreover, p_d cannot be negligibly close to 1, and so $(1-p_d)$ is non negligible, and therefore so is $p_b = (1-p_d)\cdot p_a/q$. □

There are other approaches to showing that restricting UOV to be homogenous should not compromise EUF-CMA security. We do not present them fully, as we have not found a formal basis for their safety and/or their loss factor is too large. These are:

A modification of $\mathtt{B}_1$ without the ROM. We call this $\mathtt{B}_2$, illustrated in Fig. 3.5. A similar issue to $\mathtt{B}_1$ crops up, in that the signatures that A recieves from $\mathtt{B}_2$ are all of the form $B^{-1}(\sigma\| \pm 1)$. Analogously to $\mathsf{MH}_q(n)$, we define the **hidden hyperplanes problem**, $\mathsf{HHs}_q(n)$, of distinguishing an uniform distribution over some unknown $B^{-1}(\sigma\| \pm 1)$ from an uniform distribution over $\mathbb{F}_q^n$. This problem is also discussed in Appendix B, and shares many of the properties of $\mathsf{MH}_q(n)$. However, $\mathtt{B}_2$ also runs into the problem that A's forgeries may not be of the form that $\mathtt{B}_2$ needs to produce a forgery of its own — and the problem seems significantly more severe in this case, as there is no $R(m)$ allowing us to make an argument like in $\mathtt{B}_1$

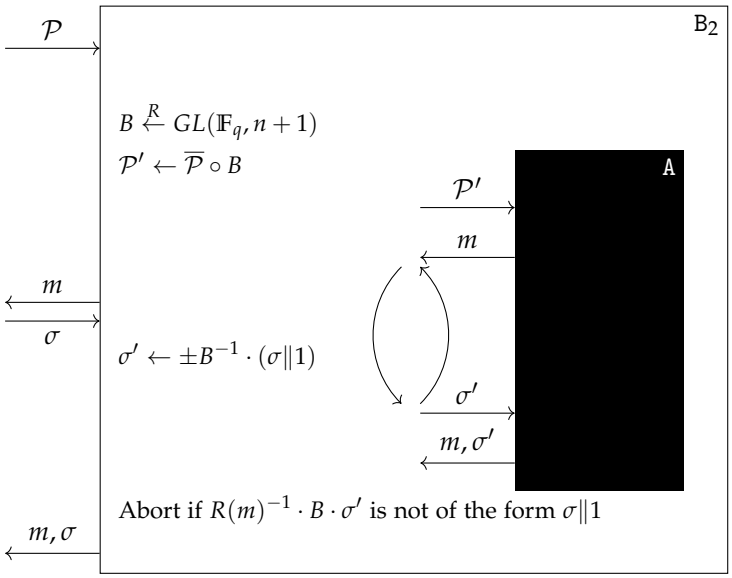


Figure 3.5: Reduction from breaking EUF-CMA security of $\mathsf{UOV}_q(n,m)$ to EUF-CMA security of $\mathsf{homogenousUOV}_q(n+1,m)$, without the ROM. The instruction $\sigma' \leftarrow \pm B^{-1}\cdot(\sigma\|1)$ means that σ' is set to $B^{-1}\cdot(\sigma\|1)$ with probability $1/2$, and $-B^{-1}\cdot(\sigma\|1)$ otherwise.

A standard reduction in the ROM. Since homogenousUOV is a FDH-type scheme, there is an approach for obtaining a reduction from breaking its EUF-CMA security to simply breaking its EUF security — the reduction algorithm $\mathtt{B}_3$ would simply, upon recieving a signing query m, pick a random $\mathbf{x} \in \mathbb{F}_q^n$, set $\mathbf{y} = \mathcal{P}(\mathbf{x})$, send $\mathbf{x}$ to its black box as a

signature, and story in memory that if its black box ever asks for the hash of m, it will return that this hash is $\mathbf{y}$ — this forces the signatures to be correct with respect to the simulated random oracle. Oracle queries would be handled analogously. This approach presents two issues:

- With overwhelming probability, the resulting forgery will not be valid with respect to the true hash. This can be solved with an application of the forking lemma ([4]). However, this introduces very significant loss in advantage — if the EUF forger succeeds with probability p, $\mathtt{B}_3$ can only be quaranteed to succeed with probability $\approx p^2/N$, where N is the maximum amount of oracle queries answered.
- It is not clear whether the distribution of the signatures/hashes given to the black box will follow that of a legitimate challenger — in short, this is to do whether one can distinguish whether $(input, output)$ pairs for homogenousUOV public keys have been generated output-first (with the secret key) or input-first. The necessary hardness assumption is would be that UOV, without the hashing, is a *preimage sampleable trapdoor* — this notion is defined in [15, §5.3].

Note that this (and the following approach) does not work if the hashes are not salted, as $\mathtt{B}_4$ will be unable to commit to two different signatures to the same message (once it has commited to a fake hash).

A modification of $\mathtt{B}_1$ that attempts to hide $B^{-1}(*\|0)$. A fourth possibility, $\mathtt{B}_4$, would upon recieving any query, with probability $1 - 1/q$ proceed like $\mathtt{B}_1$, but otherwise proceed like $\mathtt{B}_3$ and produce a fake signature in $B^{-1}(*\|0)$. In this approach, the validity of $\mathtt{B}_4$ would need to be supported by both the hardness of MH and UOV being preimage sampleable — however, the two problems are now combined, in such a way that what A would actually need to do for the reduction to fail is identify a hidden hyperplane where signatures are generated dishonestly. This certainly *sounds* harder, but it is also significantly more convoluted, to the point that it does not seem feasible to prove that this reduction is secure without some very specific ad-hoc hardness assumptions.

Chapter 4

Subspace encoding

In the last chapter we found that, although traditionally the secret key of UOV is described by a linear map $\mathcal{T} : \mathbb{F}_q^m \to \mathbb{F}_q^n$, this can be reformulated in such a way that we only need to know $\mathcal{T}(\{0\}^{n-m} \times \mathbb{F}_q^m)$, a m-dimensional subspace of $\mathbb{F}_q^n$. This evidently carries less information than all of $\mathcal{T}$, which leads to considering exactly how much one can reduce the amount of information necessary to describe the secret key — this will be the subject of this chapter.

As of right now, a description of $\mathcal{T}$ is simply a matrix of $\mathcal{T}$, as there is no ambiguity about the base being used for either $\mathbb{F}_q^m$ or $\mathbb{F}_q^n$ — this takes $m \cdot n$ elements of $\mathbb{F}_q$ to describe. Note that $\mathbb{F}_q$ is the natural choice of alphabet to describe elements of $\mathcal{M}(\mathbb{F}_q, m, n)$, and thus also linear maps $\mathbb{F}_q^n \to \mathbb{F}_q^m$. Now, however, we are only interested in representing linear subspaces of a certain dimension, say k, i.e. elements of $Gr(k, \mathbb{F}_q^n)$. It will be handy to keep using the same alphabet $\mathbb{F}_q$ — and so our objective becomes to find an "optimal" encoding of $Gr(k, \mathbb{F}_q^n)$ using $\mathbb{F}_q$. There are three things to note about this goal:

- Using $\mathbb{F}_q$ as an alphabet is the natural choice for any tensor with entries in $\mathbb{F}_q$, but for any practical applications it would be preferable to use $[q]$ as an alphabet. A canonical bijection $\mathbb{F}_q \to [q]$ exists only if q is prime, however. For $q = p^r$ with $r \neq 1$, a way to select a particular bijection could be to single out a particular irreducible polynomial P in $\mathbb{F}_p[X]$ (for instance, the first such polynomial lexicographically), and then consider the canonical bijection $\mathbb{F}_q \cong \mathbb{F}_p[X]/\langle P \rangle \xrightarrow{1} \mathbb{F}_p^r \xrightarrow{2} [p]^r \xrightarrow{3} [p^r] = [q]$, where the intermediate maps correspond to:

  1. Reading a polynomial with no terms of degree $\geq r$ as a tuple of its coefficients,
  2. Element-wise application of the canonical bijection $\mathbb{F}_p \to [p]$,
  3. Reading an r-tuple of elements from $[p]$ as an r-digit number in base p.

  We only specify this for completeness — to avoid confusion, however, we will simply use $\mathbb{F}_q$ as the alphabet for the remainder of this chapter. In examples, q will always be prime to avoid having to pick a particular representation of $\mathbb{F}_q$.

- As far as practical application goes, it would obviously be better to simply use $\{0,1\}$ as alphabet. We will not do this — if necessary, it would suffice to compose our code with any (uniquely decodable) encoding $[q] \to [2]$.

- It is not immediately clear what we mean by "optimal." If we had some information about the distribution of elements form $Gr(\mathbb{F}_q^n, k)$ that we'd be encoding, we might try to minimize expected word length. For the application in MVQC, this distribution should presumably be uniform, and thus we would be minimizing average word length. Nonetheless, this is **not** what we pursure here, instead trying to minimize *maximum* word length.

  Remark. This is not as much of a concession as it might seem at first — indeed, say ℓ is the shortest possible maximum word length for encoding some set S with an alphabet Σ with m symbols. This must mean that $m^{\ell-1} < \#S$. Now, assume we just build an encoding $S \to \Sigma^*$ by greedily assigning to every element the shortest codeword available, with no regard for whether the final code is uniquely decodable — clearly the average word length of this will be a lower bound for the average word length of any code. So we compute this lower bound:

$$\sum_{i=0}^{\ell-1} i \cdot m^i + \ell(\#S - m^{\ell-1}) \geq (\ell-1)\frac{m^{\ell-1}}{\#S} + \ell\frac{\#S - m^{\ell-1}}{\#S} = \ell - \frac{m^{\ell-1}}{\#S} > \ell - 1$$

  So the optimal average word length cannot be a symbol lower than the optimal maximum word length.

So we aim to find the smallest ℓ such that there exists an encoding $Gr(k, \mathbb{F}_q^n) \to [q]^\ell$. This must be an injection and thus $\#Gr(k, \mathbb{F}_q^n) \leq \#([q]^\ell) = q^\ell \implies \ell \geq \lceil \log_q \#Gr(k, \mathbb{F}_q^n) \rceil$. This will be our first step — using this expression to investigate the value of ℓ.

4.1 Theoretical bounds

So we begin by interesting ourselves in $\lceil H \rceil$, where $H = H_q^{k,n} := \log_q \#Gr(k, \mathbb{F}_q^n)$, for $0 \leq k \leq n$ and $q \geq 2$. The cases $0 = k$ and $n = k$ are both trivial ($H = 0$, as there is only one subspace in both cases), so we assume that $0 < k < n$. Then: $\#Gr(k, \mathbb{F}_q^n)$ is the q-Gaussian binomial coefficient, often written as $\binom{n}{k}_q$, and it is well known (and easy to verify) that

$$\binom{n}{k}_q = \frac{(q^n-1)(q^{n-1}-1)\cdots(q^{n-k+1}-1)}{(q-1)(q^2-1)\cdots(q^k-1)}.$$

Therefore, grouping terms appropiately:

$$H = \log_q \frac{q^n-1}{q^k-1} + \log_q \frac{q^{n-1}-1}{q^{k-1}-1} + \cdots + \log_q \frac{q^{n-k+1}-1}{q-1} = \sum_{i=0}^{k-1} \log_q \frac{q^{n-i}-1}{q^{k-i}-1}.$$

At this point, it seems natural to try making an approximation inspired by

$$\frac{q^{n-i}-1}{q^{k-i}-1} \approx \frac{q^{n-i}}{q^{k-i}} = q^{n-k}.$$

This yields a fairly neat expression:

$$H \approx \sum_{i=0}^{k-1} \log_q q^{n-k} = \sum_{i=0}^{k-1} n-k = k(n-k)$$

This will be our jumping-off point for tighter bounds on H, though we will see that, in fact, it is already pretty close. One can realize that it's not surprising that this should be the case: indeed, consider representing a subspace as the row-span of a $k \times n$ matrix M. Write $M = (P \mid Q)$, where P is the first $k \times k$ minor of M. With rather high probability ($\approx 1 - q^{-1}$, as noted in [26]), P will be of full rank, meaning that we can consider $M' = P^{-1}M = P^{-1}(P \mid Q) = (Id \mid P^{-1}Q)$. Since $M = PM'$ and P is a full-rank matrix, M and M' have the same row-span, but only $k(n-k)$ elements of $\mathbb{F}_q$ are needed to represent M', as we only need to store the last $n-k$ columns. Nonetheless, it holds that $H > k(n-k)$, which owes to the fact that P can occasionally be singular — but both in getting closer to the true value of H, and finding actual practical encodings with word-length $\lceil H \rceil$, we will proceed with the idea above as a starting point.

Following this philosophy, then:

$$\begin{aligned}
H - k(n-k) &= \sum_{i=0}^{k-1} \log_q \left(\frac{q^{n-i}-1}{q^{k-i}-1} \right) - \log_q(q^{n-k}) \\
&\overset{*}{\leq} \sum_{i=0}^{k-1} \frac{q^{n-i}-1}{q^{n-k}(q^{k-i}-1)} - 1 \\
&= \sum_{i=0}^{k-1} \frac{q^{n-k}-1}{q^{n-k}(q^{k-i}-1)} \\
&\leq \sum_{i=0}^{k-1} \frac{q^{n-k}}{q^{n-k}(q^{k-i}-1)} \\
&= \sum_{i=0}^{k-1} \frac{1}{q^{k-i}-1} \\
&\leq \sum_{i=1}^{\infty} \frac{1}{q^{i}-1}
\end{aligned}$$

Where the marked inequality comes from the fact that, for $0 < x \leq y$, we have that $\log(y) - \log(x) = \log \frac{y}{x} \leq \frac{y}{x} - 1$. Now, it is rather easy to see that, for $q > 2$, the latter sum converges to a value < 1. Indeed, given that $i > 1$ and $q > 2$, clearly $q^i - 1 > (q-1)^i$,

and therefore

$$\sum_{i=1}^{\infty} \frac{1}{q^i - 1} < \sum_{i=1}^{\infty} \frac{1}{(q-1)^i} \leq \sum_{i=1}^{\infty} \frac{1}{2^i} = 1.$$

So we have that, for $q > 2$, $H_q^{n,k} > k(n-k)$, but also $H_q^{n,k} - k(n-k) < 1$ — from this it follows that $\lceil H_q^{n,k} \rceil = k(n-k) + 1$ for $q > 2$. As per usual, the $q = 2$ case has to be handled separately:

4.1.1 The case of $\mathbb{F}_2$

For $q = 2$, simply evaluating the first term already tells us that $\sum_{i=1}^{k} (q^i - 1)^{-1} > 1$. However, some cruder analogues of the bounds above yield that $\sum_{i=1}^{k} (2^i - 1)^{-1} < 2$, so we can claim to know that $k(n-k) < H_q^{n,k} < k(n-k) + 2$. This tells us that $\lceil H_q^{n,k} \rceil = k(n-k) + M$, for $M \in \{1, 2\}$ — but we do not know which.

It will turn out that, in most cases (except for some specific small values of k and $n-k$), $M = 2$, that is, $H_2^{n,k} - k(n-k) > 1$. The proof of this inequality is rather uninteresting — focusing on the expression of $H - k(n-k)$ as a summation, it is easy to notice that all of its summands are positive, and thus we will show that the sum is > 1 by picking some fixed amount of terms that already add up to > 1. For this, it is convenient to slightly rearrange the summation:

$$\begin{aligned}
H - k(n-k) &= \sum_{i=0}^{k-1} \log_2 \left(\frac{2^{n-i} - 1}{2^{k-i} - 1} \right) - \log_2(2^{n-k}) \\
&= \sum_{i=0}^{k-1} \log_2 \left(\frac{2^{n-i} - 1}{2^{n-k}(2^{k-i} - 1)} \right) \\
&= \sum_{i=0}^{k-1} \log_2 \left(\frac{2^{(n-k)+(k-i)} - 2^{(k-i)-(k-i)}}{2^{n-k}(2^{k-i} - 1)} \right) \\
&= \sum_{i=0}^{k-1} \log_2 \left(\frac{2^{k-i}}{2^{k-i} - 1} \cdot \frac{2^{n-k} - 2^{-(k-i)}}{2^{n-k}} \right) \\
&= \sum_{i=0}^{k-1} \log_2 \left(\frac{2^{k-i}}{2^{k-i} - 1} \right) + \log_2 \left(1 - 2^{-(n-i)} \right)
\end{aligned}$$

For convenience, we switch to $j = k - i$ indices, and introduce the change of variables $(k, r) = (k, n-k)$, such that $n = r + k$ and $0 < k < n \iff 0 < k, r$:

$$H - kr = \sum_{j=1}^{k} \log_2 \left(\frac{2^j}{2^j - 1} \right) + \log_2 \left(1 - 2^{-(r+j)} \right)$$

Observe that $\log_2(1 - 2^{-x})$ is increasing w.r.t. x, and therefore $\log_2(1 - 2^{-(r+j)})$ is increasing w.r.t. r, meaning that $f(k, r) := H_2^{r+k,k} - kr$ is increasing w.r.t. r. Moreover, both of the terms in $f(k, r)$ are symmetric w.r.t. (k, r), that is, $f(k, r) = f(r, k)$, meaning that $f(k, r)$ is increasing w.r.t. both r and k. With this in mind, we can evaluate at $k = r = 2$ by hand

(computer) to find that $f(2,2) = \log_2(35/16) > \log_2(2) = 1$. Since f is increasing w.r.t. both of its variables and it is bounded by two, this implies that $\lceil f(k,r) \rceil = 2$ for $k, r > 2$. As for $r = 1$ or $k = 1$, note that

$$\begin{aligned} f(r,1) = f(1,r) &= \log_2\left(\frac{2^1}{2^1-1}\right) + \log_2\left(1 - 2^{-(r+1)}\right) \\ &= 1 + \log_2\left(1 - 2^{-(r+1)}\right) < 1, \end{aligned}$$

Therefore, these are the conclusions that we can draw about $\lceil H_2^{r+k,k} \rceil - kr$:

r \ k	1	2	3	4	5	6	7	8	9
4	1	2	2	2	2	2	2	2	2
3	1	2	2	2	2	2	2	2	2
2	1	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	1	1

Where both tables can be extended indefinitely by repeating the edges, and for the latter table we used the knowledge that $f(k,r) < 2$. In any case, the takeaway is that $H_2^{n,k} < k(n-k) + 2$, and in fact $\lceil H_2^{n,k} \rceil = k(n-k) + 2$ for most values of k, n — so we will attempt to encode elements of $\binom{\mathbb{F}_2^n}{k}$ with $k(n-k) + 2$ bits (i.e. 2-registers).

Remark. In fact, in the context of MVQC, those values for which $\lceil H_2^{n,k} \rceil = k(n-k) + 1$ will actually correspond to parameter sets for which UOV is insecure. In any case, it is easy to provide an optimal encoding when $k =\in \{1, n-1\}$ — these correspond to lines and hyperplanes, both of which can be represented by a single (generating or perpendicular, respectively) vector in $\mathbb{F}_q^n$.

4.2 Concrete encodings

At this point, after having established that $\ell := \lceil H_q^{n,k} \rceil = k(n-k) + M$, for $M = 1$ when $q > 2$ and $M \in \{1, 2\}$ when $q = 2$, we know that there must exist a way to encode subspaces from $Gr(k, \mathbb{F}_q^n)$ with ℓ elements from $\mathbb{F}_q$. Naïvely, constructing an actual encoding is not difficult — one need only find some way to enumerate $Gr(k, \mathbb{F}_q^n)$, and pair the elements with elements from $[q]^\ell$ according to the canonical enumeration of the latter.

However, since our search for this encoding is motivated by a practical application, we should look for a practical encoding — i.e. one that is reasonably efficient to compute and uniquely decodable. In fact, we will look for a prefix code — this is easier to work with, will be useful for a construction later on, and by the Kraft-McMillan inequalities (Thms. 2.1, 2.2) this requirement does not constrain the resulting word-lengths.

The core idea is that an element of $Gr(k, \mathbb{F}_q^n)$ can be represented as the row-span of a (full-rank) $k \times n$ matrix, and moreover, the row-span of a matrix is invariant with respect to left-multiplication by nonsingular matrices — in particular, it is invariant with respect to elementary row operations. Now, any matrix can be put into *reduced row echelon form* (RREF) by a series of elementary row operations, and thus any subspace can be represented by a matrix in this form — our approach, then, will be to find an optimal encoding of matrices in RREF, and use this to encode elements of $Gr(k, \mathbb{F}_q^n)$.

4.2.1 Reduced row echelon form

RREF is a rather elementary concept, but we will give a refresher for the sake of establishing terminology (and a slight convention): a $k \times n$ matrix M is in **reduced row echelon form** if there exist indices $0 < x_1 < \cdots < x_k \in [n]$, such that:

$$M_{ij} = \begin{cases} 1 & \text{if } j = x_i \\ 0 & \text{if } \exists i' > i \text{ with } j = x_{i'} \\ 0 & \text{if } j < x_i \\ * & \text{otherwise} \end{cases}$$

Where $*$ means that the corresponding entry of M can be any element of $\mathbb{F}_q$. We give an example of a matrix from $\mathcal{M}(\mathbb{F}_{11}, 5, 10)$ in RREF:

$$\begin{pmatrix} 1 & 0 & 2 & 0 & 4 & 8 & 6 & 0 & 0 & 1 \\ 0 & 1 & 5 & 0 & 7 & 2 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 1 & 3 & 9 & 6 & 0 & 0 & 8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 9 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 3 \end{pmatrix}$$

In this case, $(x_1, \ldots, x_5) = (1, 2, 4, 8, 9)$. In general, for any matrix in RREF, we will denote by x^M the associated sequence of indices (note that any positive strictly increasing sequence $0 < x_1 < \cdots < x_k$ will have $x = x^M$ for some matrix M). Below is what a generic matrix $M \in \mathcal{M}(\mathbb{F}_{11}, 5, 10)$ in RREF with $x^M = (1, 2, 4, 8, 9)$ can look like:

$$\begin{pmatrix} 1 & 0 & * & 0 & * & * & * & 0 & 0 & * \\ 0 & 1 & * & 0 & * & * & * & 0 & 0 & * \\ 0 & 0 & 0 & 1 & * & * & * & 0 & 0 & * \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & * \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & * \end{pmatrix}$$

Where any specific matrix will replace the $*$ entries with some elements in $\mathbb{F}_q$. We will denote this matrix (with entries in $\{0, 1, *\}$, $*$ being a formal symbol) as $RREF_x^{k,n}$, and write $M \in RREF_x^{k,n}$ if M is a $k \times n$ matrix in RREF with $x^M = x$.

Remark. Our definition of RREF is different from the usual definition in that it cannot represent non-full-rank matrices — usually the definition would allow for $x^M = (x_1, \ldots, x_r)$, with $r < k$, and having that $M_{ij} = 0$ for any $i > r$. This does not affect us, however, because we are only interested in full-rank matrices, as otherwise their row-span would not be a k-subspace of $\mathbb{F}_q$. Nonetheless, it is not difficult to expand the analysis of RREF matrix encoding to include non-full-rank matrices, but this is left out of this text as it is not relevant and would complicate the remainder of this chapter.

4.2.2 Encoding matrices in RREF

We begin with a slightly strange definition: for a positive strictly increasing sequence x, we denote $\tilde{x}$ the sequence of the same length such that $\tilde{x}_i = x_i - i$. Note that $\tilde{x}$ will be nonnegative and increasing. The interesting fact about this substitution is the following:

Proposition 4.1. *Let $x = x^M$, with M a $k \times n$ matrix in RREF. Then, the amount of $*$ symbols in $RREF_x^{k,n}$ is exactly $k(n-k) - \sum \tilde{x}$.*

Proof. We prove that each $i-th$ row of M has $n - k - \tilde{x}_i$ $*$ symbols — the result then directly follows by taking the sum over $i = 1, \ldots, k$. So indeed, fix an i and consider the row $(M_{ij})_j$. Note that, for $j < x_i$, $M_{ij} = 0$, and for $j = x_i$, $M_{ij} = 1$. So we only need to focus on $j > x_i$, that is, $j = x_i + 1, \ldots, n$. With this restriction, only two cases of the definition of M_{ij} can apply — either $M_{ij} = 0$, if $\exists i' > i$ with $j = x_{i'}$, or $M_{ij} = *$ otherwise. The first case will apply once for each $i' = i + 1, \ldots, x_k$, meaning that it will apply $k - i$ times. Therefore, the $M_{ij} = *$ case will apply $(n - x_i) - (k - i) = (n - k) - (x_i - i) = n - k - \tilde{x}_i$ times, as we wanted to show. □

Having this proposition, we know that, for some fixed k, n and positive strictly increasing sequence x, optimally encoding some $M \in RREF_x^{k,n}$ must take $k(n-k) - \sum \tilde{x}$ elements of $\mathbb{F}_q$ — this encoding $\langle M|x\rangle$ ([1]) will just be the elements of M corresponding to $*$ symbols in $RREF_x^{k,n}$ in some particular order (we default to column-major). So, if we want to encode M as an arbitrary matrix in RREF, we might do this by finding some encoding $\langle x\rangle$ of x, and appending $\langle M|x\rangle$ to this encoding — and so $\langle M\rangle = \langle x\rangle\langle M|x\rangle$, with juxtaposition denoting concatenation. After a specific example, we will see that this construction of "code composition" is valid (and uniquely decodable) in general.

[1] this notation should be read as "the encoding of M with previous knowledge of x"

Example. Consider again the same matrix as earlier, with $x^M = (1, 2, 4, 8, 9)$

$$M = \begin{pmatrix} 1 & 0 & 2 & 0 & 4 & 8 & 6 & 0 & 0 & 1 \\ 0 & 1 & 5 & 0 & 7 & 2 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 1 & 3 & 9 & 6 & 0 & 0 & 8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 9 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 3 \end{pmatrix} \quad RREF_x^{5,10} = \begin{pmatrix} 1 & 0 & * & 0 & * & * & * & 0 & 0 & * \\ 0 & 1 & * & 0 & * & * & * & 0 & 0 & * \\ 0 & 0 & 0 & 1 & * & * & * & 0 & 0 & * \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & * \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & * \end{pmatrix}$$

We would have that $\langle M \rangle = \langle x \rangle \langle M|x \rangle = \langle x \rangle$ 25 473 829 606 15893, with the spaces only being for clarity. To decode this, one would first read $\langle x \rangle$, construct $RREF_x^{5,10}$, and then fill the $*$ elements in column-major order as read from $\langle M|x \rangle$.

This encoding would take $|\langle M \rangle| = |\langle x \rangle \langle M|x \rangle| = |\langle x \rangle| + |\langle M|x \rangle| = |\langle x \rangle| + k(n-k) - \sum \tilde{x}$ symbols from $\mathbb{F}_q$. According to the results of §4.1, for optimality we should have that

$$k(n-k) + M \geq |\langle M \rangle| = |\langle x \rangle| + k(n-k) - \sum \tilde{x} \implies |\langle x \rangle| \leq \sum \tilde{x} + M.$$

So, for this to be a viable strategy, we need to find a way to encode increasing strictly positive sequences x with $|\langle x \rangle| \leq \sum \tilde{x} + M$.

On the viability of encoding positive strictly increasing sequences

Though not strictly necessary, as we will directly construct such encoding, it is somewhat informative to, through Kraft's inequality (Thm.2.1), verify that such encodings can exist. So — to encode x, it is easier to work with $\tilde{x}$, and it is easier to generalize this to encoding strictly positive increasing sequences of *any* length. If such a sequence x must be encoded with $\sum x + M$ characters, then the sequences encoded with ℓ characters will be those that sum to $\ell - M$, and thus Kraft's inequality tells us that this encoding will exist if

$$\sum_{n=0}^{\infty} \#\{\text{increasing sequences adding up to } n - M\} \cdot q^{-n} \overset{?}{\leq} 1$$

Now, note that positive increasing sequences adding up to $\ell - M$ are in bijection with positive multisets adding up to $\ell - M$ — indeed, an increasing sequence may be regarded simply as a particular representation of a multiset. More importantly, the amount of multisets adding up to some number n is the well-known[2] **partition function** $p(n)$. With this, we rewrite the inequality above as

$$q^{-M} \sum_{n=0}^{\infty} p(n) \cdot q^{-n} \overset{?}{\leq} 1$$

Note that the summation is the generating function of $p(n)$ evaluated at $1/q$ — a useful way to rewrite this generating function is $1/\sum_{k \in \mathbb{Z}} (-1)^k \cdot q^{-k \cdot (3k-1)/2}$, and it is not difficult to, leveraging this expression, show that the optimum integer values of M for the bound

[2] For reference, a whole chapter is devoted to this function in [1, §14]. In particular, §14.3 deals with generating functions of $p(n)$, one of which we will be using shortly.

above to hold are $M = 2$ when $q = 2$, and $M = 1$ for $q > 2$.

Remark. In [26], Waterhouse points out that this expression, the generating function of $p(n)$ at $1/q$, is the reciprocal of (the limit as $n \to \infty$ of) the proportion of nonsingular matrices in $\mathcal{M}(\mathbb{F}_q, n, n)$. Therefore, the bound above holds when this proportion is greater than q^{-M}. Interestingly, although our current concerns are far removed from matrices[3], this echoes our observation from §4.1¶5 that the only reason that M must be > 0 at all is that, over a finite field, a non-negligible proportion of matrices will have a null determinant.

4.2.3 Code composition

We describe a more general version of the construction above, and specify conditions with which this construction can be shown to be uniquely decodable:

Definition 4.2. *Given a set F, an encoding $\phi\colon F \to \Sigma^*$, a family of sets indexed by F, $\{S_x\}_{x\in F}$, and for each $x \in F$, an encoding $\phi_x\colon S_x \to \Sigma^*$, the **composition** of ϕ and $\{\phi_x\}_{x\in F}$ is the code*

$$\begin{aligned}\Phi\colon \{(x,y) \mid x \in F, y \in S_x\} &\to \Sigma^* \\ (x,y) &\mapsto \phi(x)\phi_x(y).\end{aligned}$$

Proposition 4.3. *Let ϕ and $\{\phi_x\}_{x\in F}$ be as above. If all of these are prefix codes, then their composition Φ is also a prefix code.*

Proof. By contrapositive: assume there is some $(x_1, y_1) \neq (x_1, y_2)$ such that $\Phi(x_1, y_1)$ is a prefix of $\Phi(x_2, y_2)$, that is, $\phi(x_1)\phi_{x_1}(y_1)$ is a prefix of $\phi(x_2)\phi_{x_2}(y_2)$. Notice that either $\phi(x_1)$ is a prefix of $\phi(x_2)$, in which case ϕ is not a prefix code and we are finished, or $\phi(x_1) = \phi(x_2)$, and thus $x_1 = x_2 =: x$ and we have that $\phi_x(y_1)$ is a prefix of $\phi_x(y_2)$, and thus ϕ_x is not a prefix code. □

In the application relevant to this chapter, for fixed q, k, n, ϕ would be an encoding of strictly increasing k-tuples from $[1, n]$ using $\mathbb{F}_q$ as an alphabet, and for one such tuple x, $\phi_x\colon RREF_x(k, n) \to \mathbb{F}_q$ simply encodes a matrix M by writing its elements corresponding to $*$ entries from $RREF_x(k, n)$ in column-major order. This is a code in which every codeword is of the same length, and thus it must be a prefix code — so to see that the code composition is a prefix code, we only need to find a prefix code for describing strictly increasing k-tuples in $[1, n]$.

4.2.4 Encoding positive strictly increasing sequences

Immediately, something about the requirement that $|\langle x\rangle| \leq \sum \tilde{x} + M$ might seem noteworthy. If we intend to encode elements of $Gr(k, \mathbb{F}_q^n)$, for this we will need to encode increasing k-tuples of elements from $[n]$. The set of tuples that we're trying to encode doesn't depend on q, but our bound on the length, $\sum \tilde{x} + M$, does — usually $M = 1$, but

[3] A similar but more detailed observation has already been made by Knuth, regarding the connection between subspaces and partitions, in [18].

if $q = 2$ then $M = 2$. We start with the $q = 2$ case.

It turns out that finding an encoding like the one we want is not particularly hard. To encode x, we will consider $\tilde{x}$ — and, since x (and thus also $\tilde{x}$) is prescribed to be of length k, we will be ignoring all leading zeroes of $\tilde{x}$, as they can be recovered later by simply adding them back until we have k elements. Indeed, let i be the first index such that $\tilde{x}_i > 0$ (or $k+1$, if no such index exists), and moreover, s be the smallest integer such that $\tilde{x}_{i+s} > 1$ (or $k+1-i$, if no such integer exists) — put another way, s is the amount of leading ones in $\tilde{x}$. With these two values in mind, our encoding will be:

$$\langle x \rangle = 1^s 0 \Big(\prod_{j=i+s}^{k} 1^{\tilde{x}_j - 1} 0 \Big) 0$$

We need to check that this code has the following three properties:

- It is a prefix code. Indeed, for any sequence x, $\langle x \rangle$ ends with 00, and there can be no other 00 in $\langle x \rangle$, on account of the fact that $\tilde{x}_j - 1 > 0$, since $j \geq i + s$.
- It is a valid code, i.e. injective. Indeed, it is easy to recover $\tilde{x}$, and therefore x, from $\langle x \rangle$ — we can always find s to be the length of the starting sequence of 1s, and $\tilde{x}_j - 1$ to be the length of successive sequences. This obviously allows one to recover all $\tilde{x}_j > 2$, after which one only need prepend s ones, and enough zeroes for the resulting chain to have length k.
- Its length satisfies the expected bound. Indeed:

$$\begin{aligned}
|\langle x \rangle| &= |1^s 0| + |\Big(\prod_{j=i+s}^{k} 1^{\tilde{x}_j - 1} 0 \Big)| + |0| \\
&= s + 1 + \Big(\sum_{j=i+s}^{k} \tilde{x}_j - 1 + 1 \Big) + 1 \\
&= \underbrace{\sum_{j=1}^{i-1} \tilde{x}_j}_{0} + \underbrace{\sum_{j=i}^{i+s-1} \tilde{x}_j}_{s} + \sum_{j=i+s}^{k} \tilde{x}_j + 2 \\
&= \sum \tilde{x}_j + 2
\end{aligned}$$

On the other hand, the case of $q > 2$ allows us to use a larger alphabet for our encoding, at the cost of having one less q-register to work with. It is rather easy to adapt our previous strategy to work for this case — let g be any element in our alphabet different from $0, 1$, and let i and s be as before. Then:

$$\langle x \rangle = g^s \Big(\prod_{j=i+s}^{k} 1^{\tilde{x}_j - 1} 0 \Big) 0$$

It is easy to see, with arguments analogous to the ones from the $q = 2$ case, that this is a valid prefix code. Moreover, it uses one less symbol, and thus $|\langle x \rangle| = \sum \tilde{x} + 1$, as expected.

Example. We give an instance of the full encoding process for $q > 2$. Say we want to encode the element of $Gr(5, \mathbb{F}_{11}^{10})$ represented as the row-span of the following matrix:

$$M = \begin{pmatrix} 2 & 0 & 4 & 3 & 6 & 10 & 8 & 4 & 9 & 1 \\ 9 & 8 & 3 & 1 & 7 & 9 & 5 & 1 & 4 & 1 \\ 6 & 0 & 1 & 6 & 9 & 3 & 6 & 0 & 4 & 0 \\ 8 & 10 & 0 & 0 & 3 & 7 & 4 & 10 & 9 & 10 \\ 2 & 0 & 4 & 8 & 10 & 0 & 5 & 1 & 9 & 3 \end{pmatrix}$$

We begin by putting the matrix in RREF, which takes $O(k^2 n)$ operations:

$$M = \begin{pmatrix} 1 & 0 & 2 & 0 & 4 & 8 & 6 & 0 & 0 & 1 \\ 0 & 1 & 5 & 0 & 7 & 2 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 1 & 3 & 9 & 6 & 0 & 0 & 8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 9 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 3 \end{pmatrix}$$

From here we extract that $x := x^M = (1, 2, 4, 8, 9)$. As per our definition, $\langle M \rangle = \langle x \rangle \langle M|x \rangle$. We saw in the earlier example that $\langle M|x \rangle = 25\,473\,829\,606\,15893$. As for $\langle x \rangle$, we begin by computing $\tilde{x} = (0, 0, 1, 4, 4)$, and selecting a $g \in \mathbb{F}_{11}$ different from both 0 and 1 — for instance, 2. Then, we set $i = 3$, the first nonzero index, and $s = 1$, the amount of leading ones. With this, we have:

$$\langle x \rangle = 2^s 1^{x_4 - 1} 01^{x_5 - 1} 00 = 2^1 1^3 01^3 00 = 2111011100$$

And so, $\langle M \rangle = \langle x \rangle \langle M|x \rangle = 2111011100\ 25\,473\,829\,606\,15893$

4.2.5 Encoding affine subspaces

Having found an optimal encoding of k-subspaces of $\mathbb{F}_q^n$, we now use this as a jumping-off point to find likewise optimal encodings of a related object — affine subspaces of $\mathbb{F}_q^n$.

For each k-dimensional linear subspace V of $\mathbb{F}_q^n$, there exist q^{n-k} affine subspaces of the form $p + V$ — precisely the elements of $\mathbb{F}_q^n / V$. It follows, then, that the amount of k-dimensional affine subspaces of $\mathbb{F}_q^n$ should be $q^{n-k} \cdot \#Gr(k, \mathbb{F}_q^n)$, and therefore the amount of q-registers necessary to encode one such affine subspace should be

$$\lceil \log_q \left(q^{n-k} \cdot \#Gr(k, \mathbb{F}_q^n) \right) \rceil = n - k + \lceil H_q^{n,k} \rceil = n - k + k(n-k) + M,$$

where $H_q^{n,k}$ is as in §4.1, and $M = 2$ when $q = 2$ and $M = 1$ otherwise.

It turns out that we can readily find an encoding using this amount of q-registers. For this, consider the inclusion π from $\mathbb{F}_q^n$ to its projective closure $\mathbb{P}(\mathbb{F}_q \times \mathbb{F}_q^n) \cong \mathbb{P}(\mathbb{F}_q^{n+1})$, given

by $\pi\colon (x_1, \ldots, x_n) \mapsto (1 : x_1 : \cdots : x_n)$. Recall that this projection will send[4] k-dimensional affine subspaces of $\mathbb{F}_q^n$ to k-dimensional projective subspaces of $\mathbb{P}(\mathbb{F}_q^{n+1})$, which correspond to $(k+1)$-dimensional linear subspaces of $\mathbb{F}_q^{n+1}$. Restating this more explicitly — we are associating each k-dimensional affine variety in $\mathbb{F}_q^n$, affinely generated by a set of points $(p^1, \ldots, p^{k+1})$, with coordinates $p^i = (p_1^i, \ldots, p_n^i)$, to the $(k+1)$-dimensional linear subspace of $\mathbb{F}_q^{n+1}$ generated by $(v^1, \ldots, v^{k+1})$, where $v^i = (1, p_1^i, \ldots, p_n^i)$.

Now, each k-dimensional affine subspace of $\mathbb{F}_q^n$ can be represented as a $(k+1)$-dimensional linear subspace of $\mathbb{F}_q^{n+1}$. Encoding these will take $(k+1)(n+1-(k+1)) + M$ q-registers, which is the amount that we wanted.

Remark. There will be some redundancy if one directly applies the methods of linear subspace encoding to obtain affine subspace encoding — for instance, since the vectors generating the subspace of $\mathbb{F}_q^{n+1}$ will all be of the form $(1, p_1^i, \ldots, p_n^i)$, inserting them as the rows in a $(k+1) \times (n+1)$ matrix and performing row reduction to obtain a matrix in RREF will always have the following first step:

$$\begin{pmatrix} 1 & p_1^1 & \ldots & p_n^1 \\ 1 & p_1^2 & \ldots & p_n^2 \\ \vdots & \vdots & \vdots\vdots\vdots & \vdots \\ 1 & p_1^{k+1} & \ldots & p_n^{k+1} \end{pmatrix} \to \begin{pmatrix} 1 & p_1^1 & \ldots & p_n^1 \\ 0 & p_1^2 - p_1^1 & \ldots & p_n^2 - p_n^1 \\ \vdots & \vdots & \vdots\vdots\vdots & \vdots \\ 0 & p_1^{k+1} - p_1^1 & \ldots & p_n^{k+1} - p_n^1 \end{pmatrix}.$$

If one intends to actually use this method to encode affine subspaces, it would be sensible to tease out every such redundancy. Though we do not prove this, we claim that the process described above to represent an affine subspace $p + V$ is tantamount to expressing V as the row-span of a matrix in RREF M, and p as a vector such that $p_{x_1^M} = \cdots = p_{x_k^M} = 0$, and then encoding M followed by the nonzero coordinates of p.

4.3 Applicability to MVQC

While there is clear applicability and potential benefit in using the material of this chapter in UOV, there are two notable shortcomings with this idea that should be taken into consideration. We will discuss them in this section:

Secret keys can already be (efficiently) represented by their seed

In UOV, the trapdoor information necessary to invert the map described by the public key is a subspace. We have shown how to represent this subspace in a space-optimal way — however, this subspace is the *secret* key of UOV. This presents a problem: the holder of the secret key is often, especially in the case of traditional signature schemes, the one who generated the keys in the first place, and thus someone who knows the seed that

[4] note that this is an injection, not a bijection. The linear subspaces obtained this way cannot b te contained in the "hyperplane at infinity" $\{0\} \times \mathbb{F}_q^n$ — however, we will also be able to encode affine subspaces in this hyperplane, if for whatever reason this kind of object from the projective closure is useful to us.

was input to `keyGen` (or, in any case, someone who could reasonably know the seed, e.g. by asking the authority that ran `keyGen`). For this reason, if the signer were interested in compressing the secret key as much as possible, they could just discard the key altogether and keep the seed value, rerunning `keyGen` to generate the secret key again whenever necessary.

There are a couple of reasons to consider our approach to be a valid alternative, however:

- At least in this context, storing the seed as opposed to the entire key is not standard — indeed, the MVQC-based submission to NIST ([12]) only suggests storing the seed instead of the secret key for small devices, where memory might be more constrained. This is due to the fact that `keyGen` is rather slow[5] when compared to the signature/verification processes.
- Our representation of the secret key might not only be useful to increase space efficiency, but time efficiency aswell — though we have not explored the topic in this text, it is not difficult to imagine how one might speed up basic operations from linear algebra knowing that a matrix is in RREF.

However, both of these points become moot when considering the next shortcoming:

Most subspaces are the rowspan of a $(I \mid A)$ matrix

We recall the observations from §4.1¶5: A k-dimensional subspace of $\mathbb{F}_q^n$ can be represented as the rowspan of a $k \times n$ matrix M. The rowspan of such a matrix is invariant with respect to left-multiplication by nonsingular matrices. As long as the first $k \times k$ minor of M is nonsingular, which happens with probability $\approx 1 - q^{-1}$, we have that M can be left-multiplied by the inverse of this minor to obtain $M' = (I_k \mid A)$. Therefore, in the context of MVQC, we could store A as the secret key, a $k \times (n-k)$ matrix, and understand that this refers to the row-span of $(I_k \mid A)$. Chapter 4 has been largely dedicated to patching up this idea so that it can also represent the remaining $1/q$ portion of subspaces which cannot be represented in this way — however, it is also a possibility to simply remove these subspaces from consideration, i.e. have `keyGen` always output subspaces that can be represented as the rowspan of a matrix in the form $(I_k \mid A)$.

This is suggested in [7, §3.1¶5], with good reason — it provides several improvements at the cost of extremely marginal security loss. We go through these:

Security loss. Excluding $1/q$ of the possible secret keys from consideration incurs a loss of $\log_2(1 - q^{-1}) \approx \ln(2)/q$ bits[6] of security *against direct key-guessing attacks*. Practical attacks would likely not see any increase in effectiveness at all, and in any case this increase

[5] Slow in practice. In theory, the asymptotic complexity of `keyGen` is smaller than that of `sign` or `verify`, but `keyGen` requires running a (usually not parallelizable) PRG $O(n^2m)$ times, while `sign` and `verify` primarily involve the execution of Basic Linear Algebra Subprograms (BLAS), which are exceedingly fast on modern computers.

[6] The parameter sets for the MVQC NIST submission take $q = 16$ and $q = 256$, meaning a loss of $< 5\%$ and $< 0.3\%$ of a bit, respectively

would be bounded above by $\approx \ln(2)/q$ bits.

Reduced key sizes. Secret keys can be represented with $k(n-k)$ elements of $\mathbb{F}_q$.

Efficient computation. Essentially the same point as above regarding matries in RREF. Computation would be strictly faster with matrices in general RREF (as they may have less than $k(n-k)$ nonzero elements), but it is out of scope to extend BLAS routines to handle RREF matrices, whereas computation with matrices of the form $(I \mid A)$ can be efficiently represented as traditional linear algebra operations.

Reduced attack surface. Assuming that secret keys are of the form $(I \mid A)$ makes the scheme significantly simpler in practice, leading to much less probability of potentially exploitable human error in implementation.

Chapter 5

Conclusion

This thesis has dealt in decent depth with two separate aspects of UOV.

In the realm of the security of UOV simplifications — we have shown that the traditional simplification of assuming that the masking term is linear cannot compromise security in any cryptographical sense. The question of whether it is safe to assume that the central map is homogenous, however, turns out to be a lot more nuanced. We have seen that this is most likely safe in practice, as key-recovery attacks (which essentially amounts to all known attacks) transfer losslessly from UOV to homogenousUOV and back. A formal result about the EUF security of this simplification likewise straightforward.

The case of EUF-CMA, however, turns out to be significantly more convoluted, as it is difficult to use a UOV challenger to simulate the signature distribution of a homogenousUOV challenger — without somehow leaking some information that should be hidden from the UOV adversary for the reduction to work. We obtain a reduction supported on a (somewhat natural, non ad-hoc) hardness assumption that essentially serves to bound how much the adversary should be able to learn from the differences between the distribution of legitimate signatures and sufficiently clever fake signatures. The following appendix deals with these hardness assumptions, providing some heuristic arguments that suggest that the relevant problems may well be cryptographically hard.

Though we have not covered it in this text, there is a clear avenue for future work on investigating how these results apply to the sEUF (-CMA, -KMA) setting — as well as seeing if they extend to MVQC schemes derived from UOV, such as Rainbow or MAYO. Moreover, as MVQC is usually understood as a candidate for post-quantum cryptography, another very useful result would be to figure out if the reductions we give in the ROM extend to the QROM, where the black box is allowed to make quantum queries to the random oracle.

On the other hand, we have found optimal representations for UOV secret keys — which is tantamount to finding optimal representations of subspaces. This has turned out to be much too complicated for the very slim benefit it begets when applied to UOV — nonetheless, much like in the previous point, there may well be a possibility that extending these results to Rainbow or MAYO may yield ideas which are more practically applicable. Moreover, it seems likely that optimal encodings of subspaces may be useful elsewhere.

Bibliography

[1] Tom M. Apostol, *Introduction to analytic number theory*, Springer New York, New York, NY, 1976.

[2] Sanjeev Arora and Boaz Barak, *Computational complexity: A modern approach*, 1st ed., Cambridge University Press, USA, 2009.

[3] Per Austrin, Johan Håstad, and Venkatesan Guruswami, $(2+\epsilon) - \mathsf{SAT}$ *is* NP*-hard*, 2014 IEEE 55th Annual Symposium on Foundations of Computer Science, 2014, pp. 1–10.

[4] Mihir Bellare and Gregory Neven, *Multi-signatures in the plain public-key model and a general forking lemma*, Proceedings of the 13th ACM Conference on Computer and Communications Security (New York, NY, USA), CCS '06, Association for Computing Machinery, 2006, p. 390–399.

[5] Daniel J. Bernstein and Tanja Lange, *Post-quantum cryptography - dealing with the fallout of physics success*, IACR Cryptol. ePrint Arch. **2017** (2017), 314.

[6] Ward Beullens, *Improved cryptanalysis of UOV and rainbow*, Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part I (Anne Canteaut and François-Xavier Standaert, eds.), Lecture Notes in Computer Science, vol. 12696, Springer, 2021, pp. 348–373.

[7] Ward Beullens, *MAYO: practical post-quantum signatures from oil-and-vinegar maps*, Selected Areas in Cryptography - 28th International Conference,SAC 2021, Virtual Event, September 29 - October 1, 2021, Revised Selected Papers (Riham AlTawy and Andreas Hülsing, eds.), Lecture Notes in Computer Science, vol. 13203, Springer, 2021, pp. 355–376.

[8] Ward Beullens, *Breaking rainbow takes a weekend on a laptop*, IACR Cryptol. ePrint Arch. (2022), 214.

[9] Dan Boneh and Victor Shoup, *A graduate course in applied cryptography*, 2015.

[10] An Braeken, Christopher Wolf, and Bart Preneel, *A study of the security of unbalanced oil and vinegar signature schemes*, Topics in Cryptology – CT-RSA 2005 (Berlin, Heidelberg) (Alfred Menezes, ed.), Springer Berlin Heidelberg, 2005, pp. 29–43.

[11] Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir, *Efficient algorithms for solving overdefined systems of multivariate polynomial equations*, Advances in Cryptology — EUROCRYPT 2000 (Berlin, Heidelberg) (Bart Preneel, ed.), Springer Berlin Heidelberg, 2000, pp. 392–407.

[12] Jintai Ding, Ming-Shing Chen, Albrecht Petzoldt, Dieter Schmidt, and Bo-Yin Yang, *Rainbow (supporting documentation)*, NIST Digital Signature Algorithms Round 2 **4** (2020).

[13] Jintai Ding and Dieter Schmidt, *Rainbow, a new multivariable polynomial signature scheme*, vol. 3531, 06 2005, pp. 164–175.

[14] Jean Faugère, *A new efficient algorithm for computing gröbner bases without reduction to zero (f 5)*, Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC (2002), 75–83.

[15] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan, *Trapdoors for hard lattices and new cryptographic constructions*, Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing (New York, NY, USA), STOC '08, Association for Computing Machinery, 2008, p. 197–206.

[16] Aviad Kipnis, Jacques Patarin, and Louis Goubin, *Unbalanced oil and vinegar signature schemes*, Advances in Cryptology — EUROCRYPT '99 (Berlin, Heidelberg) (Jacques Stern, ed.), Springer Berlin Heidelberg, 1999, pp. 206–222.

[17] Aviad Kipnis and Adi Shamir, *Cryptanalysis of the oil and vinegar signature scheme*, Advances in Cryptology — CRYPTO '98 (Berlin, Heidelberg) (Hugo Krawczyk, ed.), Springer Berlin Heidelberg, 1998, pp. 257–266.

[18] Donald E Knuth, *Subspaces, subsets, and partitions*, Journal of Combinatorial Theory, Series A **10** (1971), no. 2, 178–180.

[19] Leon Kraft, *A device for quantizing, grouping, and coding amplitude-modulated pulses*, Thesis (M.S.) Massachusetts Institute of Technology. Dept. of Electrical Engineering (1949).

[20] Tsutomu Matsumoto and Hideki Imai, *Public quadratic polynomial-tuples for efficient signature-verification and message-encryption*, Advances in Cryptology — EUROCRYPT '88 (Berlin, Heidelberg) (D. Barstow, W. Brauer, P. Brinch Hansen, D. Gries, D. Luckham, C. Moler, A. Pnueli, G. Seegmüller, J. Stoer, N. Wirth, and Christoph G. Günther, eds.), Springer Berlin Heidelberg, 1988, pp. 419–453.

[21] B. McMillan, *Two inequalities implied by unique decipherability*, IRE Transactions on Information Theory **2** (1956), no. 4, 115–116.

[22] Jacques Patarin, *Hidden fields equations (hfe) and isomorphisms of polynomials (ip): Two new families of asymmetric algorithms*, Advances in Cryptology — EUROCRYPT '96 (Berlin, Heidelberg) (Ueli Maurer, ed.), Springer Berlin Heidelberg, 1996, pp. 33–48.

[23] Jacques Patarin, *The oil and vinegar signature scheme*, Dagstuhl Workshop on Cryptography 9/1997 (1997).

[24] Dana Randall, *Efficient generation of random nonsingular matrices*, Random Structures Algorithms **4** (1993).

[25] C. E. Shannon, *Communication theory of secrecy systems*, The Bell System Technical Journal **28** (1949), no. 4, 656–715.

[26] William Waterhouse, *How often do determinants over finite fields vanish?*, Discrete Mathematics **65** (1987), 103–104.

Appendix A

Hardness of quadratic systems

Theorem A.1. *Let $\mathbb{K}$ be a nontrivial field. The problem MQD of deciding whether a given system of quadratic equations over $\mathbb{K}$ has a solution is NP-complete.*

This is rather well known, but no complete proof is available online without journal access, so a sketch of a reduction of 3SAT to MQD is provided. This implies that MQD is NP-hard — moreover, MQD is obviously in NP, so it is NP-complete.

Proof. Consider some boolean formula over variables in 3CNF form $C = \phi_1 \wedge \cdots \wedge \phi_m$, where each ϕ_i is a disjunctive clause of three literals, and call the variables $x_1, \ldots, x_n$. We aim to reduce this to an instance of MQD.

First we introduce a number of variables that model the literals in the formula — define n variables $\tilde{x}_1, \ldots, \tilde{x}_n$, all living in $\mathbb{K}$. Add to the system of equations the constraint $\tilde{x}_i^2 - \tilde{x}_i = 0$, for each $i \in [n]$. Notice that, independent of $\mathbb{K}$, this implies that each $\tilde{x}_i$ only takes values in $\{0, 1\}$. Intuitively, this will represent the truthiness of x_i. Moreover, add n new variables $\neg\tilde{x}_i$, and n new equations $\neg\tilde{x}_i + \tilde{x}_i = 1$. As before, $\neg x_i$ represents the truthiness of $\neg\alpha_i$. At this point, for any literal α (irrespective of polarity), we can unambiguously refer to a variable $\tilde{\alpha}$, where we expect that $\alpha \iff \tilde{\alpha} = 1$

Now we intend to model conjunction — for this, notice that $\alpha_i \wedge \alpha_j \iff \tilde{\alpha}_i \cdot \tilde{\alpha}_j = 1$. Possible justifications aside, it is easy to just draw up the truth tables of both of these statements and see that they match (and that the RHS evaluates only to 0 or 1). Similarly, we can model disjunction: $\alpha_i \vee \alpha_j \iff \neg(\neg\alpha_i \wedge \neg\alpha_j) \iff 1 - (1 - \tilde{\alpha}_i) \cdot (1 - \tilde{\alpha}_j) = 1$.

The only thing that remains is to expand this to disjunctions and conjunctions of several terms, and we can proceed to model the entire formula. Consider, for some $i \in [m]$, $\phi_i = \alpha_1 \vee \alpha_2 \vee \alpha_3$. Add two new variables $P_i, \tilde{\phi}_i$, and two new equations, $P_i = 1 - (1 - \tilde{\alpha}_1) \cdot (1 - \tilde{\alpha}_2)$ and $\tilde{\phi}_i = 1 - (1 - P_i) \cdot (1 - \tilde{\alpha}_3)$. As per the prior paragraph, $\tilde{\phi}_i$ takes values in $\{0, 1\}$ and $\tilde{\phi}_i = 1 \iff (P_i = 1) \vee \alpha_3 \iff (\alpha_1 \vee \alpha_2) \vee \alpha_3 \iff \phi_i$.

Finally, define variables S_i for $i \in [0, m]$, with the constraints that $S_0 = 1$, and for each $i \in [m]$, $S_i = S_{i-1} \cdot \tilde{\phi}_i$. A straightforward induction yields that $S_i \in \{0, 1\}$, and that $S_i = 1 \iff \phi_1 \wedge \cdots \wedge \phi_i$, and thus $C \iff S_m = 1$. Finally, we add the equation $S_m = 1$, and obtain that our polynomial system has a solution iff C has a satisfying assignment. A cursory check reveals that we used an amount of variables and equations linear in $max(n, m)$, and that all the equations of degree ≤ 2, so this is a valid reduction. □

We make a couple remarks.

On generality. Note that, aside having NP-hardness independent of the base field chosen, the fact that this result holds for quadratic systems means that it trivially also holds for generic polynomial systems (as well of systems of any constrained degree ≥ 2 — the linear case is well-known to be in P, as per any linear algebra course).

On underdetermined systems. Importantly, here MQD is phrased as a decision problem (hence the D). However, one is often rather interested in the task of *solving* systems of equations, rather than deciding whether a system has a solution at all. This can be interpreted in (at least) two different ways:

- Given a polynomial system, return a solution, or $\perp$ if no solution exists.
- Given a polynomial system and the knowledge that it is solvable, return a solution.

There is a trivial reduction from MQD to the former of these two interpretations, so this interpretation must also be computationally hard. However, the same is not immediately clear for the latter interpretation — and this is not a farfetched scenario, since this very text is often interested in finding solutions to underdetermined quadratic systems. In this context, one can imagine an algorithm that acts as if though it has been promised that the equation system fed as an input is solvable, and if the algorithm is correct when this assumption is true, then it will be correct *in general* with high probability.

However, it turns out that finding a satisfying assignment of a CNF formula is hard, even with the knowledge that a satisfying assignment exists (this is a specific case of the main result in [3]). Thus, the same reduction as above shows that it is computationally hard to solve a quadratic system even if one is promised that the system is solvable.

Appendix B

Hardness of MH and related problems

In §3.2.4, we show that homogenousUOV is, in some sense, as secure as UOV. In particular, for showing that homogenousUOV retains EUF-CMA security, we need to introduce some new problems and base the reductions on assuming the hardness of these problems. In this appendix, we support the claim that these problems are computationally hard, by dicussing the relationships between these problems and showing that some variants of them are indeed hard.

The first problem that we introduce is the missing hyperplane problem — $\mathsf{MH}_q(n)$ consists of, given a sampleable distribution $\mathcal{X}_i$, distinguishing which of these distributions it is:

- $\mathcal{X}_1$, uniform on $\mathbb{F}_q^n$
- $\mathcal{X}_2$, uniform on $\mathbb{F}_q^n \setminus V$, where V is a random hyperplane unknown to the adversary.

We (arbitrarily) focus on one type of strategy: sampling a set P of an amount $N(q,n)$ of points from $\mathcal{X}_i$, and then checking if there exists a hyperplane H such that $P \cap H = \varnothing$. This will always be the case if $i = 2$ — however, we can see that the probability that such ah H exists when $i = 1$ can be made arbitrarily low. Indeed, consider the expected number of hyperplanes H not intersecting $N(q,n)$ uniform samples from $\mathbb{F}_q^n$: since these are uniform, the probability of a certain H not intersecting one of them is $1 - 1/q$, and thus the probability of not intersecting any of them is $(1-1/q)^{N(q,n)}$, and thus the expected number of such hyperplanes is

$$\binom{n}{n-1}_q \cdot (1-1/q)^{N(q,n)}.$$

Moreover, by Markov's inequality, this expression above bounds the probability that any such hyperplane exists. If we allow some crude estimations, we have that

$$\begin{aligned}
\binom{n}{n-1}_q \cdot (1-1/q)^{N(q,n)} &= \frac{q^n-1}{q-1} \cdot (1-1/q)^{N(q,n)} \\
&= \exp\left\{ \log\left(\frac{q^n-1}{q-1}\right) + N(q,n)\log(1-1/q) \right\} \\
&\approx \exp\left\{ (n-1)\log(q) - N(q,n)/q \right\}
\end{aligned}$$

Now, if we want to have this be bounded above by some p (such that $1 - p$ is non-negligible), isolating $N(q, n)$ yields that this is equivalent to

$$N(q, n) \geq q \cdot \Big((n-1)\log(q) - \log(p) \Big).$$

All in all, we have that $N(q, n) \in O(n \cdot q \cdot \log(q))$ is enough for the strategy we described to be able to distinguish $\mathcal{X}_1$ and $\mathcal{X}_2$ with non-negligible advantage. However, we skipped over one important point — the process of finding out whether there exists an hyperplane H not intersecting any point of our sample of $N(q, n)$ points. This is a nontrivial computational problem:

Definition B.1. *The **non-intersecting hyperplane problem** in order q, abbreviated NIH_q, consists of, given n, N, and a set P of N points in $\mathbb{F}_q^n$, deciding whether there exists an hyperplane $H \subseteq \mathbb{F}_q^n$ such that $H \cap P = \varnothing$.*

It is not clear that this problem should be easy or hard — so our previous argument only amount to showing that $\mathsf{MH}_q(n)$ is easy if NIH_q is easy. We will give two results about NIH. Before this, however, it is convenient to slightly reformulate NIH: any hyperplane H is uniquely determined by a vector v, such that $H = \{x \mid \langle x, v\rangle = 0\}$. Evidently, $p \notin H \iff \langle p, v\rangle \neq 0$. Thus, an instance of NIH_q can be reformulated as, given a set of vectors $P \subseteq \mathbb{F}_q^n$, deciding whether there exists a vector v such that, for any $p \in P$, it holds that $\langle v, p\rangle \neq 0$.

Proposition B.2. *NIH_2 is in P.*

Proof. Given a set of vectors P, we have to decide whether there exists a vector v such that, for every $p \in P$, $\langle v, p\rangle \neq 0$. Because the underlying field is $\mathbb{F}_2$, we have that $\langle v, p\rangle \neq 0 \iff \langle v, p\rangle = 1$. This is a linear system (and moreover, the amount of variables and constraints are sublinear in the size of the problem instance). □

Corollary B.3. *$\mathsf{MH}_2(n)$ can be efficiently solved with non-negligible advantage.*

While this is not great news for the hardness of our problems, it turns out that NIH is NP-complete in every other case. This result will be based on the well-known NP-completeness of 3-COL, the problem of deciding whether a given graph has a 3-coloring — however, it will be convenient to go through the intermediate step of verifying that k-COL is NP-complete for any $k > 2$.

Lemma B.4. *Let $k > 2$. Then, k-COL, the problem of deciding whether a given graph has a k-coloring, is NP-complete.*

Proof. By reduction to 3-COL — trivially if $k = 3$, so we assume that $k > 3$. Let G be a graph for which we want to figure out if there exists a 3-coloring. Consider a new graph G', constructed by adding $k - 3$ new vertices to G and connecting them to every other vertex (including each of the other $k - 4$ new vertices). If we have a solver for k-COL, then we can figure out whether G' has a k-coloring — to complete the reduction, we only need to check that G' has a k-coloring iff G has a 3-coloring.

But this is easy. ($\Longleftarrow$) If G has a 3-coloring, then G' can be k-colored by starting with the 3-coloring of G and using each of the remaining $k-3$ new colors on the $k-3$ new vertices. ($\Longrightarrow$) On the other hand, if G' has a k-coloring, since the $k-3$ new vertices are all connected among themselves, they must use $k-3$ different colors. Moreover, since they are connected to each of the original vertices in G, none of these vertices can use these $k-3$ colors, leaving only 3 colors — so we can recover a 3-coloring of G. □

Proposition B.5. *Let $q > 2$. Then, NIH_q is NP-complete.*

Proof. By reduction to q-COL — in fact, NIH_q essentially amounts to a very broad generalization of q-COL. So consider a graph $G = (V, E)$ for which we want to decide if there exists a q-coloring. We fix $n = \#V$ and $N = \#E$, and we intend to select N vectors $\{p_i\}_{i\in[N]}$ from $\mathbb{F}_q^n$, such that the existence of another $v \in \mathbb{F}_q^n$ with $\langle p_i, v\rangle \neq 0$ for each i determines the existence of a q-coloring for G.

The idea is rather straightforward. Each coordinate of $v = (v_1, \ldots, v_n)$ will correspond to a vertex in V (for convenience, we relabel such that $V = [n]$). The value of this coordinate will be the color of the associated vertex. Let b_i denote the i-th element of the canonical basis of $\mathbb{F}_q^n$ — then, since $\langle v, b_i - b_j\rangle = v_i - v_j$, we have that $\langle v, b_i - b_j\rangle \neq 0 \iff v_i \neq v_j$. So, letting α_i and ω_i be the vertices incident to each edge e_i, we set $p_i = b_{\alpha_i} - b_{\omega_i}$. Therefore, imposing that $\langle p_i, v\rangle \neq 0$ for each i is equivalent to imposing that $v_{\alpha_i} \neq v_{\omega_i}$ for each edge e_i — clearly a k-coloring of G can be recovered from such a v and viceversa, so this v exists iff G is k-colorable. □

Corollary B.6. *If NP≠BPP, then $\mathsf{NIH}_q \notin$ BPP for $q > 2$.*

The antecedent in the corollary above is generally believed, with rather high confidence, to be true, so we can conclude with reasonable confidence that NIH_q is not in BPP. Nonetheless, we remark that this is two steps removed from being useful: the corollary above only *suggests* that NIH_q is cryptographically hard, which in turn only *suggests* that $\mathsf{MH}_q(n)$ might also be cryptographically hard (as we only focused on one possible approach to solving $\mathsf{MH}_q(n)$ with the hopes that it would be reasonably generic).

Another problem that comes up in §3.2.4 is the hidden hyperplanes problem, $\mathsf{HHs}_q(n)$, which asks us to, given a sampleable distribution $\mathcal{Y}_i$, distinguish which of these distributions it corresponds to:

- $\mathcal{Y}_1$, uniform on $\mathbb{F}_q^n$
- $\mathcal{Y}_2$, uniform on $V \cup -V$, where V is a random affine hyperplane unknown to the adversary.

Recall the reformulation that we gave earlier of NIH_q. We could have given a similar reformulation of $\mathsf{MH}_q(n)$ — distinguishing a uniform distribution on $\mathbb{F}_q^n$ from a uniform distribution on $\{x \in \mathbb{F}_q^n \mid \langle x, v\rangle \neq 0\}$ for some unknown v. An analogous idea applies to $\mathsf{HHs}_q(n)$, as, if v is a (the) vector such that $V = \{x \mid \langle x, v\rangle = 1\}$, then we have that $\mathcal{X}_2$ is in fact uniform on $\{x \in \mathbb{F}_q^n \mid \langle x, v\rangle = \pm 1\}$. now, since $X = \pm 1 \iff X \neq 0$ when $q \in \{2, 3\}$, we immediately obtain that $\mathsf{MH}_q(n)$ and $\mathsf{HHs}_q(n)$ are equivalent when $q \in \{2, 3\}$, and

therefore $\mathsf{HHs}_2(n)$ is easy and we suspect $\mathsf{HHs}_3(n)$ to be hard.

As for $q > 5$, we show that, unsurprisingly, $\mathsf{MH}_q(n)$ must always be harder than $\mathsf{HHs}_q(n)$:

Proposition B.7. *For any efficient algorithm* `A` *that solves* $\mathsf{MH}_q(n)$*, there exists an efficient algorithm* `B` *that solves* $\mathsf{HHs}_q(n)$ *with the same advantage and linear overhead in execution time.*

Proof. The idea is that there exists an efficiently computable function $f\colon \mathbb{F}_q^n \to \mathbb{F}_q^n$ such that $f(\mathcal{Y}_i) \sim \mathcal{X}_i$ — so `B` will simply recieve samples, take their image under f, and feed these images to `A`, outputting whatever `A` eventually outputs.

The function f is rather simple — on input y, it simply selects a random nonzero $r \in \mathbb{F}_q^*$, and returns $r \cdot y$. Since $y \mapsto r \cdot y$ is a bijection for each r, we have that it preserves uniform distributions, and since f simply applies this transform for a random r, we obtain that $f(\mathcal{Y}_1) \sim \mathcal{X}_1$. We also have to verify that $f(\mathcal{Y}_2) \sim \mathcal{X}_2$ — so say that y is sampled from $\mathcal{Y}_2$, and therefore it is an element from $V \cup -V$. As we saw before, there is a certain vector v such that this is tantamount to saying that $\langle v, y\rangle = \pm 1$. Now, if r is randomly selected from $\mathbb{F}_q^*$, we have that $\langle v, r \cdot y\rangle = r\langle v, y\rangle = \pm r$ will be uniformly distributed over $\mathbb{F}_q^*$. Moreover, again since multiplication by nonzero r is bijective, and since y is uniformly distributed over $\{z \mid \langle v, z\rangle = \pm 1\}$, then $r \cdot y$ is uniformly distributed over $\{z \mid \langle v, z\rangle = \pm r\}$. We obtain that the distribution of $f(y)$ is the one obtained by uniformly selecting a random pair of (mirrored) hyperplanes $\{z \mid \langle v, z\rangle = \pm r\}$, and then uniformly selecting one of their elements — since all these hyperplanes have the same cardinality and they span all of $\mathbb{F}_q^n$ except for $\{z \mid \langle v, z\rangle = 0\}$, it follows that the distribution of $f(y)$ is uniform on $\mathbb{F}_q^n \setminus V$, that is, $f(\mathcal{Y}_2) \sim \mathcal{X}_2$. □

Along the same lines as before, we can also (non-canonically) turn HHs into a decision problem to make a study of its hardness more feasible. Following a reasoning analogous to the case of MH and NIH, we obtain:

Definition B.8. *The* ***spanning hyperplanes problem*** *in order q, abbreviated SHs_q, consists of, given n, N, and a set P of N points in $\mathbb{F}_q^n$, deciding whether there exists an affine hyperplane $V \subseteq \mathbb{F}_q^n$ such that $P \subseteq V \cup -V$.*

For the same reason that $\mathsf{MH}_q(n)$ and $\mathsf{HHs}_q(n)$ are equivalent when $q \in \{2, 3\}$, we have that NIH_q and SHs_q are equivalent when $q \in \{2, 3\}$ meaning that SHs_2 is in P, while SHs_3 is NP-complete and likely cryptographically hard. As for $q > 5$, clearly NIH_q is harder than SHs_q — but this tells us nothing, since we already new NIH_q to be NP-complete. However, it turns out that SHs_q is also NP-complete. The same observations about the relationship between MH and NIH apply to that of HHs and SHs.

Proposition B.9. *Let $q > 2$. Then, SHs_q is NP-complete.*

Proof. We already know this to be true for $q = 3$, so we assume that $q > 5$. We will show that, for any q, there is a reduction from 3SAT to SHs_q.

So, let us consider an instance F of 3SAT, with X variables $x_1, \ldots, x_V$ and C clauses $\varphi_1, \ldots, \varphi_C$. We begin by fixing $n = 1 + 2X + C$ and $N = 1 + 3X + 2C$, and we intend to select N vectors $\{p_i\}_{i \in [0,3X+2C]}$ from $\mathbb{F}_q^n$, such that the existence of another $v \in \mathbb{F}_q^n$ with $\langle p_i, v \rangle = \pm 1$ for each i determines the existence of a satisfying assignment for F.

It will be convenient to, for the length of this proof, 0-index our vectors, such that $v = (v_0, v_1, \ldots, v_{2X+C})$. Notice that we also 0-indexed the set of points $\{p_i\}_{i \in [0,3X+C]}$ earlier. Let b_i denote the i-th element of the canonical basis of $\mathbb{F}_q^n$, also 0-indexed.

We begin by setting $p_0 = b_0$ — therefore, v will have to satify $v_0 = \langle v, p_0 \rangle = \pm 1$. At this point, it is important to note that if v satisfies that $\langle p_i, v \rangle = \pm 1$ for each i, then clearly $-v$ also satisfies this, and thus we may assume WLOG that $v_0 = 1$.

So we begin to model the variables. For each $i \in [1, \ldots, X]$, we set $p_i = b_i$, $p_{i+X} = b_{i+X}$, and $p_{i+2X} = 2^{-1} b_i - 2^{-1} b_{i+X}$. This determines p_j for $j = 1, \ldots, 3X$. For any fixed i, the values of p_i and p_{i+X} imply that $v_i = \pm 1$ and $v_{i+X} = \pm 1$, respectively, and moreover, $\pm 1 = \langle p_{i+2X}, v \rangle = 2^{-1}(v_i - v_{i+X})$ — since this cannot hold if v_i and v_{i+X} have the same sign, we have that $v_i = -v_{i+X}$. Conceptually, the sign of v_i represents the truthiness of x_i, and v_{i+X} that of $\neg x_i$. Thus, for any literal α in F, there is an unique index $i(\alpha)$ such that $v_{i(\alpha)}$ represents the truthiness of α.

We move on to the more interesting subject of modeling clauses. Each clause $\varphi_i = \alpha_1 \vee \alpha_2 \vee \alpha_3$, with $i \in [1, \ldots, C]$, will be modeled by v_{3X+i}, with $p_{3X+i} = b_{i(\alpha_1)} + b_{i(\alpha_2)} + b_{i(\alpha_3)} - b_{3X+i}$ and $p_{3X+C+i} = b_{3X+i} - b_0$. Note that $\pm 1 = \langle p_{3X+C+i}, v \rangle = v_{3X+i} - v_0 = v_{3X+i} - 1$, and so $v_{3X+i} \in \{0, 2\}$. Moreover, $\pm 1 = \langle p_{3X+i}, v \rangle = (v_{i(\alpha_1)} + v_{i(\alpha_2)} + v_{i(\alpha_3)}) - b_{3X+i}$, and therefore, $(v_{i(\alpha_1)} + v_{i(\alpha_2)} + v_{i(\alpha_3)}) \in \{-1, 1, 3\}$. Since $v_{i(\alpha_1)}$, $v_{i(\alpha_2)}$ and $v_{i(\alpha_3)}$) are all ± 1, the fact that their sum is in $\{-1, 1, 3\}$ is equivalent to at least one of them being 1 — i.e., at least one of $\alpha_1, \alpha_2, \alpha_3$ is truthy, and so we have succesfully modeled a disjunctive clause.

We obtain that an v such that $\langle p_i, v \rangle \neq 0$ for each i exists if and only if there is a satisfying assignment for F — moreover, the set of points p is polynomial in the size of F and can be constructed efficiently, thus completing the reduction. □

Yet another problem that comes up in §3.2.4 is the missing hyperplane sampling problem, $\mathsf{MHS}_q(n)$, which asks us, given a distribution over $\mathbb{F}_q^n \setminus V$, where V is some unkown hyperplane, to output a $p \in V$, $p \neq \vec{0}$. In the bounded-error setting, we have a somewhat underwhelming result (which is nonetheless sufficient for §3.2.4):

Proposition B.10. *If there exists an efficient algorithm $\mathtt{A}$ that can solve $\mathsf{MHS}_q(n)$ with probability $p(n)$, then there exists another efficient algorithm $\mathtt{B}$ which can solve $\mathsf{MH}_q(n)$ with probability $p(n) \cdot p(n-1) \ldots p(1)$.*

Proof. We describe B.

- B initially sets $V = \{0\}$ (as a subspace).
- For each $m = n, \ldots, 1$
  1. B repeatedly samples values x_i from the given distribution, computes $y_i \leftarrow \pi_V(x_i)$, writes $z_i \leftarrow \texttt{base}(V)^{-1} \cdot y_i$, and sends y_i to A.
  2. When A outputs some point p_j, B resets A, updates $V \leftarrow V + \langle \texttt{base}(V) p_j \rangle$, and continues to the next m.
- B checks that V does not contain any of the points it has sampled so far. If so, it accepts, otherwise it rejects.

Note that B would simply keep track of a basis of V, such that it is always the same basis. Essentially, for each value of m, B is playing the role of the challenger of $\mathsf{MHS}_q(m)$ against A, feeding it the distribution projected to the quotient of the parts of the hyperplane that A has already figured out. If B is sampling from $\mathcal{X}_2$, this is valid if A correctly finds a p in the (reparametrized) hidden hyperplane at each step, which happens with probability $p(m)$ at each step. On the other hand, if B is sampling from $\mathcal{X}_1$, with high probability the check at the end will fail, since no hyperplane like the one B wants will exists (unless not enough points have been sampled, but B can simply remedy this by sampling $\approx n \cdot q \cdot \log(q)$ points at the start). □

Corollary B.11. *If $\mathsf{MH}_q(n)$ is hard, there does not exist any algorithm that can solve $\mathsf{MHS}_q(n)$ with probability that is negligibly close to* 1.

Proof. Otherwise, the construction of Prop. B.10 would yield an algorithm capable of solving $\mathsf{MH}_q(n)$ with non-negligible probability. □

This result essentially just says that, if $\mathsf{MH}_q(n)$ is very hard, then $\mathsf{MHS}_q(n)$ isn't very easy — this is essentially true, since one needs to get $\mathsf{MHS}_q(n)$ right n times in a row to solve $\mathsf{MH}_q(n)$. However, it is worthwhile to note that a tighter reduction exists in the polynomial expected time setting:

Proposition B.12. *If there exists an efficient algorithm A that can solve $\mathsf{MHS}_q(n)$ with probability* 1 *in polynomial expected time, then there exists another efficient algorithm B which can solve $\mathsf{MH}_q(n)$ with probability negligibly close to* 1 *in polynomial expected time.*

Proof. The same construction as in Prop. B.10. Regarding B's execution time — in aggregate, the internal executions of A take an expected $poly(n) + poly(n+1) + \cdots + poly(1) \leq n \cdot poly(n) = poly(n)$, and B only has some extra polynomial overhead, so B finishes in expected polynomial time, as we wanted. Regarding correctness, if B is sampling from $\mathcal{X}_2$, then A correctly finds an element of the hidden hyperplane at each step, and therefore the final check by B passes and B accepts. If B is sampling from $\mathcal{X}_1$, it rejects with high probability, with the same considerations as in the proof of Prop. B.10. □

Appendix C

Proof of Lemma 3.15

Claim. *Let* $\mathtt{A}$ *be an efficient algorithm that can break the* EUF-CMA *security of* $\mathsf{homogenousUOV}_q(n+1,m)$ *with probability* p*, and* $\mathtt{B}_1$ *as constructed in Fig. 3.4, letting* p_a *be the probability that* $\mathtt{A}$ *outputs a valid forgery with respect to its simulated random oracle. Then, there exists an efficient algorithm* $\mathtt{C}$ *such that, letting* p_c *be its probability of solving* $\mathsf{MH}_q(n+1-m)$*, it holds (in the Random Oracle Model) that* $p_c \geq |p_a - p|$*.*

Since it is a key part of this statement, we repeat Fig. 3.4:

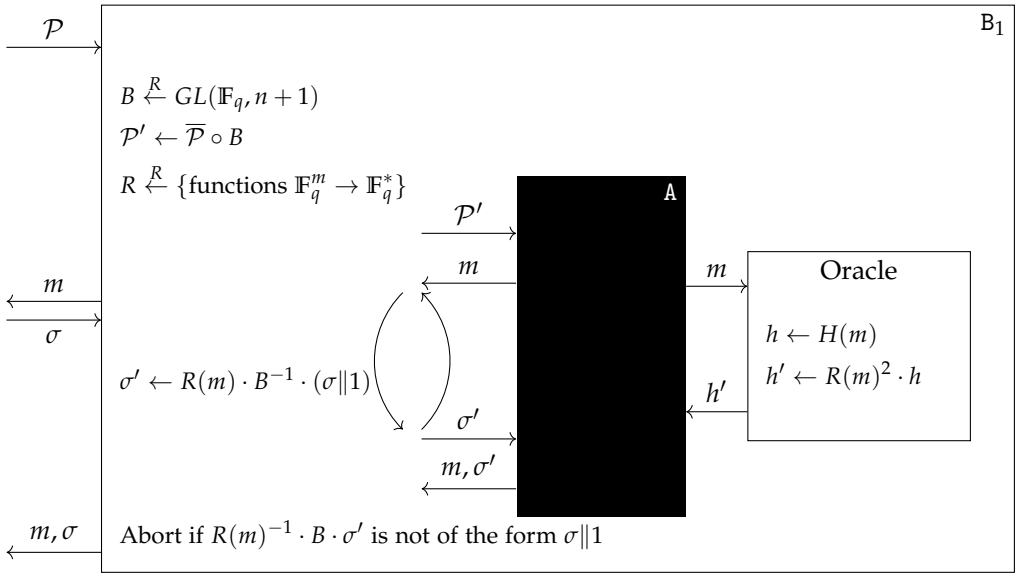


Figure C.1: Reduction from breaking EUF-CMA security of $\mathsf{UOV}_q(n,m)$ to EUF-CMA security of $\mathsf{homogenousUOV}_q(n+1,m)$.

Proof. We begin by pitting $\mathtt{A}$ against the standard challenger for the EUF-CMA security of $\mathsf{homogenousUOV}_q(n+1,m)$. As per the definition of $\mathtt{A}$, there is a probability of p that this game ends with the challenger accepting. This is illustrated in Fig. C.2, where we simply use `keyGen`, `sign` and `verify` to refer to the corresponding algorithms in $\mathsf{homogenousUOV}_q(n+1,m)$.

We will introduce some variations in how the signing process of the challenger works, while justifying that these do not significantly alter the outcome of the game — these are given in the table below. `base` denotes a function that, given a subspace, produces a matrix

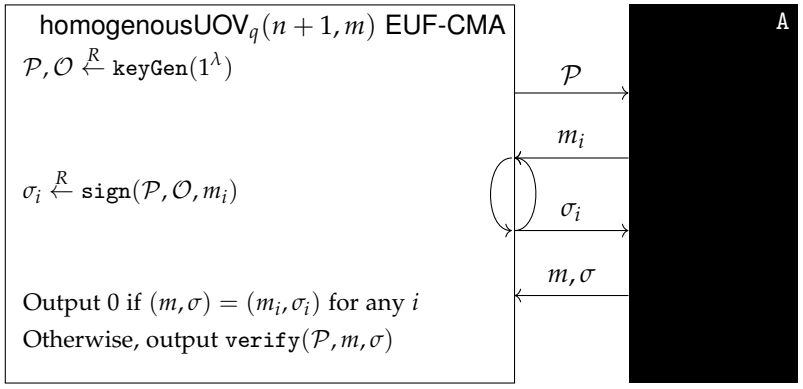


Figure C.2: Game for the EUF-CMA security of homogenousUOV$_q(n+1,m)$, played by A.

with this subspace as its column-span. $\pi_V(\mathbf{x})$ denotes the orthogonal projection of some vector $\mathbf{x}$ to a subspace V.

$\texttt{sign}'(\mathcal{P}, \mathcal{O}, m)$	$\texttt{sign}''(\mathcal{P}, \mathcal{O}, m)$	$\texttt{sign}'''(\mathcal{P}, \mathcal{O}, m)$
1. $\mathbf{t} \leftarrow H(m)$	1. $\mathbf{t} \leftarrow H(m)$	1. $\mathbf{t} \leftarrow H(m)$
2. $\mathbf{v} \xleftarrow{R} \mathbb{F}_q^{n+1}$	2. $\mathbf{x} \xleftarrow{R} \mathcal{O}^\perp$	2. $\mathbf{v} \xleftarrow{R} \mathbb{F}_q^{n+1-m}$
3. $\mathbf{x} \leftarrow \pi_{\mathcal{O}^\perp}(\mathbf{v})$	3. $\mathcal{Q}\colon \mathcal{O} \to \mathbb{F}_q^m$	3. $\mathbf{x} \leftarrow \texttt{base}(\mathcal{O}^\perp) \cdot \mathbf{v}$
4. $\mathcal{Q}\colon \mathcal{O} \to \mathbb{F}_q^m$	$\mathbf{y} \mapsto \mathcal{P}(\mathbf{x}+\mathbf{y})$	3. $\mathcal{Q}\colon \mathcal{O} \to \mathbb{F}_q^m$
$\mathbf{y} \mapsto \mathcal{P}(\mathbf{x}+\mathbf{y})$	4. If $\mathcal{Q}$ is not of full rank,	$\mathbf{y} \mapsto \mathcal{P}(\mathbf{x}+\mathbf{y})$
5. If $\mathcal{Q}$ is not of full rank,	return to step 2.	5. If $\mathcal{Q}$ is not of full rank,
return to step 2.	5. $\mathbf{z} \leftarrow \mathcal{Q}^{-1}(\mathbf{t})$	return to step 2.
6. $\mathbf{z} \leftarrow \mathcal{Q}^{-1}(\mathbf{t})$	6. Output $\mathbf{x}+\mathbf{z}$	6. $\mathbf{z} \leftarrow \mathcal{Q}^{-1}(\mathbf{t})$
7. Output $\mathbf{x}+\mathbf{z}$		7. Output $\mathbf{x}+\mathbf{z}$

It is rather easy to check that the three algorithms presented above are equivalent, as their only difference is in the way that they uniformly sample a vector $\mathbf{x}$ from $\mathcal{O}^\perp$. Moreover, $\texttt{sign}'$ is equivalent to the standard sign of homogenousUOV$_q(n+1,m)$ — note that the only difference between the two algorithms is that $\texttt{sign}'$ orthogonally projects the vector it samples to $\mathcal{O}^\perp$. This cannot change which coset of $\mathcal{O}$ the map $\mathbf{y} \mapsto \mathbf{x}+\mathbf{y}$ runs over, and thus it does not change whether $\mathcal{Q}$ is invertible — and if it is indeed invertible, it does not change the unique solution returned in step 7.

It follows, then, that we can substitute sign for $\texttt{sign}'''$ in the game in Fig. C.2, and this will not change the distribution of the outcome. Note, then, that $\texttt{sign}'''$ contains an instruction to sample uniformly from $\mathbb{F}_q^{n+1-m}$ — therefore, the game in Fig. C.2 is equivalent to the game in Fig. C.3 when $b = 0$.

We remark that C should also check that the final forgery (m, σ) does not coincide with any of A's previous signing queries. In any case, this is the C from the statement of Lemma 3.15 — by definition, the probability of the challenger accepting cannot change by any amount

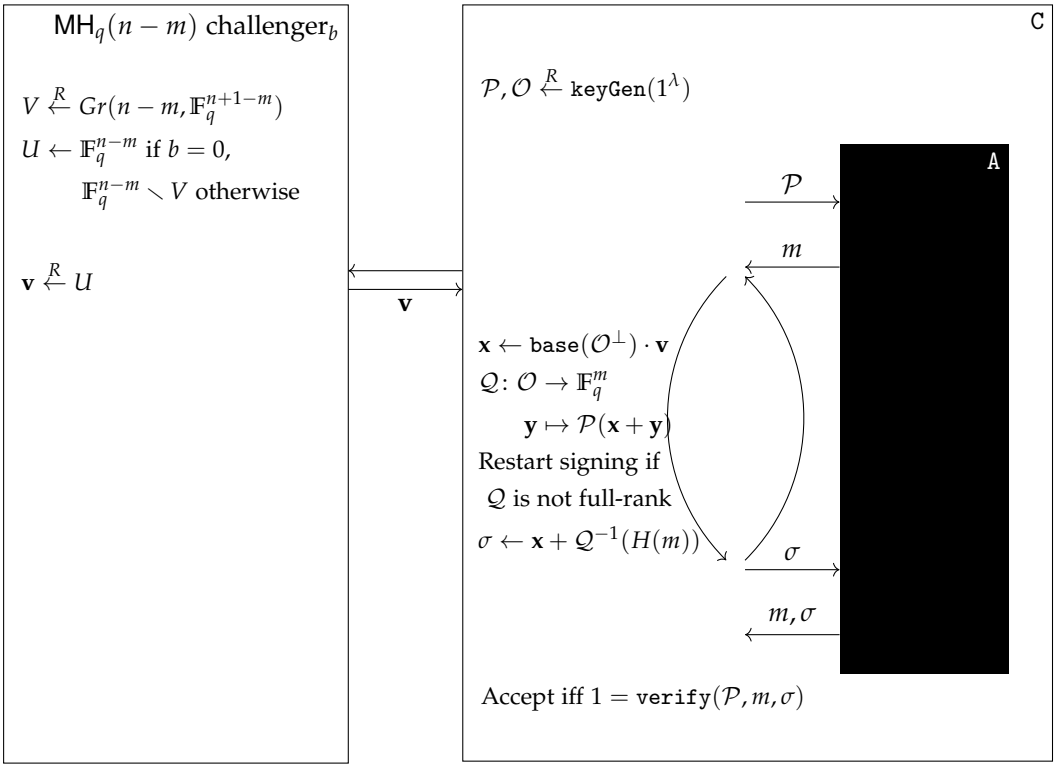


Figure C.3: Rewriting of the previous game as an instance of $\mathsf{MH}_q(n-m)$.

greater than p_c when we change b from 0 to 1. So we set $b = 1$, with the knowledge that the resulting game will accept with a probability at most p_c away from p.

What we intend to prove, then, is that the game above when $b = 1$ (which we will call Game$_1$) and the game in the reduction $\mathtt{B}_1$ (which we will call Game$_2$) look the same from the perspective of $\mathtt{A}$ — that is, the messages to $\mathtt{A}$ have the same distribution. If this is true, then it follows that the probability p_a that $\mathtt{A}$ outputs a valid forgery in Game$_1$ is the probability that $\mathtt{C}$ accepts in Game$_2$ — and so we have just seen that it will hold that $p_c \geq |p - p_a|$, as we intend to show.

So we need to show that the distribution of messages to $\mathtt{A}$ in Game$_1$ is identical to in Game$_2$. We begin by repeating a result we obtained in the proof of Prop. 3.11, regarding the distribution of the public and secret keys. Letting sk be the secret key of the challenger in Game$_2$, we saw that if we define $\mathcal{O}' = B^{-1} \cdot (sk \times \{0\})$, then $(\mathcal{P}', \mathcal{O}')$ follows the distribution of a genuine key pair for $\mathsf{homogenousUOV}_q(n+1, m)$ — and so it has the same distribution as $(\mathcal{P}, \mathcal{O})$ in Game$_1$.

We focus ourselves on Game$_2$ for a paragraph. Consider the hyperplane $B^{-1} \cdot (*\|0)$, and note that it must contain $\mathcal{O}' = B^{-1} \cdot (sk \times \{0\})$. Moreover, since $B^{-1} \cdot (*\|0)$ is the row-span

of the first $n-1$ columns of B^{-1}, it is uniformly distributed, and therefore $B^{-1} \cdot (*\|0)/\mathcal{O}'$ will be uniformly distributed over the quotient. Therefore, for any value of $\mathcal{O}'$, $B^{-1} \cdot (*\|0)$ is uniformly distributed over all hyperplanes containing $\mathcal{O}'$.

Going back to Game_1, we can see that analogous behaviour arises for $\mathcal{O} + \texttt{base}(\mathcal{O}^\perp) \cdot V$ — for any value of $\mathcal{O}$, the term $\texttt{base}(\mathcal{O}^\perp)$ only amounts to a choice of basis, and since V is uniformly distributed and of dimension $n - \dim(\mathcal{O}) - 1$, the expression $+\texttt{base}(\mathcal{O}^\perp) \cdot V$ extends $\mathcal{O}$ to any hyperplane containing it, uniformly and at random.

So we obtain that $(\mathcal{P}, \mathcal{O}, \mathcal{O} + \texttt{base}(\mathcal{O}^\perp) \cdot V)$ in Game_1 and $(\mathcal{P}', \mathcal{O}', B^{-1} \cdot (*\|0))$ in Game_2 have the same distribution. Next, we see that they are used in the same way to answer A's signing queries.

An important note about the distribution of the signatures is that in both games, the challenger to A is does not change state when answering A's queries, and thus the distribution of each signature will be identical and independent from the other signatures. Therefore, we simply have to show that for some signing query m from A, the distribution of the signature it recieves will be the same in Game_1 and Game_2.

Game_1 is easier to describe — upon recieving a signing query m, C repeatedly picks uniformly random cosets of $\mathcal{O}$ not contained in $\mathcal{O} + \texttt{base}(\mathcal{O}^\perp) \cdot V$ — indeed, note that $\mathbf{v}$ determines the choice of coset and it is uniformly distributed everywhere except $\texttt{base}(\mathcal{O}^\perp) \cdot V$. C repeats this process until it finds a coset where $\mathcal{P}$ is of full rank, which, as far as the resulting distribution goes, amounts to uniformly selecting one such coset. It then returns the only possible signature within this coset.

Game_2 is more convoluted, but it amounts to the same process. Upon recieving a signing query m, B_1 passes it back to its challenger, which will uniformly select a random coset of sk where pk is invertible, and send back the unique antiimage σ of $H(m)$ in this coset. Regarding the next few steps:

- B_1 will then consider $(\sigma\|1)$. The resulting distribution of $(\sigma\|1)$ would be the same as if A uniformly selected a coset of $sk \times \{1\}$ from $\mathbb{F}_q^n \times \{1\}$ where is invertible, and returned the unique signature there — A simply carries out the same operations, but in the homogeneization.

- Next, B_1 will consider $B^{-1} \cdot (\sigma\|1)$. Now the process of uniformly selecting a coset of $sk \times \{1\}$ from $\mathbb{F}_q^n \times \{1\}$ where is invertible is tantamount to uniformly selecting a coset of $B^{-1}(sk \times \{1\})$ from $B^{-1}(\mathbb{F}_q^n \times \{1\}) = B^{-1}(*\|1)$ where $\circ B^{-1} = \mathcal{P}'$ is invertible. Moreover, any coset of $B^{-1}(sk \times \{1\})$ is a coset of $\mathcal{O}'$, and so we can rephrase this as selecting a random coset of $\mathcal{O}'$ in $B^{-1}(*\|1)$ where $\mathcal{P}'$ is invertible.

- Finally, B_1 sets $\sigma' = R(m) \cdot B^{-1} \cdot (\sigma\|1)$. The process of selecting σ', therefore, is equivalent to first picking an $R(m)$, then a coset of $\mathcal{O}'$ within $B^{-1}(*\|R(m))$ where $\mathcal{P}'$ is invertible, and returning the unique signature σ' within this coset. Note that

the amount of such cosets in $B^{-1}(*\|R(m))$ does not depend on $R(m)$ (indeed, multiplication by any constant nonzero preserves the property of being a coset of $\mathcal{O}'$, and since $\mathcal{P}'$ is homogenous, it also preserves the property of having a unique solution within a certain coset) — therefore, this uniformly picking an $R(m)$ first and then a coset of $\mathcal{O}'$ within $B^{-1}(*\|R(m))$ is tantamount to simply picking any coset of $\mathcal{O}'$ lying in some $B^{-1}(*\|r)$, with $r \neq 0$ — so, picking a random coset of $\mathcal{O}'$ from $\mathbb{F}_q^{n+1} \setminus B^{-1} \cdot (*\|0)$ where $\mathcal{P}'$ is invertible.

So the signature process in Game_2 amounts to uniformly selecting a coset of $\mathcal{O}'$, not within $B^{-1}(*\|0)$, where $\mathcal{P}'$ is invertible, and then returning the unique antiimage of $H(m)$ within. We have seen that Game_1 does the same, substituting $(\mathcal{P}', \mathcal{O}', B^{-1} \cdot (*\|0))$ for $(\mathcal{P}, \mathcal{O}, \mathcal{O} + \texttt{base}(\mathcal{O}^\perp) \cdot V)$ — but these two triplets have the same distributions in either game, so the signatures returned to `A` also have the same distribution.

Finally, since we are in the random oracle model and $R(m)$ is independent from the "random" value $H(m)$, the simulated random oracle of `A` also behaves indistinguishably from a legitimate random oracle, thus completing the proof. □