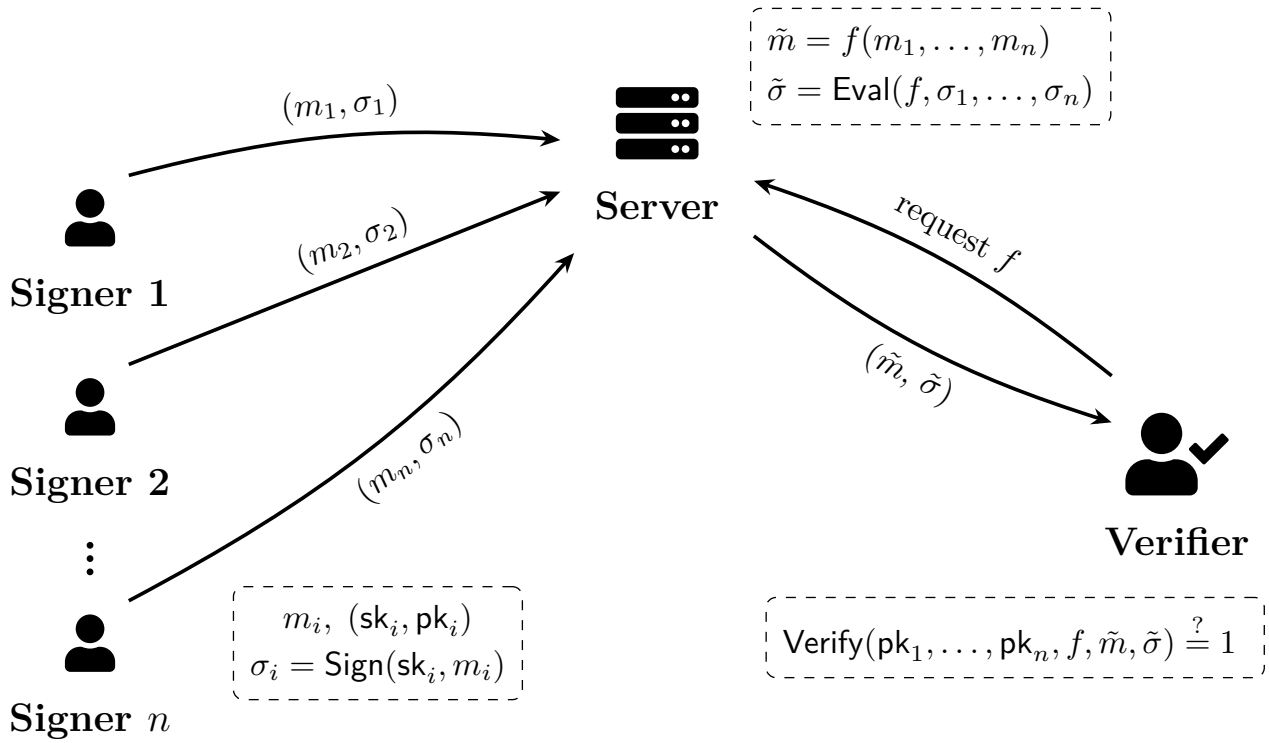
CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG



Beyond Linear

Pairing-Based Multi-Key Homomorphic Signatures for Verifiable Statistics

Master's thesis in Computer science and engineering

Anna Davoodi, Aram Jamal

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2026

Master's thesis 2026

Beyond Linear

Pairing-Based Multi-Key Homomorphic Signatures for Verifiable Statistics

Anna Davoodi, Aram Jamal

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2026

Beyond Linear:
Pairing-Based Multi-Key Homomorphic Signatures for Verifiable Statistics
Anna Davoodi, Aram Jamal

Supervisor: Hanna Ek, Department of Computer Science and Engineering
Examiner: Elena Pagnin, Department of Computer Science and Engineering

Master's Thesis 2026
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: The cover illustrates the three-party structure of multi-key homomorphic signatures. Independent signers produce individual signatures and send them to an untrusted server. A verifier queries the server to evaluate the signatures, and checks the result using public keys.

Typeset in LaTeX
Gothenburg, Sweden 2026

Beyond Linear:
Pairing-Based Multi-Key Homomorphic Signatures for Verifiable Statistics
Anna Davoodi, Aram Jamal
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Multi-key homomorphic signatures (MKHS) address the challenge of verifying outsourced computations. Multiple independent parties each sign their own data and upload it to an untrusted server, allowing anyone to outsource computations over the signed data by request. In response to a query, the server returns both the computed result and a compact signature certifying its correctness, which can be verified using only the public keys of the participating signers. Existing implemented MKHS only support evaluation of linear functions.

In this thesis, we address the problem of constructing implementable pairing-based MKHS schemes that support functions beyond linear, by extending the multi-key linearly homomorphic signature scheme of Aranha and Pagnin [Latincrypt, 2019]. We begin by reproving its security in the Type 3 pairing setting, bringing it in line with current cryptographic practices.

Building on this, we present mkqhs-br, a pairing-based MKHS supporting bounded-rank quadratic evaluation, proven secure under the co-CDH* assumption. We then give two independent extensions of mkqhs-br: (i) mkqhs-$\widetilde{\text{br}}$, which compresses evaluated signature sizes, and (ii) mkqhs-br-m^2, which extends the supported function class, enabling applications such as variance and least-squares computations. The extensions are compatible and can be combined, yielding a practical construction for statistical applications.

Keywords: Cryptography, Homomorphic Signature Schemes, Multi-Key, Pairing-Based Cryptography, Verifiable Statistics, Quadratic Functions.

Acknowledgements

We would first like to thank our supervisor Hanna Ek for her guidance, patience, and encouragement throughout this whole project. We are also grateful to our examiner Elena Pagnin, whose course first sparked our interest in cryptography, and whose research this thesis builds directly upon.

Finally, we also want to thank each other. Writing this took a long time, but various ideas kept the overall rhythm even. Between everything, keeping kindness and humour made the whole process something we'll look back on fondly.

Anna Davoodi and Aram Jamal, Gothenburg, 2026-06-09

Contents

List of Figures

List of Tables

1

Introduction

Modern computing tasks are often too large or expensive for a single party to carry out alone. Instead, the computation may be outsourced to a remote server that returns the result. This approach is practical and increasingly common, yet it has a clear limitation: there is no way to verify that the result is correct without redoing the computation yourself.

For instance, consider computing statistical analyses on user-provided data. Individuals or organisations each contribute a dataset, which a server processes to produce results end users rely on. The server might be entirely honest, but nothing in the returned results lets its users verify this. This lack of verifiability may be concerning when the results inform sensitive decisions, such as whether a patient receives a particular diagnosis, or whether a weather prediction triggers a flood warning.

Signature schemes offer a partial solution. These let data owners attach a short tag, a *signature*, to their data before sending it for processing. Anyone that receives the signed data can verify that it originated from the claimed signer and has not been tampered with. This provides authenticity for the original data. However, if someone were to perform computations on the signed data, the result would no longer carry a valid signature. A party who receives this result and wants assurance must still redo the computation.

Homomorphic signature schemes are designed to close this gap. When a server computes on signed data, it can simultaneously combine the individual signatures into an *evaluated signature* on the result. A verifier can then check this evaluated signature against the result to confirm it was correctly computed from the original signed inputs, without redoing the computation themselves. The data owner signs their data once, and verifiability of computations follows at no extra cost.

Multi-key homomorphic signature schemes (MKHS) generalise this to the setting where inputs are signed under different independent users. Each contributor signs their dataset with their own secret key, and a verifier receiving computed results can still check that it was correctly produced from those different sources. Figure 1.1 illustrates this setup.

Aranha and Pagnin introduced a concrete scheme of this kind, referred to as mklhs for *multi-key linearly homomorphic signatures* [1], and is built from tools of pairing-based cryptography that allow for efficient verification compared to other cryptographic primitives. The scheme supports linear computations over data signed by

multiple independent users, and covers verifiability of computations such as weighted sums and averages.

Linear functions, however, do not cover many common computations. Variance, for instance, involves squaring and multiplying data points, and the same is true for Euclidean distances, least-squares error, and polynomial regressions, none of which can be verified using mklhs. While homomorphic signature schemes supporting higher-degree computations exist in the single-key setting [2], no implemented MKHS scheme currently evaluates functions beyond linear.

In this thesis, we make two main contributions. First, we update the security proof of mklhs to a setting matching the current state of implemented cryptographic primitives, and give it a tighter security bound. Second, we extend mklhs to support a certain class of quadratic functions efficiently. This enables signature evaluation of functions such as variance, squared Euclidean distance, and least-squares loss, making pairing-based MKHS applicable to a wide range of statistical and machine-learning applications.

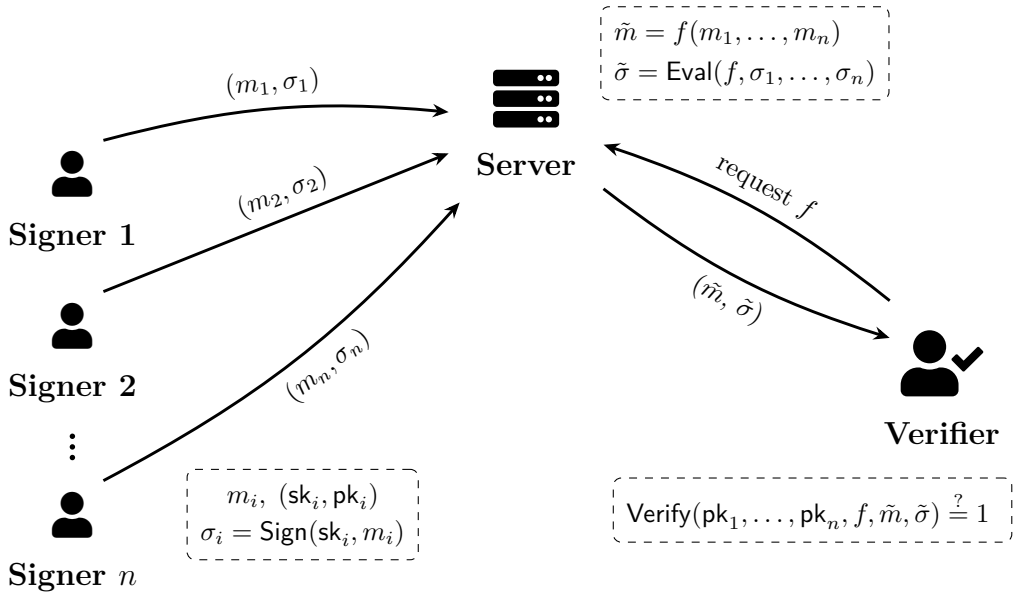


Figure 1.1: The multi-party structure of a multi-key homomorphic signature scheme. Each signer $i \in [n]$ holds an independently generated key pair $(\mathsf{sk}_i, \mathsf{pk}_i)$ and signs their own data point m_i under sk_i, producing a signature σ_i. The resulting pairs (m_i, σ_i) are sent to an untrusted server, which evaluates a function f over the data points and their signatures, producing an evaluated signature $\tilde{\sigma}$ on the computation $\tilde{m} = f(m_1, \ldots, m_n)$. A verifier, holding all public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_n$, then confirms that $\tilde{m}$ was correctly computed from the originally signed inputs, without redoing the computation or interacting with the signers.

1.1 Technical Overview

Since our constructions build heavily on the scheme of Aranha and Pagnin [1], we provide a technical overview of their work, which also serves as the foundation for understanding our contributions.

Following cryptographic terminology, we refer to data points as *messages*. In mklhs, messages are simply numerical values represented as elements of $\mathbb{Z}_q$, *i.e.*, as integers modulo a large prime q. In particular, any binary string of length at most $\lfloor \log_2 q \rfloor$ bits can be interpreted as an integer and represented in $\mathbb{Z}_q$.

How signatures work in mklhs. When a user signs a message using mklhs, the signature is not just a stamp of approval. It is a structured mathematical object living in an algebraic structure called a *group* (usually denoted $\mathbb{G}$), and is a set of elements where a multiplication operation is well-defined.

Concretely, mklhs signatures on a message m consist of tuples $\sigma = (\gamma, m)$, with the first component taking the form $\gamma = (H(\ell) \cdot g_1^m)^{\mathsf{sk}}$. Here, g_1 is a fixed public element of a group $\mathbb{G}_1$ and $H(\ell)$ is a hash of the label realised as another group element in $\mathbb{G}_1$. Importantly, the hash is a *one-way-function*, meaning it is infeasible to find an input that results in a specific output.

The key property of mklhs is that group operations on signature components correspond to linear combinations of the underlying messages. Given $\gamma_1 = (H(\ell_1) \cdot g_1^{m_1})^{\mathsf{sk}}$ and $\gamma_2 = (H(\ell_2) \cdot g_1^{m_2})^{\mathsf{sk}}$, we can combine them to obtain a signature component on any linear combination of their underlying messages. Indeed, for $a_1, a_2 \in \mathbb{Z}_q$, we have

$$\gamma_1^{a_1} \cdot \gamma_2^{a_2} = \Big(H(\ell_1)^{a_1} H(\ell_2)^{a_2} \cdot g_1^{a_1 m_1 + a_2 m_2}\Big)^{\mathsf{sk}},$$

which is itself a valid signature component on the linear combination $a_1 m_1 + a_2 m_2$ for the label combination $H(\ell_1)^{a_1} H(\ell_2)^{a_2}$. This shows that the scheme is *linearly homomorphic*: a server can compute a valid signature on any weighted sum of signed messages by exponentiating signatures and multiplying them together.

Verification in mklhs. The challenge is that a verifier needs to confirm the evaluated signature is correct without knowing sk or $\{m_1, m_2\}$. This is where the role of a *bilinear pairing* comes in. A bilinear pairing is a function $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ mapping elements from two groups into a third, satisfying $e(g^a, h^b) = e(g, h)^{ab}$ for all $a, b \in \mathbb{Z}_q$. This property lets the verifier move sk out of the signature in $\mathbb{G}_1$ and into the exponent of the public key $\mathsf{pk} = g_2^{\mathsf{sk}} \in \mathbb{G}_2$, since

$$e\Big(\gamma,\, g_2\Big) = e\Big((H(\ell) \cdot g_1^m)^{\mathsf{sk}},\, g_2\Big) = e\Big(H(\ell) \cdot g_1^m,\, g_2^{\mathsf{sk}}\Big) = e\Big(H(\ell) \cdot g_1^m,\, \mathsf{pk}\Big).$$

The leftmost expression involves the signature, which the server provides. The rightmost expression only involves the label hash, the claimed message, and the public key, all of which are known to the verifier. Checking this equation therefore confirms that the signature was honestly created, without ever revealing sk.

Publishing $\mathsf{pk} = g_2^{\mathsf{sk}}$ does not reveal the secret key, since recovering sk from pk amounts to solving the discrete logarithm problem in $\mathbb{G}_2$, which is assumed to be infeasible.

Type-2 and Type-3 pairings. The pairing above maps elements from two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ into a third $\mathbb{G}_T$. Sometimes, one also assumes the existence of an efficiently computable *homomorphism* $\phi\colon \mathbb{G}_2 \to \mathbb{G}_1$, which is a mapping that respects the group operation in the sense that $\phi(gh) = \phi(g)\phi(h)$ for all $g, h \in \mathbb{G}_1$. Intuitively, a homomorphism may be seen as a structure preserving translation between the two groups. When such a homomorphism is assumed to exist, we call the pairing a *Type 2 pairing*. On the contrary, if no such homomorphism is assumed exist, we call it a *Type 3 pairing*.

In modern cryptographic implementations, Type 2 bilinear pairings have largely been replaced by Type 3 bilinear pairings as the preferred standard, since they are considered more efficient [3], [4] and do not offer less security guarantees [4].

The mklhs scheme [1] was proven secure in the Type 2 pairing setting under the *co-Computational Diffie–Hellman* (co-CDH) assumption, with the original proof relying on ϕ. In the Type 3 setting, to compensate for the missing ϕ, we instead work with a variant called co-CDH*. This variant is considered at least as hard as co-CDH in the Type 3 setting [4]. Part of our contribution in this thesis is proving that mklhs remains secure in the Type 3 setting under co-CDH*, bringing it in line with current cryptographic practices.

1.2 Our Contributions

This thesis makes five total contributions, each building on the previous.

A modified security definition. The security proof of the mklhs scheme by Aranha and Pagnin [1] is argued using a security notion called *Homomorphic Unforgeability under Chosen Message Attack* (HomUF-CMA). Their proof relies on certain technical restrictions that are introduced to rule out problematic cases. We formalize these restrictions through a modified security notion called HomUF-CMA with *delayed hash queries* (HomUF-CMA-DHQ), which captures the exact setting used in the original proof more explicitly. This avoids repeating the same technical details throughout subsequent proofs and makes security analyses cleaner.

A new security proof for mklhs. The original security proof of mklhs was formulated for Type 2 bilinear pairings, which assumes an efficiently computable group homomorphism $\phi\colon \mathbb{G}_2 \to \mathbb{G}_1$ between the two source groups. The proof uses ϕ in a key step, and this map does not exist in the Type 3 setting. We give a new security proof for the mklhs construction in the Type 3 setting, following the approach of Chatterjee and Menezes [4]. To compensate for the missing ϕ, our proof reduces the security to the co-CDH* assumption in place of co-CDH, an equivalently hard

problem in the Type 3 setting. The resulting proof also achieves a tighter bound compared to the original.

Extending mklhs to quadratic evaluation. We extend mklhs to support a certain class of quadratic functions while remaining secure under the co-CDH* assumption in the Type 3 setting. The resulting construction is a pairing-based MKHS scheme that supports polynomials of the form

$$f(m_1, \ldots, m_n) = \sum_{i=1}^{n} a_i m_i + \sum_{r=1}^{R} \Big(\sum_{i=1}^{n} u_{i,r}\, m_i \Big) \Big(\sum_{i=1}^{n} v_{i,r}\, m_i \Big),$$

where the quadratic part is a sum of R products of linear polynomials, and hence has *rank* R. The evaluated signature size grows as $O(tR)$ in the number of signers t and rank R. To achieve succinctness, we restrict R to grow at most logarithmically in the number of message inputs. Hence, we refer to this construction as mkqhs-br for *multi-key quadratic homomorphic signatures* with *bounded rank*.

Compressing signatures and extending the function class. We present two independent extensions of mkqhs-br, each addressing one of its limitations, and a combined construction that inherits both extensions.

The first extension, mkqhs-$\widetilde{\text{br}}$, reduces the evaluated signature size from $O(tR)$ to $O(t + R)$, making the scheme practical for large rank R and many signers t, where the multiplicative $O(tR)$ cost of mkqhs-br would otherwise become too costly.

The second extension, mkqhs-br-m^2, extends the admissible function class to include direct square terms by having each signer additionally sign m_i^2 alongside m_i, allowing for evaluation of polynomials of the form

$$f(m_1, \ldots, m_n) = \sum_{i=1}^{n} \big(a_i m_i + b_i m_i^2 \big) + \sum_{r=1}^{R} \Big(\sum_{i=1}^{n} u_{i,r}\, m_i \Big) \Big(\sum_{i=1}^{n} v_{i,r}\, m_i \Big).$$

This enables exact evaluation of functions such as variance, least-squares loss, and squared Euclidean distance. Both extensions preserve security under the co-CDH* assumption in the Type 3 setting. Since the extensions modify independent parts of mkqhs-br, they may be applied simultaneously to create mkqhs-$\widetilde{\text{br}}$-m^2, which achieves both the compressed signature size of mkqhs-$\widetilde{\text{br}}$ and the extended function class of mkqhs-br-m^2.

Reference implementation. In addition to the theoretical contributions above, we provide a reference implementation of all developed constructions, written in Rust using the `arkworks` cryptographic library [5] and is publicly available on GitHub [6]. The implementation demonstrates the practical feasibility of our proposed constructions on statistical computations, including variance and squared Euclidean distances of signed datasets. Our library is solely intended as a research artifact accompanying the thesis.

1.3 Related Work

Homomorphic signature schemes. Homomorphic signature schemes, introduced by Johnson *et al.* [7], enable signatures on inputs of functions to be combined into a valid signature on their result. Early constructions focused on evaluation of linear functions, motivated by applications in network coding [8], [9], [10]. Support for polynomial functions was later introduced by Boneh and Freeman [11], at the cost of signature size growing with the degree of the evaluated polynomial. Catalano *et al.* [12] improved verification efficiency by splitting it into an offline precomputation phase and a cheaper online phase. Gorbunov *et al.* [13] subsequently proposed the first scheme supporting evaluation of arbitrary circuits, at the cost of fixing a maximum circuit depth at key generation.

MKHS schemes. All constructions above assume that the authenticated input originates from a single signer. This assumption is too restrictive for many practical settings such as computing statistics over data contributed from multiple independent parties, each with their own secret key. MKHS schemes address this by allowing homomorphic evaluation over messages signed under different public keys.

The concept was first considered by Agrawal *et al.* [14]. Fiore *et al.* [15] formally defined the first MKHS by extending the scheme of Gorbunov *et al.* [13] to the multi-key setting, though at the cost of signature size growing linearly with the number of signers. Fiore and Pagnin [16] later showed that any sufficiently expressive single-key homomorphic signature scheme can be generically transformed into a multi-key one, though the resulting construction supports only a bounded constant number of signers and does not yield a practical implementation. Lai *et al.* [17] both strengthened the security model to account for corrupted signers, where an adversary may obtain the secret keys of some signers, and achieved the first fully succinct MKHS at the cost of relying on non-standard assumptions. For the restricted case of linear functions, Aranha and Pagnin [1] proposed a simple pairing-based scheme with signature size linear in the number of signers, along with a concrete implementation. Finally, Anthoine *et al.* [18] resolved the problem of achieving full succinctness under standard assumptions, supporting evaluation of arbitrary circuits of unbounded depth. Their construction relies on batch arguments, which are non-trivial to implement in their case, and to the best of our knowledge no such implementation exists to date.

Lattice-based homomorphic signature schemes. An important family of homomorphic signature schemes is built on lattice-based assumptions, most notably the hardness of the Short Integer Solution (SIS) problem [19]. These constructions are of particular interest since they are believed to resist quantum computer attacks, unlike pairing-based schemes. Early constructions supported linear functions [20], and were later extended to polynomial functions and eventually arbitrary circuits [11], [13]. However, lattice-based schemes generally suffer from signature size growth during homomorphic evaluation, which limits the depth of admissible computations. In the multi-key setting, constructions achieving full succinctness exist [15], [18], but are not yet implemented due to the argument in the previous

paragraph.

Pairing-based signature schemes. Pairing-based constructions provide an efficient alternative to lattice-based approaches, using bilinear groups to achieve compact signatures and efficient verification. In the single-key setting, Catalano *et al.* [2] constructed the first pairing-based homomorphic signature scheme supporting evaluation of arbitrary multivariate polynomials. In the multi-key setting, Aranha and Pagnin [1] have constructed an implemented pairing-based construction, but with evaluation limited to linear functions. To the best of our knowledge, no implemented pairing-based construction in the multi-key setting has yet to support functions beyond linear.

Zero-knowledge proofs. A zero-knowledge proof systems allows a prover to convince a verifier that a statement is true, without revealing any information beyond the truth of the statement itself [21]. Especially relevant to homomorphic signatures are *Succinct Non-Interactive Arguments of Knowledge* (SNARKs) [22], which provide proofs whose size and verification time are very short. This succinctness makes SNARKs a natural generic tool for certifying correctness of homomorphic evaluations. There are several constructions such as [17] that use SNARKs to achieve multi-key homomorphic signatures. More recent constructions based on falsifiable assumptions [23] improve theoretical foundations, but practical efficiency remains challenging due to the computational overhead and complexity of underlying machinery.

1.4 Organisation

The remainder of this thesis is organised as follows.

Chapter 2 establishes and introduces the necessary background in mathematics and cryptography needed for the rest of the chapters.

Chapter 3 motivates and defines HomUF-CMA-DHQ, a modification of the standard MKHS security notion HomUF-CMA, that more explicitly captures the restrictions used in the original security proof of mklhs.

Chapter 4 presents an updated proof for mklhs with a tighter bound in the Type 3 pairing setting.

Chapter 5 presents our first construction, mkqhs-br, a pairing-based MKHS supporting bounded-rank quadratic evaluation, and is proved secure in the Type 3 pairing setting.

Chapter 6 introduces two extensions of mkqhs-br: (i) mkqhs-$\widetilde{\text{br}}$, a compressed variant reducing the evaluated signature size, and (ii) mkqhs-br-m^2, a message-square extension broadening the supported function class. These constructions are then combined into mkqhs-$\widetilde{\text{br}}$-m^2.

Chapter 7 discusses efficiency tradeoffs and trust assumptions across the four constructions, and concludes with a recommended default for practical use. Lastly, we present our implementation of the different schemes.

Chapter 8 discusses Weak-HomUF-CMA, the standard weakening of HomUF-CMA. We show that all our presented constructions remain secure under this more standard notion via a simple transformation.

Finally, Chapter 9 concludes the thesis and outlines directions for future work.

2

Preliminaries

This chapter covers the mathematical and cryptographic background needed for the rest of this thesis.

The first half is mathematical in nature. Section 2.1 fixes the notation and gives some basic definitions. Section 2.2 introduces the algebraic concepts our constructions rely on, including groups and bilinear pairings. Section 2.3 introduces quadratic forms over $\mathbb{Z}_q$. Section 2.4 contains the probability-theoretic tools used in our proofs.

The second half is cryptographic in nature, covering the primitives and definitions the constructions build on. Section 2.5 presents all relevant cryptographic tools, including security games, hardness assumptions, and the random oracle model. Section 2.6 gives the formal definition of multi-key homomorphic signature schemes following [15], including their correctness and security requirements. Section 2.7 presents the mklhs construction.

2.1 Basic Definitions and Notation

Bit string. A bit string of arbitrary length is denoted by $\{0,1\}^*$.

Integer set. For $n \in \mathbb{N}$ we write $[n] := \{1, \ldots, n\}$.

Sampling. We write $s \xleftarrow{\$} \mathcal{S}$ signifying the element s was uniformly and randomly sampled from the set $\mathcal{S}$.

Security parameter λ. We denote the security parameter by $\lambda \in \mathbb{N}$. It determines the size of cryptographic objects and their level of security. When an algorithm takes λ as input, it is given in unary as 1^λ, *i.e.* the string consisting of λ number of 1's.

Polynomial in λ. We write $\mathsf{poly}(\lambda)$ to denote an unspecified polynomial function of the security parameter. A value is said to be *polynomial in* λ if it is bounded by $\mathsf{poly}(\lambda)$.

Polynomial bit-length. We say a positive integer n has *polynomial bit-length* in λ if $\log_2(n) = \mathsf{poly}(\lambda)$ *i.e.*, the number of bits required to represent n grows polynomially in the security parameter.

Negligible function. We write $\mathsf{negl}(\lambda)$ to denote a negligible function in the security parameter λ. Formally, a function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}$ is said to be negligible in λ if for every polynomial $p : \mathbb{N} \to \mathbb{R}$ there exists $\lambda_0 \in \mathbb{N}$ such that for all $\lambda > \lambda_0$ it holds that

$$\mathsf{negl}(\lambda) < \frac{1}{p(\lambda)}.$$

Probabilistic algorithm. An algorithm is said to be *probabilistic* if its outcome may depend on randomness, *i.e.*, on random bits sampled during execution. On the same input, a probabilistic algorithm may therefore produce varying outputs on different runs.

Probabilistic polynomial-time (PPT). A probabilistic algorithm is said to be *probabilistic polynomial-time* (PPT) if its running time is bounded by $\mathsf{poly}(\lambda)$, *i.e.*, it terminates in at most $\mathsf{poly}(\lambda)$ steps.

Notation	Example	Description
Letters (A, a, B, b, C, c)	–	Placeholder variables used in proofs. Uppercase (A, B, C) denote group elements (*e.g.* in $\mathbb{G}$); lowercase (a, b, c) denote scalars (*e.g.* in $\mathbb{Z}_q$).
Letters (a, b, u, v)	–	Coefficients in $\mathbb{Z}_q$ for admissible functions. Note that a and b are also used as placeholder variables in proofs, the intended meaning is clear from context.
Letter q	–	A large prime with $\log_2(q) = \mathsf{poly}(\lambda)$.
Hat variables	$\hat{A}$, $\hat{b}$	Values explicitly computable by the reduction during a security proof.
Tilde variables	$\tilde{\gamma}$	Aggregated or evaluated versions of a variable, e.g. sums or homomorphically derived values.
Parenthesised superscript	$\mu^{(u)}$	A value indexed or parametrised by u, typically representing aggregation with respect to u.
Star superscript	ω^*	Values output by the adversary, *i.e.*, forgeries.
Circle superscript	μ°	Honest counterpart to the adversarial star superscript, *i.e.*, values derived from legitimate inputs.
Greek letters	μ, γ, ρ	Scheme variables.

Table 2.1: Summary of notation used in the design and security analysis of the schemes.

2.2 Groups

The constructions in this thesis are defined over cyclic groups of prime order. We recall their definitions here.

Definition 2.1 (Group). A *group* $(\mathbb{G}, \cdot)$ is a set of elements $\mathbb{G}$, together with a binary operation $(\cdot)$, that satisfy the following properties:

- *Closure*: If $g, h \in \mathbb{G}$ then $g \cdot h \in \mathbb{G}$.
- *Associativity*: For all $g, h, k \in \mathbb{G}$ we have that $(g \cdot h) \cdot k = g \cdot (h \cdot k)$.
- *Identity*: There exists an identity element $1_{\mathbb{G}} \in \mathbb{G}$ such that for every element $g \in \mathbb{G}$, $1_{\mathbb{G}} \cdot g = g \cdot 1_{\mathbb{G}} = g$.
- *Inverse*: For every element $g \in \mathbb{G}$ there exists an inverse $g^{-1} = h \in \mathbb{G}$ such that $g \cdot h = h \cdot g = 1_{\mathbb{G}}$.

The *order* of $\mathbb{G}$, denoted as $|\mathbb{G}|$ signifies the number of elements in $\mathbb{G}$.

Definition 2.2 (Cyclic Group). A group $(\mathbb{G}, \cdot)$ is called *cyclic* if there exists a single element $g \in \mathbb{G}$ such that each element in $\mathbb{G}$ can be written as g^n for some integer n. Such g is called a *generator* of $\mathbb{G}$, and is in general not unique.

The following hold for cyclic groups $\mathbb{G}$ of prime order q.

- $\mathbb{G}$ does not have any subgroups except for itself and the trivial subgroup $\{1_{\mathbb{G}}\}$ consisting only of the identity element.
- Every element $g \in \mathbb{G} \setminus \{1_{\mathbb{G}}\}$ is a generator of $\mathbb{G}$.

Remark 2.3 (The Groups $\mathbb{Z}_q$ and $\mathbb{Z}_q^*$). Let q be a prime number. The set $\mathbb{Z}_q = \{0, 1, \ldots, q-1\}$ forms a cyclic group of order q under addition modulo q, and $\mathbb{Z}_q^* = \mathbb{Z}_q \setminus \{0\}$ forms a cyclic group of order $q-1$ under multiplication modulo q. We remark that $\mathbb{Z}_q$ is in fact a *finite field*, but this is only mentioned for completeness. In this thesis, $\mathbb{Z}_q$ serves as the scalar space for all exponents: for any cyclic group $\mathbb{G}$ of prime order q with generator g, the expression g^x implies $x \in \mathbb{Z}_q$ and all arithmetic in the exponent is performed modulo q.

Remark 2.4 ($\mathbb{Z}_q^n$). For $n \in \mathbb{N}$ with $n > 1$, we write $\mathbb{Z}_q^n$ signifying the set of all column vectors $\mathbf{x} = (x_1, \ldots, x_n)^\top$ with $x_i \in \mathbb{Z}_q$.

Definition 2.5 (Homomorphism). Let $(\mathbb{G}_1, \cdot)$ and $(\mathbb{G}_2, \star)$ be groups with their respective group operations. A mapping $\phi : \mathbb{G}_1 \to \mathbb{G}_2$ is a *(group) homomorphism* if

$$\phi(x \cdot y) = \phi(x) \star \phi(y),$$

for all $x, y \in \mathbb{G}_1$.

Example 2.6. The homomorphism property is what enables homomorphic signature schemes by allowing operations on messages to translate into operations on

signatures. For example, if we let the signature σ_i for message $m_i \in \mathbb{G}_1$ be defined as $\sigma_i = \phi(m_i) \in \mathbb{G}_2$, then

$$\phi(m_1 \cdot m_2) = \phi(m_1) \star \phi(m_2) = \sigma_1 \star \sigma_2.$$

Thus, the combined signature $\sigma_1 \star \sigma_2$ corresponds to the signature of the combined message $m_1 \cdot m_2$.

2.2.1 Bilinear Pairings

The constructions in this thesis also rely on additional algebraic concepts built on top of cyclic groups, namely bilinear pairings.

Definition 2.7 (Admissible Bilinear Pairings [3]). An *admissible bilinear pairing* is a map

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T,$$

where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are cyclic groups of prime order q, satisfying:

- *Bilinearity*: For all $u \in \mathbb{G}_1, v \in \mathbb{G}_2$, and all $a, b \in \mathbb{Z}_q$,

$$e(u^a, v^b) = e(u^{ab}, v) = e(u, v^{ab}) = e(u, v)^{ab}.$$

- *Non-degeneracy*: $e(g_1, g_2) \neq 1_{\mathbb{G}_T}$ for generators g_1 of $\mathbb{G}_1$ and g_2 of $\mathbb{G}_2$.
- *Efficient computability*: There exists a polynomial-time algorithm that computes $e(u, v)$ for all $u \in \mathbb{G}_1, v \in \mathbb{G}_2$.

A useful consequence of non-degeneracy is that fixing a generator in either argument of e results in an injective map.

Corollary 2.8 (Injectivity from Non-Degeneracy). *Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be an admissible bilinear pairing with generators $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$. Fixing a generator in either argument of e results in an injective map. More concretely, for all $X, X' \in \mathbb{G}_1$ and all $Y, Y' \in \mathbb{G}_2$,*

$$\begin{aligned} &\textit{if} \quad e(X, g_2) = e(X', g_2) \quad \textit{then} \quad X = X', \\ &\textit{if} \quad e(g_1, Y) = e(g_1, Y') \quad \textit{then} \quad Y = Y'. \end{aligned}$$

Proof. Suppose $e(X, g_2) = e(X', g_2)$. Multiplying both sides by $e(X', g_2)^{-1}$ and using bilinearity, we get $e(X(X')^{-1}, g_2) = 1_{\mathbb{G}_T}$. Non-degeneracy then forces $X(X')^{-1}$ to not be a generator of $\mathbb{G}_1$, and since the identity is the only non-generator in a cyclic prime-order group, $X(X')^{-1} = 1_{\mathbb{G}_T}$. Hence $X = X'$. The second implication follows by the same argument on $\mathbb{G}_2$. □

Admissible pairings can further be classified by the relationship between the input groups.

Definition 2.9 (Admissible Bilinear Pairing Types [3]). An admissible bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ can be classified into three categories:

- *Type 1*: $\mathbb{G}_1 = \mathbb{G}_2$.
- *Type 2*: $\mathbb{G}_1 \neq \mathbb{G}_2$ and there exists an efficiently computable homomorphism $\phi : \mathbb{G}_2 \to \mathbb{G}_1$.
- *Type 3*: $\mathbb{G}_1 \neq \mathbb{G}_2$ and there are no known efficiently computable homomorphisms between $\mathbb{G}_1$ and $\mathbb{G}_2$.

In practice, Type 3 pairings are preferred over Type 1 and Type 2 ones, since they are considered more efficient [3], [4] and there are no security benefits to using Type 2 over Type 3 [4]. Throughout this thesis, we therefore work exclusively in the Type 3 setting. However, since the original mklhs construction of Aranha and Pagnin [1] is formulated in the Type 2 setting, and one of our contributions is extending its security proof to the Type 3 setting, we introduce notation for both.

We write $\mathsf{BilinGroup}_2(1^\lambda)$ for a probabilistic algorithm that on input 1^λ outputs a description $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, q, e, \phi)$, where

- $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are cyclic groups of prime order q such that $\log_2(q) = \mathsf{poly}(\lambda)$,
- $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ are generators,
- $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an admissible bilinear pairing of Type 2,
- $\phi : \mathbb{G}_2 \to \mathbb{G}_1$ is an efficiently computable homomorphism.

For the Type 3 pairing setting, we write $\mathsf{BilinGroup}_3(1^\lambda)$ for a probabilistic algorithm that on input 1^λ outputs a description $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, q, e)$, where

- $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are cyclic groups of prime order q such that $\log_2(q) = \mathsf{poly}(\lambda)$,
- $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ are generators,
- $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an admissible bilinear pairing of Type 3.

2.3 Quadratic Forms Over $\mathbb{Z}_q$

Many practical computations require going beyond linear where the natural step is quadratic polynomials. This section develops the tools needed to handle such terms in our cryptographic constructions.

We begin by fixing some basic terminology. By a *monomial* we refer to polynomials that are products of the form $c \cdot m_1^{e_1} \cdots m_n^{e_n}$, where $c, m_i \in \mathbb{Z}_q$ with exponents $e_i \in \mathbb{N}$. A polynomial is said to be a *degree-d form*, if each of its monomials has degree exactly d.

Example 2.10. The polynomial $f(m_1, m_2, m_3) = m_1^2 + 3m_1m_3 + 2m_2^2$ is a degree-2 form, or a *quadratic form*, since each of its monomials has degree 2. Conversely,

the polynomial $f(m_1, m_2) = 1 + m_1^2 + m_2$ is not a form, since its monomials have differing degrees.

2.3.1 Rank One Decompositions

Let $\mathbf{m} = (m_1, \ldots, m_n)^\top \in \mathbb{Z}_q^n$. Given vectors $\mathbf{u}, \mathbf{v} \in \mathbb{Z}_q^n$, we write $\mathbf{u}^\top \mathbf{m}$ or $\langle \mathbf{u}, \mathbf{m} \rangle$ for the *inner product* over $\mathbb{Z}_q$, *i.e.*,

$$\mathbf{u}^\top \mathbf{m} = \langle \mathbf{u}, \mathbf{m} \rangle = \sum_{i=1}^{n} u_i m_i.$$

Furthermore, $\mathbf{u}^\top \mathbf{m}$ is a degree-1 form, or a *linear form*, in the variables $\mathbf{m} = (m_1, \ldots, m_n)^\top$.

We write $\mathbf{u}\mathbf{v}^\top$ for the *outer product* of $\mathbf{u}, \mathbf{v} \in \mathbb{Z}_q^n$, namely

$$\mathbf{u}\mathbf{v}^\top = \begin{pmatrix} u_1 v_1 & u_1 v_2 & \cdots & u_1 v_n \\ u_2 v_1 & u_2 v_2 & \cdots & u_2 v_n \\ \vdots & \vdots & \ddots & \vdots \\ u_n v_1 & u_n v_2 & \cdots & u_n v_n \end{pmatrix}.$$

Equivalently, $\mathbf{u}\mathbf{v}^\top$ is the matrix whose (i, j)-th entry is given by $u_i v_j$.

Definition 2.11 (Rank One Quadratic Forms). A quadratic form $f(\mathbf{m})$ is called *rank one* if it can be written as the product of two linear forms,

$$f(\mathbf{m}) = \left(\sum_{i=1}^{n} u_i m_i \right) \left(\sum_{i=1}^{n} v_i m_i \right) = (\mathbf{u}^\top \mathbf{m})(\mathbf{v}^\top \mathbf{m}), \tag{2.1}$$

for non-zero vectors $\mathbf{u}, \mathbf{v} \in \mathbb{Z}_q^n$.

Definition 2.12 (Rank One Matrices). A matrix $Q \in \mathbb{Z}_q^{n \times n}$ is called *rank one*, written $\mathrm{rank}(Q) = 1$, if it can be expressed as an outer product $Q = \mathbf{u}\mathbf{v}^\top$ for non-zero vectors $\mathbf{u}, \mathbf{v} \in \mathbb{Z}_q^n$.

Equivalently, a matrix Q has rank one if and only if the maximum number of linearly independent columns (equivalently, rows) in Q is one.

Remark 2.13. Any quadratic form f can be written as

$$f(\mathbf{m}) = \sum_{i=1}^{n} \sum_{j=1}^{n} q_{i,j} m_i m_j = \mathbf{m}^\top Q \mathbf{m}, \tag{2.2}$$

for $\mathbf{m} = (m_1, \ldots, m_n)^\top$ and some matrix $Q \in \mathbb{Z}_q^{n \times n}$. In fact, we may express all rank one quadratic forms using rank one matrices by rearranging (2.1) as

$$\begin{aligned} f(\mathbf{m}) &= (\mathbf{u}^\top \mathbf{m})(\mathbf{v}^\top \mathbf{m}) \\ &= \mathbf{m}^\top \mathbf{u}\, \mathbf{v}^\top \mathbf{m} \\ &= \mathbf{m}^\top (\mathbf{u}\mathbf{v}^\top) \mathbf{m}. \end{aligned}$$

Definition 2.14 (Rank R Matrices). A matrix $Q \in \mathbb{Z}_q^{n\times n}$ is said to have *rank* R, written $\text{rank}(Q) = R$, if R is the smallest positive integer for which there exist non-zero vectors $\mathbf{u}_1, \ldots, \mathbf{u}_R, \mathbf{v}_1, \ldots, \mathbf{v}_R \in \mathbb{Z}_q^n$ such that

$$Q = \sum_{r=1}^{R} \mathbf{u}_r \mathbf{v}_r^\top.$$

Equivalently, a matrix Q has rank R if and only if R is the maximum number of linearly independent columns (equivalently, the maximum number of linearly independent rows) of Q.

Corollary 2.15. *Every quadratic form $f(\mathbf{m}) = \mathbf{m}^\top Q\,\mathbf{m}$ over $\mathbb{Z}_q^n$ can be expressed as a sum of at most n rank one quadratic forms.*

Proof. Since $Q \in \mathbb{Z}_q^{n\times n}$, its rank satisfies $\text{rank}(Q) \le n$. Writing $R = \text{rank}(Q)$ and decomposing $Q = \sum_{r=1}^{R} \mathbf{u}_r \mathbf{v}_r^\top$ as in Definition 2.14, substitution into (2.2) yields

$$\begin{aligned} f(\mathbf{m}) &= \mathbf{m}^\top \left(\sum_{r=1}^{R} \mathbf{u}_r \mathbf{v}_r^\top \right) \mathbf{m} \\ &= \sum_{r=1}^{R} \mathbf{m}^\top (\mathbf{u}_r \mathbf{v}_r^\top) \mathbf{m} \\ &= \sum_{r=1}^{R} (\mathbf{u}_r^\top \mathbf{m})(\mathbf{v}_r^\top \mathbf{m}), \end{aligned}$$

where each term $(\mathbf{u}_r^\top \mathbf{m})(\mathbf{v}_r^\top \mathbf{m})$ is a rank one quadratic form. □

Example 2.16 (Rank of the Identity Matrix). An important example and observation is that $\text{rank}(I_n) = n$, where I_n is the identity matrix. This follows directly from the fact that all n columns in I_n are linearly independent.

Hence, in order to express I_n as a sum of rank one matrices, we need at least n rank one terms. Let $\mathbf{e}_i \in \mathbb{Z}_q^n$ denote the i-th standard basis vector, *i.e.*, the vector whose i-th entry is 1, and the rest 0. Then

$$I_n = \sum_{i=1}^{n} \mathbf{e}_i \mathbf{e}_i^\top.$$

Each outer product $\mathbf{e}_i \mathbf{e}_i^\top$ is a rank one matrix having a single 1 in the (i, i)-th entry and 0s elsewhere.

Furthermore, the quadratic form associated to I_n is

$$\mathbf{m}^\top I_n \mathbf{m} = \sum_{i=1}^{n} m_i^2 = \sum_{i=1}^{n} (\mathbf{e}_i^\top \mathbf{m})^2.$$

That is, each square term m_i^2 contributes one independent rank one component.

2.4 Probability Theory

Security in cryptography often relies on probabilistic analysis. This section collects important probabilistic tools used throughout this thesis.

Theorem 2.17 (Law of Total Probability [24]). *Let E be an event and let Y be a random variable taking values in a set $\mathcal{Y}$. Then*

$$\Pr[E] \;=\; \sum_{y \in \mathcal{Y}} \Pr[Y = y] \cdot \Pr[E \mid Y = y].$$

A complementary tool is the union bound, which allows us to bound the probability that any one of several events occurs.

Theorem 2.18 (Union Bound). *For any events $A_1, \ldots, A_n$,*

$$\Pr[A_1 \vee \cdots \vee A_n] \;\leq\; \sum_{i=1}^{n} \Pr[A_i].$$

Another tool we use is the Schwartz–Zippel lemma, which bounds how often a non-zero polynomial can evaluate to zero when its variables are sampled uniformly at random.

Theorem 2.19 (Schwartz–Zippel [25], [26]). *Let q be prime and $f(x_1, \ldots, x_n)$ be a non-zero polynomial of degree $d < q$, where all coefficients are in $\mathbb{Z}_q$. Let $\mathcal{S}$ be a finite subset of $\mathbb{Z}_q$ and $(\rho_1, \ldots, \rho_n) \xleftarrow{\$} \mathcal{S}^n$. Then*

$$\Pr[f(\rho_1, \ldots, \rho_n) = 0] \leq \frac{d}{|\mathcal{S}|}.$$

2.5 Cryptographic Primitives

This section introduces the cryptographic background used throughout this thesis. We begin by recalling security games and reductions, before stating the hardness assumptions our security proofs rely on. Lastly, we introduce the random oracle model, which our security proofs are formulated in.

2.5.1 Security Games

Security in modern cryptography is often defined using *security games* [27]. Rather than informally saying a scheme is secure if no attacker can break it, a security game formalises both what it means to break a scheme and what resources could be used when doing so. This allows security to be reasoned about precisely and compared across different constructions. In this work, we consider security games of the following form.

Challenger and adversary. A security game is generally an interaction between two parties: a *challenger* $\mathcal{C}$ and an *adversary* $\mathcal{A}$. The challenger models an honest execution of some scheme, while the adversary models a PPT attacker that tries to violate the security of that scheme.

Structure of a security game. For our purposes, a security game generally proceeds in the following phases.

1. **Setup:** $\mathcal{C}$ starts the game by generating all public information and runs the setup algorithm to generate public parameters. All public parameters are given to the $\mathcal{A}$.
2. **Query phase:** $\mathcal{A}$ may adaptively issue queries to $\mathcal{C}$, who responds according to the specification of the scheme and the rules of the game.
3. **Challenge phase:** Eventually, $\mathcal{A}$ returns a value to $\mathcal{C}$ intended to demonstrate that it has broken the scheme. The game then determines whether this output meets the winning condition.

Winning condition and advantage. Each security game defines a *winning condition*, a specific criterion that determines whether $\mathcal{A}$ has succeeded in breaking the scheme. The game outputs 1 if this condition is met and 0 otherwise. The *advantage* of $\mathcal{A}$ in a game Game is defined as

$$\mathsf{Adv}^{\mathsf{Game}}_{\mathcal{A},\mathsf{Scheme}}(\lambda) = \Pr[\mathsf{Game}_{\mathcal{A}}(\lambda) = 1]\,,$$

where the probability is over all possible outcomes of the random bits sampled during execution by both $\mathcal{A}$ and $\mathcal{C}$. A scheme is said to be *secure* with respect to a given notion if the advantage of every PPT adversary is negligible in the security parameter.

Reductions. Security proofs are typically carried out via *reductions*. To show that a scheme is secure, one assumes the existence of a PPT adversary $\mathcal{A}$ that wins the security game with non-negligible advantage, and constructs another algorithm $\mathcal{B}$ that uses $\mathcal{A}$ as a subroutine to solve an underlying hard problem. Concretely, $\mathcal{B}$ simulates the role of the challenger for $\mathcal{A}$ while embedding an instance of the hard problem into the simulation. If $\mathcal{A}$ succeeds in breaking the scheme, $\mathcal{B}$ can extract a solution to the hard problem. Conversely, if no PPT algorithm can solve the hard problem, then no PPT adversary $\mathcal{A}$ can exist either, thus the scheme is secure.

An important requirement is that the simulation is *indistinguishable* from a real execution of the scheme. If $\mathcal{A}$ could detect that it is interacting with a reduction rather than an honest challenger, it might behave differently and the argument would break.

2.5.2 Hardness Assumptions

The security of cryptographic schemes typically reduces to the hardness of a computational problem, meaning that breaking the scheme under a security game is at

least as hard as solving a problem believed to be infeasible for any PPT algorithm to solve. We now introduce the hardness assumptions used throughout this thesis, following [28].

Assumption 2.20 (co-CDH Hardness Assumption). The *co-Computational Diffie–Hellman problem* (co-CDH), also called *co-Diffie–Hellman Problem* (co-DHP), states that given $g_1 \in \mathbb{G}_1$ and $g_2, g_2^x \in \mathbb{G}_2$, it is hard to compute $g_1^x \in \mathbb{G}_1$ for unknown $x \xleftarrow{\$} \mathbb{Z}_q$. Formally, consider the following game.

co-CDH$_{\mathcal{A}}(\lambda)$

1 : $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, q, e, \phi) \xleftarrow{\$} \mathsf{BilinGroup}_2(1^\lambda)$
2 : $x \xleftarrow{\$} \mathbb{Z}_q$
3 : $X_2 := g_2^x$
4 : $\omega^* \leftarrow \mathcal{A}(g_1, g_2, X_2)$
5 : **return** $[\omega^* == g_1^x]$

The game returns 1 if the adversary $\mathcal{A}$ correctly computes g_1^x. The advantage of $\mathcal{A}$ in solving the co-CDH problem is defined as

$$\mathsf{Adv}_{\mathcal{A}}^{\text{co-CDH}}(\lambda) = \Pr[\text{co-CDH}_{\mathcal{A}}(\lambda) = 1]\,.$$

We say that the co-CDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$ if no PPT adversary can win the above game with non-negligible probability, *i.e.*, if for all PPT adversaries $\mathcal{A}$,

$$\mathsf{Adv}_{\mathcal{A}}^{\text{co-CDH}}(\lambda) \leq \mathsf{negl}(\lambda).$$

Assumption 2.21 (co-CDH* Hardness Assumption). The co-CDH* assumption is at least as hard as co-CDH. Informally, given $g_1, g_1^x, h \in \mathbb{G}_1$, and $g_2, g_2^x \in \mathbb{G}_2$, it is hard to compute $h^x \in \mathbb{G}_1$ for an unknown $x \xleftarrow{\$} \mathbb{Z}_q$. Formally, consider the following game.

co-CDH*$_{\mathcal{A}}(\lambda)$

1 : $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, q, e) \xleftarrow{\$} \mathsf{BilinGroup}_3(1^\lambda)$
2 : $h \xleftarrow{\$} \mathbb{G}_1$
3 : $x \xleftarrow{\$} \mathbb{Z}_q$
4 : $X_1 := g_1^x$
5 : $X_2 := g_2^x$
6 : $\omega^* \leftarrow \mathcal{A}(g_1, X_1, h, g_2, X_2)$
7 : **return** $[\omega^* == h^x]$

The output from the game is 1 if the adversary $\mathcal{A}$ correctly computes h^x. The advantage of $\mathcal{A}$ in solving the co-CDH* problem is defined as

$$\mathsf{Adv}_{\mathcal{A}}^{\text{co-CDH}^*}(\lambda) = \Pr[\text{co-CDH}^*{}_{\mathcal{A}}(\lambda) = 1]\,.$$

The co-CDH* assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$ if for all probabilistic polynomial-time adversaries $\mathcal{A}$,

$$\mathsf{Adv}_{\mathcal{A}}^{\text{co-CDH}^*}(\lambda) \leq \mathsf{negl}(\lambda).$$

Remark 2.22. In the Type 2 pairing setting, there is an efficiently computable homomorphism $\phi : \mathbb{G}_2 \to \mathbb{G}_1$, and Chatterjee *et al.* use it to show that co-CDH and co-CDH* are equivalent via reductions in both directions [29, Lemma 2].

In the Type 3 pairing setting, no such homomorphism ϕ is known, and the reduction from co-CDH* to co-CDH used in the Type 2 setting no longer applies. However, any co-CDH solver $\mathcal{A}$ such that $g_1^x \leftarrow \mathcal{A}(g_1, g_2, g_2^x)$ solves a co-CDH* instance $(g_1, g_1^x, h, g_2, g_2^x)$ by calling $h^x \leftarrow \mathcal{A}(h, g_1, g_1^x)$.

Hence, co-CDH* is considered at least as hard as co-CDH in the Type 3 setting [4], and we base our security reductions on the co-CDH* assumption to account for the lack of ϕ.

2.5.3 Random Oracle Model

Definition 2.23 (Random Oracle). A *random oracle* $\mathcal{H} : \mathcal{X} \to \mathcal{Y}$ is a black-box function that on every new input $x \in \mathcal{X}$ returns a value $y \xleftarrow{\$} \mathcal{Y}$ sampled uniformly at random, and then stores the pair (x, y) internally. On repeated queries with the same input x, the oracle returns the same previously sampled value y.

A cryptographic construction that is proven secure when some hash function is replaced by a random oracle is said to be *secure in the random oracle model*.

Importantly, the random oracle model allows the challenger in a reduction to program the random oracle. This means that the challenger can choose the oracle output for all fresh queries, as long as the returned values still appear uniformly random and independent to the adversary.

We also note that the random oracle does not correspond to any implementable hash function, but instead acts as an abstraction that allows for easier security analyses.

2.6 Multi-Key Homomorphic Signature Schemes

We recall the formal definition of multi-key homomorphic signatures (MKHS) following [15]. We begin by defining the syntax of a MKHS scheme, and then present its correctness requirements and security notion.

A MKHS scheme is defined with respect to the following spaces:

Message space $\mathcal{M}$. The set of all messages that can be signed and verified by the scheme.

Tag space $\mathcal{T} = \{0,1\}^n$. Tags are identifiers attached to messages in order to distinguish different pieces of data belonging to the same user or dataset. Intuitively, tags can be thought of as the equivalent of filenames for files.

Admissible functions space $\mathcal{F}$. Admissible functions are the functions that the scheme explicitly supports for homomorphic evaluation.

Identity space $\mathsf{ID} = \{0,1\}^n$. Identities are unique identifiers for each user participating in the scheme.

To connect messages to the users who signed them, we introduce the notion of a *label*, which binds a message to a specific identity and tag.

Definition 2.24 (Label). A *label* $\ell = (\mathsf{id}, \tau) \in \mathsf{ID} \times \mathcal{T}$ uniquely identifies a message m within a dataset Δ, binding it to an identity id and tag τ.

Using these labels, we now define a way to identify the input to an admissible function via a labeled program.

Definition 2.25 (Labeled Program). Following [30], a *labeled program* $\mathcal{P}$ is a tuple $(f, \{\ell_i\}_{i=1}^n)$ consisting of an admissible function $f : \mathcal{M}^n \to \mathcal{M}$ and labels $\{\ell_i\}_{i=1}^n$ that uniquely identify the i-th input of f.

We write $\mathsf{id} \in \mathcal{P}$ to denote a distinct identity appearing in the labeled program $\mathcal{P}$, and $|\mathsf{id} \in \mathcal{P}|$ for the number of such identities.

Definition 2.26 (Identity Program). We define the *identity program* for label ℓ as the labeled program $\mathcal{I}_\ell = (f_\mathcal{I}, \ell)$, where $f_\mathcal{I} : \mathcal{M} \to \mathcal{M}$ is the identity function defined by $f_\mathcal{I}(m) = m$ for all $m \in \mathcal{M}$.

We now define the full syntax of a MKHS scheme.

Definition 2.27 (Multi-Key Homomorphic Signature Scheme [15]). A MKHS scheme is a tuple of PPT algorithms

$$\mathsf{MKHS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Eval}, \mathsf{Verify})$$

defined as the following.

$\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$ takes a security parameter λ, and outputs public parameters pp describing the spaces ID, $\mathcal{T}$, $\mathcal{M}$, and $\mathcal{F}$.

$\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{sk}_\mathsf{id}, \mathsf{pk}_\mathsf{id})$ takes public parameters pp and outputs a key pair $(\mathsf{sk}_\mathsf{id}, \mathsf{pk}_\mathsf{id})$ for some identity $\mathsf{id} \in \mathsf{ID}$.

$\mathsf{Sign}(\mathsf{sk}_\mathsf{id}, \Delta, \ell, m) \to \sigma$ takes a dataset identifier Δ, a label $\ell = (\mathsf{id}, \tau)$ with $\tau \in \mathcal{T}$ identifying the message $m \in \mathcal{M}$, and outputs a signature σ.

$\mathsf{Eval}(\mathcal{P}, \{\mathsf{pk}_\mathsf{id}\}_{\mathsf{id} \in \mathcal{P}}, \{\sigma_i\}_{i=1}^n) \to \tilde{\sigma}$ takes a labeled program $\mathcal{P} = (f, \{\ell_i\}_{i=1}^n)$, the set of public keys for all involved identities, and their corresponding signatures $\{\sigma_i\}_{i=1}^n$. It outputs an evaluated signature $\tilde{\sigma}$ verifiable against the output of f on messages corresponding to each σ_i.

$\mathsf{Verify}(\mathcal{P}, \Delta, \{\mathsf{pk_{id}}\}_{\mathsf{id}\in\mathcal{P}}, \tilde{m}, \tilde{\sigma}) \to \{0,1\}$ outputs 1 (accept) if σ_f is accepted as a valid signature, attesting that $\tilde{m}$ is the output of program $\mathcal{P}$ on the inputs identified by $\{\ell_i\}_{i=1}^n$ under the dataset identifier Δ and public keys $\{\mathsf{pk_{id}}\}_{\mathsf{id}\in\mathcal{P}}$. Otherwise, Verify outputs 0 (reject).

Figure 2.1 illustrates the interactions between the different parties using the above algorithms.

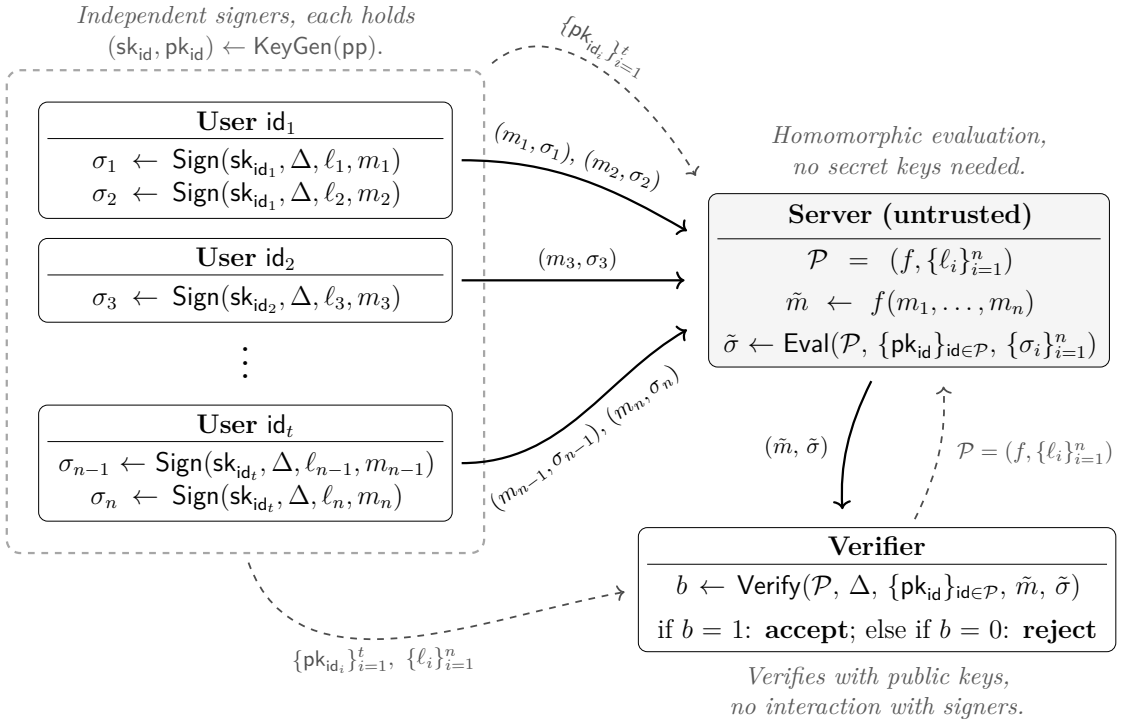


Figure 2.1: A detailed illustration of a multi-key homomorphic signature scheme. Each of t independent signers hold a key pair $(\mathsf{sk_{id}}, \mathsf{pk_{id}}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ and sign their messages under a shared dataset identifier Δ with labels $\ell_i = (\mathsf{id}_i, \tau_i)$. The untrusted server holds the labeled program $\mathcal{P} = (f, \{\ell_i\}_{i=1}^n)$, evaluates $\tilde{m} = f(m_1, \ldots, m_n)$, and derives $\tilde{\sigma} \leftarrow \mathsf{Eval}(\mathcal{P}, \{\mathsf{pk_{id}}\}, \{\sigma_i\})$ without knowledge of any secret key. The verifier checks $\mathsf{Verify}(\mathcal{P}, \Delta, \{\mathsf{pk_{id}}\}, \tilde{m}, \tilde{\sigma})$ using only the public keys, and outputs accept or reject. Solid arrows denote explicit communication between the parties. Dashed arrows denote implicit communication: public keys and the labeled program $\mathcal{P}$ are assumed known to the verifier without being sent directly.

Additionally, a MKHS scheme must satisfy the properties of authentication correctness, evaluation correctness and succinctness.

Authentication correctness. Informally, *authentication correctness* states that an honestly signed message should always verify correctly.

Formally, for any security parameter λ, let $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ and $(\mathsf{sk_{id}}, \mathsf{pk_{id}}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$. A MKHS scheme satisfies authentication correctness if for any dataset

identifier Δ, message $m \in \mathcal{M}$ and label ℓ with corresponding identity program $\mathcal{I}_\ell$, letting $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{id}}, \Delta, \ell, m)$ and $\sigma_f \leftarrow \mathsf{Eval}(\mathcal{I}_\ell, \mathsf{pk}_{\mathsf{id}}, \sigma)$, it holds that

$$\Pr[\mathsf{Verify}(\mathcal{I}_\ell, \Delta, \mathsf{pk}_{\mathsf{id}}, m, \tilde{\sigma}) = 1] = 1 - \mathsf{negl}(\lambda).$$

Evaluation correctness. Informally, *evaluation correctness* states that honestly evaluated signatures should always verify correctly.

Formally, for any security parameter λ, let $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ and $(\mathsf{sk}_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ for all identities id in a labeled program $\mathcal{P}$. A MKHS scheme satisfies evaluation correctness if for any dataset identifier Δ and messages $m_1, \ldots, m_n \in \mathcal{M}$, letting $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{id}_i}, \Delta, \ell_i, m_i)$ for $i \in [n]$, $\tilde{\sigma} \leftarrow \mathsf{Eval}\Big(\mathcal{P}, \{\mathsf{pk}_{\mathsf{id}_i}\}_{i=1}^n, \{\sigma_i\}_{i=1}^n\Big)$ and $\tilde{m} = f(m_1, \ldots, m_n)$, it holds that

$$\Pr\Big[\mathsf{Verify}\Big(\mathcal{P}, \Delta, \{\mathsf{pk}_{\mathsf{id}_i}\}_{i=1}^n, \tilde{m}, \tilde{\sigma}\Big) = 1\Big] = 1 - \mathsf{negl}(\lambda).$$

Succinctness. Informally, *succinctness* requires that verifying an evaluated signature does not require access to all individual messages. If the verifier needed all n input messages to check the result, it would defeat the purpose of outsourcing the computation in the first place. Instead, the derived signature should remain compact.

Formally, let $t = |\,\mathsf{id} \in \mathcal{P}\,|$ denote the number of distinct identities appearing in the labels of any labeled program $\mathcal{P}$. For any dataset identifier Δ and messages $m_1, \ldots, m_n \in \mathcal{M}$, letting $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{id}_i}, \Delta, \ell_i, m_i)$ for $i \in [n]$ and $\tilde{\sigma} \leftarrow \mathsf{Eval}\Big(\mathcal{P}, \{\mathsf{pk}_{\mathsf{id}_i}\}_{i=1}^n, \{\sigma_i\}_{i=1}^n\Big)$, a MKHS scheme is *succinct* if the size of the derived signature $\tilde{\sigma}$ satisfies

$$|\tilde{\sigma}| \leq \mathsf{poly}(\lambda, t, \log n).$$

Security. We now present a security notion for MKHS. The original experiment was introduced by Fiore *et al.* [15] under the name *Homomorphic Unforgeability under Chosen Message Attack* (HomUF-CMA). Aranha and Pagnin [1] present an equivalent formulation, which we follow here with minor syntactic changes.

Intuitively, the HomUF-CMA security experiment requires that an adversary, even after obtaining signatures on messages of its choice, cannot produce a valid signature on the output of a labeled program, unless that output corresponds to a correct evaluation on previously authenticated inputs.

Definition 2.28 (Homomorphic Unforgeability under Chosen Message Attack Security Game – HomUF-CMA [1], [15]). The HomUF-CMA security experiment is played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, using a MKHS scheme in the random oracle model.

The HomUF-CMA-DHQ security game is played in the following stages.

Setup. The challenger $\mathcal{C}$ runs $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ and returns pp to the adversary $\mathcal{A}$. $\mathcal{C}$ then initialises the bookkeeping lists

$$L_{\mathsf{ID}} \leftarrow \varnothing, \quad L_\sigma \leftarrow \varnothing, \quad L_{\mathsf{corr}} \leftarrow \varnothing,$$

to keep track of generated identities, signing queries, and corrupted identities, respectively. Each list entry may also store auxiliary information, denoted by aux, used for internal bookkeeping by $\mathcal{C}$.

Identity query phase. $\mathcal{A}$ can adaptively submit identity queries of the form $\mathsf{id} \in \mathsf{ID}$. Whenever $(\mathsf{id}, \cdot, \cdot, \cdot) \notin L_{\mathsf{ID}}$, $\mathcal{C}$ generates new keys for this identity by running $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, and stores

$$L_{\mathsf{ID}} \leftarrow L_{\mathsf{ID}} \cup \{(\mathsf{id}, \mathsf{sk}, \mathsf{pk}, \mathsf{aux})\}.$$

and returns pk to $\mathcal{A}$. Whenever $(\mathsf{id}, \cdot, \cdot, \cdot) \in L_{\mathsf{ID}}$, $\mathcal{C}$ interprets the query as a corruption query and updates the list of corrupted identities

$$L_{\mathsf{corr}} \leftarrow L_{\mathsf{corr}} \cup \{(\mathsf{id}, \mathsf{aux})\},$$

retrieves $(\mathsf{id}, \mathsf{sk}, \cdot, \cdot) \in L_{\mathsf{ID}}$ and returns sk to $\mathcal{A}$.

Sign query phase. $\mathcal{A}$ can adaptively submit queries of the form $(\Delta, \ell, m, \cdot)$, where Δ is a database identifier, $\ell = (\mathsf{id}, \tau)$ is a label in $\mathsf{ID} \times \mathcal{T}$ and $m \in M$ is a message. $\mathcal{C}$ ignores the query whenever

- $(\Delta, \ell, \cdot, \cdot) \in L_\sigma$, *i.e.*, $\mathcal{A}$ has already asked a signature for label ℓ in the dataset Δ; or
- $(\mathsf{id}, \cdot, \cdot, \cdot) \notin L_{\mathsf{ID}}$, *i.e.*, the keys of the identity specified in $\ell = (\mathsf{id}, \tau)$ have not yet been generated.

Otherwise, $\mathcal{C}$ computes $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{id}}, \Delta, \ell, m)$, updates the list

$$L_\sigma \leftarrow L_\sigma \cup (\Delta, \ell, m, \mathsf{aux}),$$

and returns σ to $\mathcal{A}$.

Forgery. At the end of the game, $\mathcal{A}$ outputs a tuple

$$(\mathcal{P}^*, \Delta^*, \{\mathsf{pk}^*_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}^*}, \tilde{m}^*, \tilde{\sigma}^*).$$

The experiment outputs 1 if the tuple returned by $\mathcal{A}$ is a $\mathsf{HomUF\text{-}CMA}$ forgery (defined below), and 0 otherwise.

Having defined the experiment, we now specify what it means for the adversary to win. Intuitively, a forgery is any valid signature that the adversary could not have produced by honestly combining previously authenticated inputs.

Definition 2.29 ($\mathsf{HomUF\text{-}CMA}$ Forgery [1], [15]). Assume an execution of $\mathsf{HomUF\text{-}CMA}$ between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, where

$$(\mathcal{P}^*, \Delta^*, \{\mathsf{pk}^*_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}^*}, \tilde{m}^*, \tilde{\sigma}^*)$$

is the tuple returned by $\mathcal{A}$ and $\mathcal{P}^* = (f^*, \{\ell_i^*\}_{i=1}^n)$. We say that the adversary outputs a $\mathsf{HomUF\text{-}CMA}$ *forgery* if

$$\mathsf{Verify}(\mathcal{P}^*, \Delta^*, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}^*}, \tilde{m}^*, \tilde{\sigma}^*) = 1,$$

and at least one of the following hold:

Type-1 forgery. The database Δ^* was never initialized during the game, *i.e.*, $(\Delta^*, \cdot, \cdot) \notin L_\sigma$.

Type-2 forgery. $\tilde{\sigma}^*$ is an incorrect evaluation of f^* on previously signed messages by uncorrupted identites, *i.e.*, for all $\mathsf{id} \in \mathcal{P}^*$, $\mathsf{id} \notin L_{\mathsf{corr}}$ and $(\Delta^*, \ell_i^*, m_i, \cdot) \in L_\sigma$ for all $i \in [n]$, but $\tilde{m}^* \neq f^*(m_1, \ldots, m_n)$.

Type-3 forgery. There exists (at least) one index $i \in [n]$ such that ℓ_i^* was never queried, *i.e.*, $(\Delta^*, \ell_i^*, \cdot, \cdot) \notin L_\sigma$ and corresponding $\mathsf{id} \notin L_{\mathsf{corr}}$ is a non-corrupted identity.

Figure 2.2 illustrates the HomUF-CMA security game for a MKHS scheme.

We note that these forgery-types are not mutually exclusive. It is possible for a single adversarial output to simultaneously satisfy one condition. However, it suffices to show that no PPT adversary can produce a forgery of any type with non-negligible probability for the scheme to be secure, which is formalised by the following definition.

Definition 2.30 (HomUF-CMA Unforgeable [1], [15]). A MKHS scheme is said to be HomUF-CMA *unforgeable* if, for every PPT adversary $\mathcal{A}$, its advantage in winning the HomUF-CMA game is negligible in the security parameter of the scheme. Formally,

$$\mathsf{Adv}^{\mathsf{HomUF\text{-}CMA}}_{\mathcal{A},\,\mathsf{MKHS}}(\lambda) = \Pr[\mathcal{A} \text{ wins the HomUF-CMA game for } \mathsf{MKHS}(\lambda)] \leq \mathsf{negl}(\lambda).$$

Non-adaptive corruption queries. The HomUF-CMA experiment as defined above allows the adversary to corrupt any identity at any point during the game. A weaker variant of the experiment restricts the adversary to *non-adaptive corruption queries*, where a corruption query for an identity id is non-adaptive if id has not previously appeared in a signing query during the experiment [15].

Furthermore, Fiore *et al.* [15, Proposition 1] show that a MKHS scheme is HomUF-CMA unforgeable against adversaries making no corruption queries if and only if it is HomUF-CMA unforgeable against adversaries making non-adaptive corruption queries. Consequently, when assuming adversaries that make non-adaptive corruption queries, we may analyse security with the total removal of corruption queries instead.

$\mathsf{HomUF\text{-}CMA}_{\mathcal{A},\mathsf{MKHS}}(\lambda)$

1 : $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
2 : $L_{\mathsf{ID}},\ L_\sigma,\ L_{\mathsf{corr}} \leftarrow \varnothing$
3 : $(\mathcal{P}^*, \Delta^*, \{\mathsf{pk_{id}}\}_{\mathsf{id}\in\mathcal{P}^*}, \tilde{m}^*, \tilde{\sigma}^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{ID}}, \mathcal{O}_{\mathsf{corr}}, \mathcal{O}_{\mathsf{Sign}}}(\mathsf{pp})$
4 : **return** $\mathsf{IsForgery}(\mathcal{P}^*, \Delta^*, \{\mathsf{pk_{id}}\}_{\mathsf{id}\in\mathcal{P}^*}, \tilde{m}^*, \tilde{\sigma}^*)$

$\mathcal{O}_{\mathsf{ID}}(\mathsf{id})$

1 : **if** $(\mathsf{id}, \cdot, \cdot, \cdot) \in L_{\mathsf{ID}}$
2 : **return** $\perp$
3 : $(\mathsf{sk_{id}}, \mathsf{pk_{id}}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$
4 : $L_{\mathsf{ID}} \leftarrow L_{\mathsf{ID}} \cup \{(\mathsf{id}, \mathsf{pk_{id}}, \mathsf{sk_{id}}, \mathsf{aux})\}$
5 : **return** $\mathsf{pk_{id}}$

$\mathcal{O}_{\mathsf{corr}}(\mathsf{id})$

1 : **if** $(\mathsf{id}, \cdot, \cdot, \cdot) \notin L_{\mathsf{ID}}$
2 : **return** $\perp$
3 : $L_{\mathsf{corr}} \leftarrow L_{\mathsf{corr}} \cup \{\mathsf{id}\}$
4 : **return** $\mathsf{sk_{id}}$

$\mathcal{O}_{\mathsf{Sign}}(\Delta, \ell, m)$

1 : **parse** $\ell = (\mathsf{id}, \tau)$
2 : **if** $(\Delta, \ell, \cdot, \cdot) \in L_\sigma$
3 : **return** $\perp$
4 : **if** $(\mathsf{id}, \cdot, \cdot, \cdot) \notin L_{\mathsf{ID}}$
5 : **return** $\perp$
6 : $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk_{id}}, \Delta, \ell, m)$
7 : $L_\sigma \leftarrow L_\sigma \cup \{(\Delta, \ell, m, \mathsf{aux})\}$
8 : **return** σ

$\mathsf{IsForgery}(\mathcal{P}, \Delta, \{\mathsf{pk_{id}}\}_{\mathsf{id}\in\mathcal{P}}, \tilde{m}, \tilde{\sigma})$

1 : **parse** $\mathcal{P}^* = (f^*, \ell_1^*, \ldots, \ell_n^*)$
2 : **if** $\mathsf{Verify}(\mathcal{P}, \Delta, \{\mathsf{pk_{id}}\}_{\mathsf{id}\in\mathcal{P}}, \tilde{m}, \tilde{\sigma}) \neq 1$
3 : **return** 0
4 : **if** $(\Delta^*, \cdot, \cdot, \cdot) \notin L_\sigma$
5 : **return** 1 // Type-1 forgery
6 : **if** $\forall\, \mathsf{id} \in \mathcal{P}^*\colon\ \mathsf{id} \notin L_{\mathsf{corr}} \wedge\ \tilde{m}^* \neq f^*(m_1, \ldots, m_n)$
7 : **return** 1 // Type-2 forgery
8 : **if** $\exists\, i \in [n]\colon\ (\Delta^*, \ell_i^*, \cdot, \cdot) \notin L_\sigma\ \wedge\ \mathsf{id}_i \notin L_{\mathsf{corr}}$
9 : **return** 1 // Type-3 forgery
10 : **return** 0

Figure 2.2: The HomUF-CMA security game for a MKHS scheme. The main game (top) runs the adversary with oracle access to $\mathcal{O}_{\mathsf{ID}}$, $\mathcal{O}_{\mathsf{corr}}$, and $\mathcal{O}_{\mathsf{Sign}}$, and decides the outcome via IsForgery.

2.7 The Simplest Multi-Key Linearly Homomorphic Signature Scheme (mklhs) [1]

The mklhs scheme of Aranha and Pagnin [1] is a concrete instantiation of a MKHS scheme supporting linear functions $f(m_1, \ldots, m_n) = \sum_{i=1}^{n} a_i m_i$ over messages signed by multiple independent users. It serves as the starting point for all contributions in this thesis. The full pseudocode for mklhs is given in Figure 2.3.

$\mathsf{Setup}(1^\lambda)$

$\mathcal{G} \leftarrow \mathsf{BilinGroup}_2(\lambda)$
define sets:
  $\mathsf{ID}, \mathcal{T} \subseteq \{0,1\}^*$
  $\mathcal{M} \subseteq \mathbb{Z}_q$
fix hash: $H : \{0,1\}^* \to \mathbb{G}_1$
return $\mathsf{pp} \leftarrow (\mathcal{G}, \mathsf{ID}, \mathcal{M}, \mathcal{T}, H)$

$\mathsf{KeyGen}(\mathsf{pp})$

$\mathsf{id} \xleftarrow{\$} \mathsf{ID}$
$\mathsf{sk}_{\mathsf{id}} \xleftarrow{\$} \mathbb{Z}_q^*$
$\mathsf{pk}_{\mathsf{id}} = g_2^{\mathsf{sk}_{\mathsf{id}}} \in \mathbb{G}_2$
return $(\mathsf{sk}_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}}, \mathsf{id})$

$\mathsf{Sign}(\mathsf{sk}_{\mathsf{id}}, \ell, m)$

$\gamma = (H(\ell) \cdot g_1^m)^{\mathsf{sk}_{\mathsf{id}}}$
$\mu = m$
return $\sigma = (\mathsf{id}, \gamma, \mu)$

$\mathsf{Eval}(\mathcal{P}, \{\sigma_i\}_{i=1}^n)$

parse $\mathcal{P} = (f, \{\ell_i\}_{i=1}^n) \in \mathbb{Z}_q^n \times (\mathsf{ID} \times \mathcal{T})^n$
parse $f = (\{a_i\}_{i=1}^n) \in \mathbb{Z}_q^n$
parse $\sigma_i = (\mathsf{id}_i, \gamma_i, \mu_i) \in \mathsf{ID} \times \mathbb{G}_1 \times \mathbb{Z}_q$
$\tilde{\gamma} = \prod_{i=1}^{n} \gamma_i^{a_i}$
for $\mathsf{id} \in \mathcal{P}$
  $\mathcal{I}_{\mathsf{id}} = \{\, i \in [n] \mid \ell_i = (\mathsf{id}, \cdot) \,\}$
  $\tilde{\mu}_{\mathsf{id}} = \sum_{i \in \mathcal{I}_{\mathsf{id}}} a_i \mu_i$
return $\tilde{\sigma} = (\tilde{\gamma}, \{\tilde{\mu}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}})$

$\mathsf{Verify}(\mathcal{P}, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}}, \tilde{m}, \tilde{\sigma})$

parse $\mathcal{P} = (f, \{\ell_i\}_{i=1}^n) \in \mathbb{Z}_q^n \times (\mathsf{ID} \times \mathcal{T})^n$
parse $f = (\{a_i\}_{i=1}^n) \in \mathbb{Z}_q^n$
parse $\tilde{\sigma} = (\tilde{\gamma}, \{\tilde{\mu}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}}) \in \mathbb{G}_1 \times \mathbb{Z}_q^t$
$\mathsf{ver}_1 \leftarrow \left[\tilde{m} == \sum_{\mathsf{id} \in \mathcal{P}} \tilde{\mu}_{\mathsf{id}}\right]$
for $\mathsf{id} \in \mathcal{P}$
  $\mathcal{I}_{\mathsf{id}} = \{\, i \in [n] \mid \ell_i = (\mathsf{id}, \cdot) \,\}$
$\mathsf{ver}_2 \leftarrow \left[e(\tilde{\gamma}, g_2) == \prod_{\mathsf{id} \in \mathcal{P}} e\Big(g_1^{\tilde{\mu}_{\mathsf{id}}} \cdot \prod_{i \in \mathcal{I}_{\mathsf{id}}} H(\ell_i)^{a_i}, \mathsf{pk}_{\mathsf{id}}\Big)\right]$
return $[\mathsf{ver}_1 \wedge \mathsf{ver}_2]$

Figure 2.3: The mklhs scheme by Aranha and Pagnin [1].

Keys and signing. Each user holds a secret key $\mathsf{sk}_{\mathsf{id}} \in \mathbb{Z}_q^*$ and a corresponding public key $\mathsf{pk}_{\mathsf{id}} = g_2^x \in \mathbb{G}_2$. To sign a message $m \in \mathbb{Z}_q$ under a label ℓ, the user computes $\gamma = (H(\ell) \cdot g_1^m)^{\mathsf{sk}_{\mathsf{id}}}$, binding both the label and the message into a single group element in $\mathbb{G}_1$. This structure allows for the linear evaluation of messages, since multiplying a message m with a scalar a corresponds to raising γ to the power of a.

Evaluation. Given a linear function $f = (a_1, \ldots, a_n)$ with each $a_i \in \mathbb{Z}_q$, and sig-

natures $\sigma_1, \ldots, \sigma_n$ for n individually signed messages, evaluation produces an derived signature $\tilde{\gamma} = \prod_{i=1}^{n} \gamma_i^{a_i}$. Raising γ_i to the power of a_i scales the contribution of message m_i by a_i, so $\tilde{\gamma}$ encodes the linear combination $\sum_i a_i m_i$ as a group element in $\mathbb{G}_1$. Since multiple labels may belong to the same identity, evaluation also tracks the per-identity aggregate $\tilde{\mu}_{\mathsf{id}} = \sum_{i \in \mathcal{I}_{\mathsf{id}}} a_i m_i$, where $\mathcal{I}_{\mathsf{id}} = \{i \in [n] | \ell_i = (\mathsf{id}, \cdot)\}$ collects all input positions associated with id.

Verification. The verifier performs two checks that both need to pass. First, it checks that the claimed output satisfies $\tilde{m} = \sum_{\mathsf{id}} \tilde{\mu}_{\mathsf{id}}$, insisting that $\tilde{m}$ equals the linear combination of the aggregated signed messages of each identity. Second, it uses the bilinear pairing to verify that $\tilde{\gamma}$ was honestly formed. For an honestly evaluated signature, grouping contributions by identity and using bilinearity of e yields

$$e(\tilde{\gamma},\, g_2) = \prod_{\mathsf{id} \in \mathcal{P}} e\left(g_1^{\tilde{\mu}_{\mathsf{id}}} \cdot \prod_{i \in I_{\mathsf{id}}} H(\ell_i)^{a_i},\ \mathsf{pk}_{\mathsf{id}} \right).$$

The right-hand side is computable from public information alone, and matching it against $e(\tilde{\gamma}, g_2)$ confirms that $\tilde{\gamma}$ is consistent with the public keys and the claimed per-identity message values.

Remark 2.31. Following Aranha and Pagnin [1], we include dataset identifiers Δ in the general definitions for completeness. However, mklhs and our constructions do not explicitly use dataset identifiers. Instead, they can be incorporated into the tag space by redefining the tag as $\tau := (\Delta, \tau')$, with τ' being the original tag [15]. This allows the scheme to handle multiple datasets without modifying the underlying construction.

Theorem 2.32 (Security of mklhs [1]). *The* mklhs *scheme is secure in the random oracle model assuming that the co-CDH problem is hard. More precisely, let $\mathcal{A}$ be a PPT adversary in the* HomUF-CMA *security experiment making non-adaptive corruption queries, being restricted such that it may no longer issue signing queries after having queried a random oracle. Then there exists a PPT algorithm $\mathcal{B}$ solving the co-CDH problem such that*

$$\mathsf{Adv}^{\mathsf{HomUF\text{-}CMA}}_{\mathcal{A},\,\mathsf{mklhs}}(\lambda) \leq \frac{1}{2}\left(\frac{1}{|\mathbb{G}_1|} + Q_{\mathsf{id}} \cdot \mathsf{Adv}^{\text{co-CDH}}_{\mathcal{B}}(\lambda) \right),$$

where $Q_{\mathsf{id}} = \mathsf{poly}(\lambda)$ denotes the total number of identities generated during the security experiment.

3

HomUF-CMA With Delayed Hash Queries (HomUF-CMA-DHQ)

In this chapter, we define a modified version of the HomUF-CMA security experiment given in Definition 2.28 that captures the restrictions used in the security proof of the mklhs scheme by Aranha and Pagnin [1] more explicitly. The modified security notion is obtained from HomUF-CMA by eliminating corruption queries and restricting the adversary to issue hash queries only after the signing phase is over.

By formalizing these restrictions directly in the security experiment, we avoid having to restate them in every subsequent security proof. This way, any proof carried out under HomUF-CMA-DHQ automatically inherits the delayed hash query restriction and the absence of corruption queries, letting us focus on the parts of each reduction that are actually specific to the scheme at hand.

3.1 Motivation

Removing corruption queries. The HomUF-CMA experiment allows the adversary to corrupt identities and obtain their corresponding secret keys. We recall that a corruption query for an identity id is *non-adaptive* if id has not been previously submitted as a signing query during the experiment [15]. Importantly, the security proof of mklhs assumes an adversary $\mathcal{A}$ that makes non-adaptive corruption queries.

In this work, we follow the approach of Aranha and Pagnin [1] and remove corruption queries from our security analysis. This decision is justified by the result of Fiore *et al.*, showing that a MKHS is secure against adversaries that do not make corruption queries if and only if it is secure against adversaries that make non-adaptive corruption queries [15, Proposition 1]. Hence, the security guarantees of the schemes we analyse are not affected by the removal of corruption queries.

Delayed hash queries. In addition to removing corruption queries, we also modify the HomUF-CMA experiment to capture a restriction on when the adversary may query hash functions used inside the signing algorithm. We do this to capture the same restriction on the adversary that appears in the security proof of Aranha and Pagnin's mklhs [1]. In their construction, signatures are produced with the help of a hash function. As a result, their security proof is carried out in the random oracle model where the challenger programs the random oracle.

To properly simulate the random oracle, the challenger must ensure consistency between the random values queried by the adversary and those used during signing. Aranha and Pagnin address this issue by restricting the adversary to query the random oracle only after the signing phase is over. In other words, the adversary is restricted to making *delayed hash queries*.

While this restriction is already explicitly stated in the proof of the mklhs scheme of Aranha and Pagnin, it is not reflected in their formulation of the HomUF-CMA experiment. To make this nuance explicit, we define a modified security experiment in which oracle queries for hash functions used in Sign must occur only after the adversary has finished issuing signing queries.

This modification does not change the power of the challenger or the adversary beyond what is already stated in their original proof, but instead helps formalize the limitations of the proof.

Combined with the removal of corruption queries discussed previously, we define the following adjusted security notion that captures the security guarantees of the mklhs construction more explicitly.

3.2 The HomUF-CMA-DHQ Security Game

Definition 3.1 (HomUF-CMA with Delayed Hash Queries – HomUF-CMA-DHQ). The HomUF-CMA-DHQ security experiment is played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, using a MKHS scheme in the random oracle model. Let $\{\mathcal{H}_i\}_{i=1}^{k}$ denote the collection of random oracles that may be queried during the execution of Sign. The delayed-hash restriction applies only to these random oracles, so the adversary may not query them until it has finished issuing signing queries. Any additional random oracles used by the scheme, but not used by Sign, are not subject to this restriction.

The HomUF-CMA-DHQ security game is played in the following stages.

Setup. The challenger $\mathcal{C}$ runs $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ and returns pp to the adversary $\mathcal{A}$. The challenger then initializes the bookkeeping lists

$$L_{\mathsf{ID}} \leftarrow \varnothing, \qquad L_\sigma \leftarrow \varnothing, \qquad L_H \leftarrow \varnothing,$$

to keep track of generated identities, signing queries, and queries to the random oracles used by Sign, respectively. Each list entry may also store auxiliary information, denoted by aux, used for internal bookkeeping by $\mathcal{C}$. Lastly, $\mathcal{C}$ maintains a bit $b_H \leftarrow 0$, which records whether the adversary has queried any random oracle that is subject to the delayed-hash restriction.

Identity query phase. $\mathcal{A}$ can adaptively submit identity queries of the form $\mathsf{id} \in \mathsf{ID}$. Whenever $(\mathsf{id}, \cdot, \cdot, \cdot) \notin L_{\mathsf{ID}}$, $\mathcal{C}$ generates new keys for this identity by running $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, and stores

$$L_{\mathsf{ID}} \leftarrow L_{\mathsf{ID}} \cup (\mathsf{id}, \mathsf{pk}, \mathsf{sk}, \mathsf{aux}).$$

If id already appears in L_{ID}, $\mathcal{C}$ ignores the query. Finally, $\mathcal{C}$ returns pk to $\mathcal{A}$.

Sign query phase. $\mathcal{A}$ can adaptively submit queries of the form $(\Delta, \ell, m, \cdot)$, where Δ is a database identifier, $\ell = (\mathsf{id}, \tau)$ is a label in $\mathsf{ID} \times \mathcal{T}$ and $m \in \mathcal{M}$ is a message. $\mathcal{C}$ ignores the query whenever

- $b_H = 1$, *i.e.*, $\mathcal{A}$ has already queried a random oracle corresponding to a hash in Sign, and signing queries are no longer allowed; or
- $(\Delta, \ell, \cdot, \cdot) \in L_\sigma$, *i.e.*, $\mathcal{A}$ has already asked a signature for label ℓ in the dataset Δ; or
- $(\mathsf{id}, \cdot, \cdot, \cdot) \notin L_{\mathsf{ID}}$, *i.e.*, the identity specified in $\ell = (\mathsf{id}, \tau)$ has not yet been generated.

Otherwise, $\mathcal{C}$ computes $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{id}}, \Delta, \ell, m)$, updates the list

$$L_\sigma \leftarrow L_\sigma \cup (\Delta, \ell, m, \mathsf{aux}),$$

and returns σ to $\mathcal{A}$.

Delayed hash query phase. $\mathcal{A}$ may adaptively query any of the random oracles $\mathcal{H}_i$ used by Sign. After receiving the first such query, $\mathcal{C}$ sets $b_H \leftarrow 1$. For a query of the form (i, x), where $x \in \{0,1\}^*$, $\mathcal{C}$ proceeds as follows: if there exists an entry $(x, \mathcal{H}_i(x), i, \mathsf{aux}) \in L_H$, it returns $\mathcal{H}_i(x)$. Otherwise, it samples/programs the value $\mathcal{H}_i(x)$, updates the list

$$L_H \leftarrow L_H \cup (x, \mathcal{H}_i(x), i, \mathsf{aux}),$$

and returns $\mathcal{H}_i(x)$.

Forgery. At the end of the game, $\mathcal{A}$ outputs a tuple

$$(\mathcal{P}^*, \Delta^*, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}^*}, \tilde{m}^*, \tilde{\sigma}^*).$$

The experiment outputs 1 if the tuple returned by $\mathcal{A}$ is a HomUF-CMA-DHQ forgery (defined below), and 0 otherwise.

Remark 3.2. Unlike HomUF-CMA in Definition 2.28, HomUF-CMA-DHQ does not include a list L_{corr} for corrupted identities, since HomUF-CMA-DHQ uses non-adaptive corruption queries and is therefore equivalent to when corruption queries have been removed. Consequently, the Type-2 and Type-3 forgeries no longer require that the involved identities are non-corrupted.

Definition 3.3 (HomUF-CMA-DHQ Forgery). Assume an execution of HomUF-CMA-DHQ between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, where

$$(\mathcal{P}^*, \Delta^*, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}^*}, \tilde{m}^*, \tilde{\sigma}^*)$$

is the tuple returned by $\mathcal{A}$ and $\mathcal{P}^* = (f^*, \{\ell_i^*\}_{i=1}^n)$. We say that the adversary outputs a HomUF-CMA-DHQ *forgery* if

$$\mathsf{Verify}(\mathcal{P}^*, \Delta^*, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}^*}, \tilde{m}^*, \tilde{\sigma}^*) = 1$$

and at least one of the following holds:

Type-1 forgery. The database Δ^* was never initialized during the game, *i.e.*, $(\Delta^*, \cdot, \cdot, \cdot) \notin L_\sigma$.

Type-2 forgery. $\tilde{\sigma}^*$ is an incorrect evaluation of f^* on previously signed messages, *i.e.*, $(\Delta^*, \ell_i^*, m_i, \cdot) \in L_\sigma$ for all $i \in [n]$, but $\tilde{m}^* \neq f^*(m_1, \ldots, m_n)$.

Type-3 forgery. There exists (at least) one index $i \in [n]$ such that ℓ_i^* was never queried, *i.e.*, $(\Delta^*, \ell_i^*, \cdot, \cdot) \notin L_\sigma$.

Definition 3.4 (HomUF-CMA-DHQ Unforgeable). A MKHS scheme is said to be HomUF-CMA-DHQ *unforgeable* if, for every PPT adversary $\mathcal{A}$, its advantage in winning the HomUF-CMA-DHQ game is negligible in the security parameter of the scheme. Formally,

$$\mathsf{Adv}^{\mathsf{HomUF\text{-}CMA\text{-}DHQ}}_{\mathcal{A},\,\mathsf{MKHS}}(\lambda) = \Pr[\mathcal{A} \text{ wins the HomUF-CMA-DHQ game for } \mathsf{MKHS}(\lambda)] \leq \mathsf{negl}(\lambda).$$

Figure 3.1 illustrates the HomUF-CMA-DHQ game in pseudo-code. The differences compared to HomUF-CMA with non-adaptive corruption queries are in green, showing the additional restrictions in our modified security game.

HomUF-CMA-DHQ$_{\mathcal{A},\mathsf{MKHS}}(\lambda)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$L_{\mathsf{ID}}, L_\sigma, L_H \leftarrow \varnothing$
$b_H \leftarrow 0$
$(\mathcal{P}^*, \Delta^*, \{\mathsf{pk_{id}}\}_{\mathsf{id}\in\mathcal{P}^*}, \tilde{m}^*, \tilde{\sigma}^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{ID}}, \mathcal{O}_{\mathsf{Sign}}, \mathcal{O}_H}(\mathsf{pp})$
return $\mathsf{IsForgery}(\mathcal{P}^*, \Delta^*, \{\mathsf{pk_{id}}\}_{\mathsf{id}\in\mathcal{P}^*}, \tilde{m}^*, \tilde{\sigma}^*)$

$\mathcal{O}_{\mathsf{ID}}(\mathsf{id})$

if $(\mathsf{id}, \cdot, \cdot, \cdot) \in L_{\mathsf{ID}}$
  return $\perp$
$(\mathsf{sk_{id}}, \mathsf{pk_{id}}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$
$L_{\mathsf{ID}} \leftarrow L_{\mathsf{ID}} \cup \{(\mathsf{id}, \mathsf{pk_{id}}, \mathsf{sk_{id}}, \mathsf{aux})\}$
return $\mathsf{pk_{id}}$

$\mathcal{O}_H(i, x)$

if $\mathcal{H}_i$ **used in** MKHS.Sign**:**
  $b_H \leftarrow 1$
if $(x, h, i, \cdot) \in L_H$
  return h
$h \xleftarrow{\$} \mathsf{range}(\mathcal{H}_i)$
$L_H \leftarrow L_H \cup \{(x, h, i, \mathsf{aux})\}$
return h

$\mathcal{O}_{\mathsf{Sign}}(\Delta, \ell, m)$

parse $\ell = (\mathsf{id}, \tau)$
if $b_H = 1$
  return $\perp$
if $(\Delta, \ell, \cdot, \cdot) \in L_\sigma$
  return $\perp$
if $(\mathsf{id}, \cdot, \cdot, \cdot) \notin L_{\mathsf{ID}}$
  return $\perp$
$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk_{id}}, \Delta, \ell, m)$
$L_\sigma \leftarrow L_\sigma \cup \{(\Delta, \ell, m, \mathsf{aux})\}$
return σ

$\mathsf{IsForgery}(\mathcal{P}, \Delta, \{\mathsf{pk_{id}}\}_{\mathsf{id}\in\mathcal{P}}, \tilde{m}, \tilde{\sigma})$

parse $\mathcal{P}^* = (f^*, \ell_1^*, \ldots, \ell_n^*)$
if $\mathsf{Verify}(\mathcal{P}, \Delta, \{\mathsf{pk_{id}}\}_{\mathsf{id}\in\mathcal{P}}, \tilde{m}, \tilde{\sigma}) \neq 1$
  return 0
if $(\Delta^*, \cdot, \cdot, \cdot) \notin L_\sigma$
  return 1 // Type-1 forgery
if $\tilde{m}^* \neq f^*(m_1, \ldots, m_n)$
  return 1 // Type-2 forgery
if $\exists\, i \in [n]:\ (\Delta^*, \ell_i^*, \cdot, \cdot) \notin L_\sigma$
  return 1 // Type-3 forgery
return 0

Figure 3.1: The HomUF-CMA-DHQ security game for a MKHS scheme. The main game (top) runs the adversary with oracle access to $\mathcal{O}_{\mathsf{ID}}$, $\mathcal{O}_{\mathsf{Sign}}$, and $\mathcal{O}_H$, and decides the outcome of the game via the IsForgery algorithm. The flag b_H enforces the delayed-hash restriction. Green text indicates changes in HomUF-CMA-DHQ compared to the standard HomUF-CMA security experiment with non-adaptive/removed corruption queries, see Figure 2.2.

4

Security of mklhs in the Type 3 Pairing Setting

This chapter provides a new security proof for mklhs in the Type 3 pairing setting. The construction itself is unchanged from Figure 2.3, with the single exception that bilinear groups are now generated by $\mathsf{BilinGroup}_3$ instead of $\mathsf{BilinGroup}_2$ to account for the new Type 3 pairing setting, as seen in Figure 4.1. We refer to this Type 3 instantiation as mklhs_3 to distinguish it from the original. Since no efficiently computable homomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$ is assumed to exist in this setting, the original proof of Aranha and Pagnin [1] no longer applies.

$\mathsf{Setup}(1^\lambda)$

1 : $\mathcal{G} \leftarrow \mathsf{BilinGroup}_3(\lambda)$
2 : **define sets:**
3 : $\quad \mathsf{ID}, \mathcal{T} \subseteq \{0,1\}^*$
4 : $\quad \mathcal{M} \subseteq \mathbb{Z}_q$
5 : **fix hash:**
6 : $\quad H : \{0,1\}^* \to \mathbb{G}_1$
7 : $\mathsf{pp} \leftarrow (\mathcal{G}, \mathsf{ID}, \mathcal{M}, \mathcal{T}, H)$
8 : **return** pp

Figure 4.1: The Setup algorithm of mklhs_3. Green text indicates changes from mklhs in Figure 2.3.

Following the recipe of Chatterjee and Menezes [4], our contribution is a new security proof under the co-CDH* assumption (Assumption 2.21). The proof is formulated in the HomUF-CMA-DHQ security game from Chapter 3 and achieves an overall better advantage bound than the original proof by Aranha and Pagnin [1].

Furthermore, we explicitly restrict the admissible linear functions $f(m_1, \ldots, m_n) = \sum_{i=1}^{n} a_i m_i$ to those satisfying $a_i \neq 0$ for all $i \in [n]$. This does not reduce expressiveness, since a zero coefficient means its corresponding input has no effect on f.

4.1 Security Theorem and Proof Outline

Theorem 4.1. *Assuming that the co-CDH* problem is hard, the* mklhs_3 *scheme is secure under* HomUF-CMA-DHQ *in the random oracle model. Formally, let* $\mathcal{A}$ *be a PPT adversary in the* HomUF-CMA-DHQ *security experiment. Then its advantage is bounded by*

$$\mathsf{Adv}^{\mathsf{HomUF\text{-}CMA\text{-}DHQ}}_{\mathcal{A},\,\mathsf{mklhs}_3}(\lambda) \leq 3 \cdot \mathsf{Adv}^{\text{co-CDH}^*}_{\mathcal{B}}(\lambda),$$

where $\mathcal{B}$ *is a PPT algorithm that solves the co-CDH* problem with advantage* $\mathsf{Adv}^{\text{co-CDH}^*}_{\mathcal{B}}(\lambda)$.

Remark 4.2. Compared with the original security bound for mklhs presented in Theorem 2.32, the advantage of $\mathcal{A}$ is tighter in our security proof. In particular, it no longer depends on the number of queried identities Q_{id} nor on the additive term $1/|\mathbb{G}_1|$, and instead scales directly with the hardness of co-CDH* up to a constant factor.

Proof Outline. The goal of this proof is to bound the adversary's advantage in terms of solving the co-CDH* assumption. This is done by first constructing a reduction $\mathcal{B}$ that simulates the HomUF-CMA-DHQ security experiment against an adversary $\mathcal{A}$ while embedding the co-CDH* challenge into its outputs. The interaction is sketched in Figure 4.2. The correctness of the simulation is argued in Section 4.3. After receiving a forgery from $\mathcal{A}$, we analyse each of the three forgery types from Definition 3.3 separately, and combine their contribution to bound the total advantage.

Type-1 forgeries require the adversary to use a new dataset identifier Δ^*. Because identifiers are bound to labels as shown in Remark 2.31, any query with Δ^* results in an unqueried label. This immediately reduces the adversary to a Type-3 forgery, resulting in the same forgery success probability.

Type-2 forgeries require the adversary to produce a valid signature on an incorrect function evaluation, despite querying all labels honestly. We construct a reduction $\mathcal{B}$ that simulates a real execution of the scheme against an adversary $\mathcal{A}$. Simultaneously, $\mathcal{B}$ embeds the co-CDH* challenge into the public parameters and keys. A Type-2 forgery guarantees that $\mathcal{A}$'s claimed result disagrees with the honestly computed one for at least one identity. This per-identity mismatch leaks the structure $\mathcal{B}$ needs to extract a solution to the co-CDH* challenge.

Type-3 forgeries require the adversary to produce a valid signature for a label it never queried. Using the same simulation as for Type-2 forgeries, the co-CDH* challenge is embedded into the hash of the unsigned labels, and a similar extraction procedure then applies.

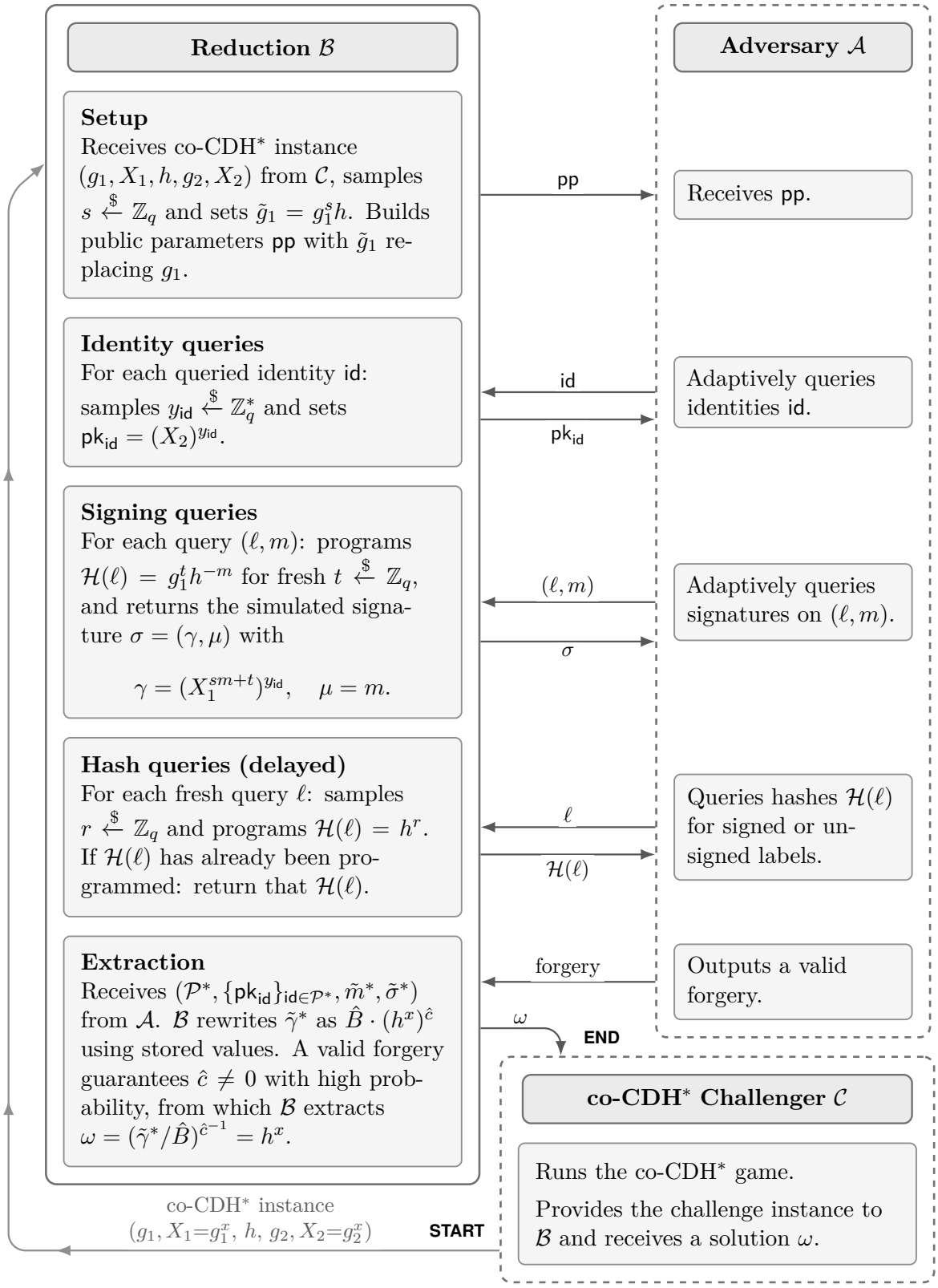


Figure 4.2: Sketch of the reduction for Theorem 4.1. The reduction $\mathcal{B}$ simulates the scheme for the adversary $\mathcal{A}$, embedding the co-CDH* challenge in its outputs Signing queries program $\mathcal{H}(\ell) = g_1^t h^{-m}$, while delayed hash queries program $\mathcal{H}(\ell) = h^r$. Any valid Type-1, Type-2, or Type-3 forgery allows $\mathcal{B}$ to extract h^x.

4.2 Proving Security

Proof of Theorem 4.1. We distinguish between the three types of forgeries from Definition 3.3.

4.2.1 Handling Dataset Identifiers and Type-1 Forgeries

We show that a Type-1 forgery, with dataset identifiers incorporated into the tag space as in Remark 2.31, reduces to a Type-3 forgery.

We recall that in a Type-1 forgery, $\mathcal{A}$ outputs a tuple $(\mathcal{P}^*, \Delta^*, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}^*}, \tilde{m}^*, \tilde{\sigma}^*)$ such that Δ^* was never used in any signing query, *i.e.*, $(\Delta^*, \cdot, \cdot) \notin L_\sigma$, but the verification equation still holds. We also recall that a forgery is of Type-3 if there exists at least one index $i \in [n]$ such that $(\Delta^*, \ell_i^*, \cdot, \cdot) \notin L_\sigma$.

We modify the scheme by incorporating the dataset identifier Δ into the tag space. Concretely, we redefine each label as

$$\ell = (\mathsf{id}, \tau) \quad \text{with} \quad \tau := (\Delta, \tau'), \qquad \textit{i.e.}, \qquad \ell = (\mathsf{id}, (\Delta, \tau')).$$

Next, we consider a Type-1 forgery. Since Δ^* was never used in any signing query, it follows that for every label $\ell_i^* = (\mathsf{id}_i, \tau_i^*)$ with $\tau_i^* = (\Delta^*, \tau_i')$, we have that ℓ_i^* was never queried during the signing phase, *i.e.* $(\Delta^*, \ell_i^*, \cdot, \cdot) \notin L_\sigma$ for all $i \in [n]$. In particular, there exists an index $i \in [n]$ such that this holds, which constitutes a Type-3 forgery.

Hence, any Type-1 forgery also results in a Type-3 forgery in the modified scheme. Therefore,

$$\Pr[\mathcal{A} \text{ outputs a Type-1 forgery}] \leq \Pr[\mathcal{A} \text{ outputs a Type-3 forgery}].$$

4.2.2 Simulating HomUF-CMA-DHQ

Consider the case of an adversary $\mathcal{A}$ that can create Type-2 or Type-3 forgeries against mklhs_3. We define a reduction $\mathcal{B}$ that turns such forgeries into solutions for any given co-CDH* instance.

As per the co-CDH* game in Assumption 2.21, $\mathcal{B}$ receives as input from the challenger:

- a bilinear group $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, q, e)$,
- generators $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$, and
- elements $h, X_1 = g_1^x \in \mathbb{G}_1, X_2 = g_2^x \in \mathbb{G}_2$.

The goal of the reduction is for $\mathcal{B}$ to output $\omega \in \mathbb{G}_1$ such that $\omega = h^x$.

$\mathcal{B}$ simulates mklhs_3 with $\mathcal{A}$ as adversary in the following way.

Setup. $\mathcal{B}$ samples $s \xleftarrow{\$} \mathbb{Z}_q$ and sets

$$\tilde{g}_1 := g_1^s h,$$

then outputs public parameters $\tilde{\mathcal{G}} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \tilde{g}_1, g_2, q, e)$. $\mathcal{B}$ also initializes the bookkeeping lists $L_{\mathsf{ID}}, L_H, L_\sigma \leftarrow \varnothing$, and the bit $b_H \leftarrow 0$ to keep track of if $\mathcal{A}$ has queried a random oracle used in $\mathsf{mklhs}_3.\mathsf{Sign}$ from Figure 5.1, and is no longer allowed to make signing queries.

Identity query phase. On input query of the form $\mathsf{id} \in \mathsf{ID}$, if id has previously been generated, *i.e.* $(\mathsf{id}, \cdot, \cdot, \cdot) \in L_{\mathsf{id}}$, $\mathcal{B}$ ignores the query. Otherwise, $\mathcal{B}$ samples $y_{\mathsf{ID}} \xleftarrow{\$} \mathbb{Z}_q^*$ and sets $\mathsf{pk}_{\mathsf{id}} = (X_2)^{y_{\mathsf{id}}}$.

Since $X_2 = g_2^x$ we have that $\mathsf{pk}_{\mathsf{id}} = g_2^{x y_{\mathsf{id}}}$. This corresponds to the secret key being $\mathsf{sk}_{\mathsf{id}} = x y_{\mathsf{id}}$, which is unknown to $\mathcal{B}$.

Lastly, $\mathcal{B}$ stores

$$L_{\mathsf{ID}} \leftarrow L_{\mathsf{ID}} \cup \{(\mathsf{id}, \cdot, \mathsf{pk}_{\mathsf{id}}, y_{\mathsf{id}})\},$$

and returns $\mathsf{pk}_{\mathsf{id}}$ to $\mathcal{A}$.

Sign query phase. On input query of the form (ℓ, m), where $\ell = (\mathsf{id}, \tau)$, $\mathcal{B}$ ignores the query if

- the label has already been signed before, *i.e.*, $(\cdot, \ell, \cdot, \cdot) \in L_\sigma$; or
- id has not been generated in the previous phase, *i.e.*, $(\mathsf{id}, \cdot, \cdot, \cdot) \notin L_{\mathsf{ID}}$; or
- the adversary has already made a hash query and signing queries are no longer allowed, *i.e.*, $b_H = 1$.

Otherwise, $\mathcal{B}$ samples $t \xleftarrow{\$} \mathbb{Z}_q$ and programs the random oracle as

$$\mathcal{H}(\ell) = g_1^t h^{-m}, \tag{4.1}$$

then stores $L_H \leftarrow L_H \cup \{(\ell, \mathcal{H}(\ell), \cdot, t)\}$.

Next, $\mathcal{B}$ retrieves id and y_{id} from $(\mathsf{id}, \cdot, \cdot, y_{\mathsf{id}}) \in L_{\mathsf{ID}}$, sets

$$\gamma := \left(X_1^{ms+t}\right)^{y_{\mathsf{id}}}, \tag{4.2}$$

and forms $\sigma = (\mathsf{id}, \gamma, \mu)$ with $\mu = m$.

Finally, $\mathcal{B}$ stores $L_\sigma \leftarrow L_\sigma \cup \{(\cdot, \ell, m, \cdot)\}$ and returns σ.

Delayed hash query phase. On the first hash query for a hash present in $\mathsf{mklhs}_3.\mathsf{Sign}$, $\mathcal{B}$ sets $b_h \leftarrow 1$ marking that sign queries are no longer allowed.

On input query ℓ, if $(\ell, \mathcal{H}(\ell), \cdot, \cdot) \in L_H$, $\mathcal{B}$ returns $\mathcal{H}(\ell)$. Otherwise, $\mathcal{B}$ samples $r \xleftarrow{\$} \mathbb{Z}_q$, sets $\mathcal{H}(\ell) = h^r$, stores $L_H \leftarrow L_H \cup \{(\ell, \mathcal{H}(\ell), \cdot, r)\}$ and returns $\mathcal{H}(\ell)$.

Claim 4.3. *The view of $\mathcal{A}$ in its interaction with $\mathcal{B}$ is identically distributed to its view in the real* HomUF-CMA-DHQ *experiment with the scheme* mklhs_3.

We defer the proof of Claim 4.3 to the end of this chapter.

Next, we show how the reduction $\mathcal{B}$ may extract the co-CDH* solution h^x from successful Type-2 and Type-3 forgeries.

4.2.3 co-CDH* Extraction From Type-2 Forgeries

Let $\mathcal{A}$, after its interaction with $\mathcal{B}$, output a successful Type-2 forgery

$$(\mathcal{P}^*, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}^*}, \tilde{m}^*, \tilde{\sigma}^*), \text{ where } \mathcal{P}^* = (f^*, \ell_1^*, \ldots, \ell_n^*).$$

By definition of a Type-2 forgery, we have

$$\mathsf{Verify}(\mathcal{P}^*, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}^*}, \tilde{m}^*, \tilde{\sigma}^*) = 1 \quad \text{and} \quad \tilde{m}^* \neq f^*(m_1, \ldots, m_n),$$

where m_i is the message that $\mathcal{A}$ queried during the sign query phase for label ℓ_i^*. More precisely, the Type-2 forgery condition guarantees that for every index $i \in [n]$, there exists a sign query of the form (ℓ_i^*, m_i) issued by $\mathcal{A}$ during the game, and therefore the tuple $(\cdot, \ell_i^*, m_i, \cdot) \in L_\sigma$ was recorded by $\mathcal{B}$. Consequently, for every label ℓ_i^* appearing in the program $\mathcal{P}^*$, $\mathcal{H}(\ell_i^*)$ was programmed during the sign query phase and *not* the hash query phase. Hence, the corresponding randomness of each $\mathcal{H}(\ell_i^*)$ comes only from t_i in (4.1).

Identifying inconsistent identities. Next, we let the index set for each $\mathsf{id} \in \mathcal{P}^*$ be $\mathcal{I}_{\mathsf{id}} = \{\, i \in [n] \mid \ell_i^* = (\mathsf{id}, \cdot) \,\}$. We also define the *mismatch term* for each id as

$$\delta_{\mathsf{id}} := \tilde{\mu}_{\mathsf{id}}^* - \sum_{i\in\mathcal{I}_{\mathsf{id}}} a_i^* m_i. \tag{4.3}$$

We note that each mismatch term δ_{id} is computable by $\mathcal{B}$, since for every $i \in \mathcal{I}_{\mathsf{id}}$ the tuple $(\cdot, \ell_i^*, m_i, \cdot)$ was recorded in L_σ by the Type-2 forgery condition, and the coefficients a_i^* and values $\tilde{\mu}_{\mathsf{id}}^*$ are part of the forgery.

Using the mismatch terms, we also define the set of *consistent* identities as

$$\mathcal{C} := \{\mathsf{id} \in \mathcal{P}^* \mid \delta_{\mathsf{id}} = 0\}.$$

Since $\tilde{m}^* = \sum_{\mathsf{id}\in\mathcal{P}^*} \tilde{\mu}_{\mathsf{id}}^*$ by line 4 in $\mathsf{mklhs}_3.\mathsf{Verify}$, while $\tilde{m}^* \neq \sum_{i=1}^n a_i^* m_i$ by the Type-2 forgery condition, it follows that

$$\sum_{\mathsf{id}\in\mathcal{P}^*} \tilde{\mu}_{\mathsf{id}}^* \neq \sum_{\mathsf{id}\in\mathcal{P}^*} \sum_{i\in\mathcal{I}_{\mathsf{id}}} a_i^* m_i.$$

Therefore, there exists at least one identity id such that $\tilde{\mu}_{\mathsf{id}}^* \neq \sum_{i\in\mathcal{I}_{\mathsf{id}}} a_i^* m_i$ and $\mathsf{id} \notin \mathcal{C}$ for that identity. Hence, the set of *inconsistent* identities $\{\mathsf{id} \in \mathcal{P}^* \mid \mathsf{id} \notin \mathcal{C}\}$ is not empty.

Finding an expression for $\tilde{\gamma}^*$. Next, we proceed with the extraction of the co-CDH* solution.

$\mathcal{B}$ parses $\tilde{\sigma}^* = (\tilde{\gamma}^*, \{\tilde{\mu}^*_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}^*})$. For each identity $\mathsf{id} \in \mathcal{P}^*$, we define the placeholder variable

$$\hat{A}_{\mathsf{id}} := \tilde{g}_1^{\tilde{\mu}^*_{\mathsf{id}}} \cdot \prod_{i\in\mathcal{I}_{\mathsf{id}}} \mathcal{H}(\ell_i^*)^{a_i^*}. \tag{4.4}$$

From the verification equation on line 7 in $\mathsf{mklhs}_3.\mathsf{Verify}$ and using that $\mathsf{pk}_{\mathsf{id}} = X_2^{y_{\mathsf{id}}}$, we have

$$e(\tilde{\gamma}^*, g_2) = \prod_{\mathsf{id}\in\mathcal{P}^*} e(\hat{A}_{\mathsf{id}}, X_2^{y_{\mathsf{id}}}).$$

Using bilinearity and $X_2 = g_2^x$, this becomes

$$\begin{aligned} e(\tilde{\gamma}^*, g_2) &= \prod_{\mathsf{id}\in\mathcal{P}^*} e(\hat{A}_{\mathsf{id}}, g_2^{xy_{\mathsf{id}}}) \\ &= \prod_{\mathsf{id}\in\mathcal{P}^*} e(\hat{A}_{\mathsf{id}}^{xy_{\mathsf{id}}}, g_2) \\ &= e\left(\prod_{\mathsf{id}\in\mathcal{P}^*} \hat{A}_{\mathsf{id}}^{xy_{\mathsf{id}}}, g_2\right), \end{aligned}$$

and by non-degeneracy of the pairing, we finally obtain

$$\tilde{\gamma}^* = \prod_{\mathsf{id}\in\mathcal{P}^*} \hat{A}_{\mathsf{id}}^{xy_{\mathsf{id}}}. \tag{4.5}$$

Substituting the programmed hashes. Next, we expand each $\hat{A}_{\mathsf{id}}$ from (4.4) using $\tilde{g}_1 = g_1^s h$ and the programmed hash values $\mathcal{H}(\ell) = g_1^t h^{-m}$:

$$\begin{aligned} \hat{A}_{\mathsf{id}} &= (g_1^s h)^{\tilde{\mu}^*_{\mathsf{id}}} \prod_{i\in\mathcal{I}_{\mathsf{id}}} (g_1^{t_i} h^{-m_i})^{a_i^*} \\ &= g_1^{s\tilde{\mu}^*_{\mathsf{id}}} h^{\tilde{\mu}^*_{\mathsf{id}}} \prod_{i\in\mathcal{I}_{\mathsf{id}}} g_1^{a_i^* t_i} h^{-a_i^* m_i} \\ &= g_1^{s\tilde{\mu}^*_{\mathsf{id}} + \sum_{i\in\mathcal{I}_{\mathsf{id}}} a_i^* t_i} \cdot h^{\tilde{\mu}^*_{\mathsf{id}} - \sum_{i\in\mathcal{I}_{\mathsf{id}}} a_i^* m_i}. \end{aligned}$$

Inspecting the exponents, we observe that the exponent of h is precisely δ_{id} as defined in (4.3). We also define the corresponding exponent of g_1 for identity id as

$$\hat{a}_{\mathsf{id}} := s\tilde{\mu}^*_{\mathsf{id}} + \sum_{i\in\mathcal{I}_{\mathsf{id}}} a_i^* t_i,$$

computable by $\mathcal{B}$ from stored values using s generated during the setup phase and t_i from $(\ell_i^*, \cdot, \cdot, t_i) \in L_H$. Hence,

$$\hat{A}_{\mathsf{id}} = g_1^{\hat{a}_{\mathsf{id}}} \cdot h^{\delta_{\mathsf{id}}}.$$

Extracting the co-CDH* solution. For identities $\mathsf{id} \in \mathcal{C}$, we have $\delta_{\mathsf{id}} = 0$ and the h-term becomes $1_{\mathbb{G}_1}$. Splitting (4.5) into consistent and inconsistent identities with the above $\hat{A}_{\mathsf{id}}$ expression yields

$$\begin{aligned}\tilde{\gamma}^* &= \prod_{\mathsf{id}\in\mathcal{C}} \left(g_1^{\hat{a}_{\mathsf{id}}}\right)^{xy_{\mathsf{id}}} \cdot \prod_{\mathsf{id}\in(\mathcal{P}^*\setminus\mathcal{C})} \left(g_1^{\hat{a}_{\mathsf{id}}} h^{\delta_{\mathsf{id}}}\right)^{xy_{\mathsf{id}}} \\ &= \prod_{\mathsf{id}\in\mathcal{P}^*} \left(g_1^{\hat{a}_{\mathsf{id}}}\right)^{xy_{\mathsf{id}}} \cdot \prod_{\mathsf{id}\in(\mathcal{P}^*\setminus\mathcal{C})} \left(h^{\delta_{\mathsf{id}}}\right)^{xy_{\mathsf{id}}} \\ &= \prod_{\mathsf{id}\in\mathcal{P}^*} X_1^{\hat{a}_{\mathsf{id}} y_{\mathsf{id}}} \cdot (h^x)^{\sum_{\mathsf{id}\in(\mathcal{P}^*\setminus\mathcal{C})} \delta_{\mathsf{id}} y_{\mathsf{id}}}.\end{aligned}$$

Continuing, we define the values

$$\hat{B} := \prod_{\mathsf{id}\in\mathcal{P}^*} X_1^{y_{\mathsf{id}}\hat{a}_{\mathsf{id}}},$$

and

$$\hat{c} := \sum_{\mathsf{id}\in\mathcal{P}^*} \delta_{\mathsf{id}} y_{\mathsf{id}}, \tag{4.6}$$

which are both computable by $\mathcal{B}$ from values y_{id} stored in L_{ID} during the identity query phase. Since, $\delta_{\mathsf{id}} = 0$ for all $\mathsf{id} \in \mathcal{C}$, we have $\hat{c} = \sum_{\mathsf{id}\in(\mathcal{P}^*\setminus\mathcal{C})} \delta_{\mathsf{id}} y_{\mathsf{id}}$.

From the expression for $\tilde{\gamma}^*$ derived above, we obtain $\tilde{\gamma}^* = \hat{B} \cdot (h^x)^{\hat{c}}$. Hence,

$$\frac{\tilde{\gamma}^*}{\hat{B}} = (h^x)^{\hat{c}}.$$

If $\hat{c} = 0$ the reduction aborts. Otherwise, since $\hat{c} \neq 0$, $\mathcal{B}$ computes and outputs

$$\omega = \left(\frac{\tilde{\gamma}^*}{\hat{B}}\right)^{\hat{c}^{-1}} = h^x,$$

which solves the co-CDH* instance.

Bounding the abort probability. It remains to bound the probability of abort, *i.e.*, the probability that $\hat{c} = 0$. Since there exists at least one identity $\mathsf{id} \notin \mathcal{C}$ with $\delta_{\mathsf{id}} \neq 0$, (4.6) is a non-zero linear polynomial in the random variables $y_{\mathsf{id}} \xleftarrow{\$} \mathbb{Z}_q^*$ with $|\mathbb{Z}_q^*| = q - 1$. By the Schwarz–Zippel lemma, $\hat{c}$ evaluates to zero with probability at most $1/(q-1)$.

Concluding, our reduction transforms Type-2 forgeries into a solution to the co-CDH* problem unless it aborts, which gives us

$$\mathsf{Adv}_{\mathcal{B}}^{\text{co-CDH}^*}(\lambda) \;\geq\; \left(1 - \frac{1}{q-1}\right) \cdot \Pr[\mathcal{A} \text{ outputs a valid Type-2 forgery}].$$

4.2.4 co-CDH* Extraction From Type-3 Forgeries

Let $\mathcal{A}$, after its interaction with $\mathcal{B}$, output a successful Type-3 forgery

$$(\mathcal{P}^*, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}^*}, \tilde{m}^*, \tilde{\sigma}^*), \text{ where } \mathcal{P}^* = (f^*, \ell_1^*, \ldots, \ell_n^*).$$

We partition the labels in $\mathcal{P}^*$ into those that were signed during the game and those that were not. Concretely, we define

$$\mathcal{U} = \{\, i \in [n] \mid (\cdot, \ell_i^*, \cdot, \cdot) \notin L_\sigma \},$$

i.e., the set of indices corresponding to *unsigned* labels in $\mathcal{P}^*$. In particular, $\mathcal{U} \neq \varnothing$ since we are in the Type-3 forgery setting.

For each $\mathsf{id} \in \mathcal{P}^*$, we also let $\mathcal{I}_{\mathsf{id}} = \{\, i \in [n] \mid \ell_i^* = (\mathsf{id}, \cdot) \}$ and $\mathcal{U}_{\mathsf{id}} = \mathcal{I}_{\mathsf{id}} \cap \mathcal{U}$. Hence, $\mathcal{U}_{\mathsf{id}}$ is the set of indices of unsigned labels for identity id.

Exactly as in the proof of the Type-2 forgery, we define for each $\mathsf{id} \in \mathcal{P}^*$ the intermediary value

$$\hat{A}_{\mathsf{id}} := \tilde{g}_1^{\tilde{\mu}_{\mathsf{id}}^*} \cdot \prod_{i\in I_{\mathsf{id}}} H(\ell_i^*)^{a_i^*},$$

and using ver_2 from Figure 2.3 and $\mathsf{pk}_{\mathsf{id}} = X_2^{y_{\mathsf{id}}} = g_2^{xy_{\mathsf{id}}}$, we obtain

$$\tilde{\gamma}^* = \prod_{\mathsf{id}\in\mathcal{P}^*} \hat{A}_{\mathsf{id}}^{xy_{\mathsf{id}}}.$$

Substituting the programmed hashes. We note that for all indices $i \in \mathcal{I}_{\mathsf{id}} \setminus \mathcal{U}_{\mathsf{id}}$, the hash values of ℓ_i were programmed during the signing phase as $\mathcal{H}(\ell_i^*) = g_1^{t_i} h^{-m_i}$, where m_i is the unique signed message associated with ℓ_i^*. For indices $i \in \mathcal{U}_{\mathsf{id}}$, the hash values were instead programmed in the delayed hash phase, or when sampled for verification, as $\mathcal{H}(\ell_i^*) = h^{r_i}$.

Substituting these hashes and $\tilde{g}_1 = g_1^s h$ into $\hat{A}_{\mathsf{id}}$ gives

$$\begin{aligned}\hat{A}_{\mathsf{id}} &= (g_1^s h)^{\tilde{\mu}_{\mathsf{id}}^*} \cdot \prod_{i\in\mathcal{I}_{\mathsf{id}}\setminus\mathcal{U}_{\mathsf{id}}} (g_1^{t_i} h^{-m_i})^{a_i^*} \cdot \prod_{i\in\mathcal{U}_{\mathsf{id}}} (h^{r_i})^{a_i^*} \\ &= g_1^{s\tilde{\mu}_{\mathsf{id}}^* + \sum_{i\in\mathcal{I}_{\mathsf{id}}\setminus\mathcal{U}_{\mathsf{id}}} a_i^* t_i} \cdot h^{\tilde{\mu}_{\mathsf{id}}^* - \sum_{i\in\mathcal{I}_{\mathsf{id}}\setminus\mathcal{U}_{\mathsf{id}}} a_i^* m_i + \sum_{i\in\mathcal{U}_{\mathsf{id}}} a_i^* r_i}.\end{aligned}$$

We therefore define the per-identity exponents

$$\begin{aligned}\hat{a}_{\mathsf{id}} &:= s\tilde{\mu}_{\mathsf{id}}^* + \sum_{i\in\mathcal{I}_{\mathsf{id}}\setminus\mathcal{U}_{\mathsf{id}}} a_i^* t_i, \\ \hat{b}_{\mathsf{id}} &:= \tilde{\mu}_{\mathsf{id}}^* - \sum_{i\in\mathcal{I}_{\mathsf{id}}\setminus\mathcal{U}_{\mathsf{id}}} a_i^* m_i + \sum_{i\in\mathcal{U}_{\mathsf{id}}} a_i^* r_i,\end{aligned} \tag{4.7}$$

giving $\hat{A}_{\mathsf{id}} = g_1^{\hat{a}_{\mathsf{id}}} h^{\hat{b}_{\mathsf{id}}}$. We note that both exponents are computable by $\mathcal{B}$, since s is sampled during setup, $\tilde{\mu}_{\mathsf{id}}^*$ and a_i^* come from the forgery, and t_i, r_i are sampled by $\mathcal{B}$ during the sign and delayed hash query phase respectively.

Extracting the co-CDH* solution. Substituting our new $\hat{A}_{\mathsf{id}}$ value into the expression for $\tilde{\gamma}^*$ gives

$$\begin{aligned}\tilde{\gamma}^* &= \prod_{\mathsf{id}\in\mathcal{P}^*} \left(g_1^{\hat{a}_{\mathsf{id}}} h^{\hat{b}_{\mathsf{id}}}\right)^{x y_{\mathsf{id}}} \\ &= \prod_{\mathsf{id}\in\mathcal{P}^*} X_1^{y_{\mathsf{id}}\hat{a}_{\mathsf{id}}} \cdot (h^x)^{\sum_{\mathsf{id}\in\mathcal{P}^*} y_{\mathsf{id}}\hat{b}_{\mathsf{id}}}.\end{aligned}$$

Lastly, we define

$$\hat{B} := \prod_{\mathsf{id}\in\mathcal{P}^*} X_1^{y_{\mathsf{id}}\hat{a}_{\mathsf{id}}} \quad \text{and} \quad \hat{c} := \sum_{\mathsf{id}\in\mathcal{P}^*} y_{\mathsf{id}}\hat{b}_{\mathsf{id}}, \tag{4.8}$$

such that

$$\tilde{\gamma}^* = \hat{B} \cdot (h^x)^{\hat{c}}.$$

If $\hat{c} = 0$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ outputs

$$\omega = \left(\frac{\tilde{\gamma}^*}{\hat{B}}\right)^{\hat{c}^{-1}} = h^x,$$

which solves the co-CDH* instance.

Bounding the abort probability. Expanding $\hat{c}$ from (4.8) using (4.7), we see it is a polynomial in the random variables $(r_i)_{i\in\mathcal{U}} \xleftarrow{\$} \mathbb{Z}_q^{|\mathcal{U}|}$ and $(y_{\mathsf{id}})_{\mathsf{id}\in\mathcal{P}^*} \xleftarrow{\$} (\mathbb{Z}_q^*)^{|\mathsf{id}\in\mathcal{P}^*|}$, where $\mathcal{U}$ is the set of indices corresponding to unsigned labels:

$$\hat{c} = \sum_{\mathsf{id}\in\mathcal{P}^*} y_{\mathsf{id}} \left(\tilde{\mu}^*_{\mathsf{id}} - \sum_{i\in\mathcal{I}_{\mathsf{id}}\setminus\mathcal{U}_{\mathsf{id}}} a_i^* m_i + \sum_{i\in\mathcal{U}_{\mathsf{id}}} a_i^* r_i\right).$$

We first *condition* on $(y_{\mathsf{id}})_{\mathsf{id}\in\mathcal{P}^*}$, *i.e.*, we assume the values $(y_{\mathsf{id}})_{\mathsf{id}\in\mathcal{P}^*}$ have been fixed to some arbitrary outcome $\mathbf{y} \in (\mathbb{Z}_q^*)^{|\mathsf{id}\in\mathcal{P}^*|}$, and study $\hat{c}$ as a polynomial in the remaining random variables $(r_i)_{i\in\mathcal{U}}$.

Viewed this way, $\hat{c}$ has degree 1 and is not identically zero, since for each $i \in \mathcal{U}$ with $\ell_i = (\mathsf{id}, \cdot)$, the coefficient of r_i is $y_{\mathsf{id}} a_i^* \neq 0$ by $y_{\mathsf{id}} \in \mathbb{Z}_q^*$ and $a_i^* \neq 0$. Hence, the Schwartz–Zippel lemma yields

$$\Pr[\hat{c} = 0 \mid (y_{\mathsf{id}})_{\mathsf{id}\in\mathcal{P}^*} = \mathbf{y}] \leq \frac{1}{q}.$$

Crucially, since $\mathbf{y}$ was arbitrary, this bound is the same for every outcome $\mathbf{y} \in (\mathbb{Z}_q^*)^{|\mathsf{id}\in\mathcal{P}^*|}$. Therefore, by the law of total probability (Theorem 2.17),

$$\begin{aligned}\Pr[\hat{c} = 0] &= \sum_{\mathbf{y}\in(\mathbb{Z}_q^*)^{|\mathsf{id}\in\mathcal{P}^*|}} \Pr[(y_{\mathsf{id}})_{\mathsf{id}\in\mathcal{P}^*} = \mathbf{y}] \cdot \Pr\left[\hat{c} = 0 \;\middle|\; (y_{\mathsf{id}})_{\mathsf{id}\in\mathcal{P}^*} = \mathbf{y}\right] \\ &\leq \sum_{\mathbf{y}\in(\mathbb{Z}_q^*)^{|\mathsf{id}\in\mathcal{P}^*|}} \Pr[(y_{\mathsf{id}})_{\mathsf{id}\in\mathcal{P}^*} = \mathbf{y}] \cdot \frac{1}{q} = \frac{1}{q}.\end{aligned}$$

Consequently,

$$\mathsf{Adv}^{\text{co-CDH}^*}_{\mathcal{B}}(\lambda) \;\geq\; \left(1 - \frac{1}{q}\right) \cdot \Pr[\mathcal{A} \text{ outputs a valid Type-3 forgery}].$$

4.2.5 Combining the Bounds

Finally, we can combine the bounds from the previous subsections to a single expression bounding the total advantage $\mathsf{Adv}^{\mathsf{HomUF\text{-}CMA\text{-}DHQ}}_{\mathcal{A},\,\mathsf{mklhs}_3}(\lambda)$.

We note that once dataset identifiers are incorporated into the tag space (Remark 2.31), any Type-1 forgery also constitutes a Type-3 forgery. Hence, all Type-1 forgeries constitute a subset of all Type-3 forgeries and we disregard Type-1 forgeries in the advantage bounds. The advantage of the adversary therefore satisfies

$$\mathsf{Adv}^{\mathsf{HomUF\text{-}CMA\text{-}DHQ}}_{\mathcal{A},\,\mathsf{mklhs}_3}(\lambda) \leq \Pr[(\text{Type-2 forgery}) \vee (\text{Type-3 forgery})]$$

and by the union bound (Theorem 2.18),

$$\mathsf{Adv}^{\mathsf{HomUF\text{-}CMA\text{-}DHQ}}_{\mathcal{A},\,\mathsf{mklhs}_3}(\lambda) \leq \Pr[\text{Type-2 forgery}] + \Pr[\text{Type-3 forgery}]$$

Substituting each probability from the previous subsections yields

$$\begin{aligned}\mathsf{Adv}^{\mathsf{HomUF\text{-}CMA\text{-}DHQ}}_{\mathcal{A},\,\mathsf{mklhs}_3}(\lambda) &\leq \left(\frac{1}{1-\frac{1}{q-1}} + \frac{1}{1-\frac{1}{q}}\right) \cdot \mathsf{Adv}^{\text{co-CDH}^*}_{\mathcal{B}}(\lambda) \\ &= \left(\frac{q-1}{q-2} + \frac{q}{q-1}\right) \cdot \mathsf{Adv}^{\text{co-CDH}^*}_{\mathcal{B}}(\lambda).\end{aligned}$$

Since $\frac{q-1}{q-2} + \frac{q}{q-1} \leq 3$ for all primes $q \geq 5$, we have

$$\mathsf{Adv}^{\mathsf{HomUF\text{-}CMA\text{-}DHQ}}_{\mathcal{A},\,\mathsf{mklhs}_3}(\lambda) \;\leq\; 3 \cdot \mathsf{Adv}^{\text{co-CDH}^*}_{\mathcal{B}}(\lambda).$$

This completes the proof of Theorem 4.1.

□

4.3 Correctness of the Simulation

We now provide justifications for the claim about simulation correctness provided in the proof of Theorem 4.1.

Proof of Claim 4.3. We argue that all values returned by $\mathcal{B}$ are distributed as in a real execution of the scheme.

Public parameters. The simulator sets $\tilde{g}_1 = g_1^s h$ where $s \xleftarrow{\$} \mathbb{Z}_q$. Since $\mathbb{G}_1$ is cyclic with generator g_1, there exists some $b \in \mathbb{Z}_q$ such that $h = g_1^b$. Hence

$$\tilde{g}_1 = g_1^{s+b}.$$

Because s is sampled uniformly and independently in $\mathbb{Z}_q$, the exponent $s + b$ is uniform in $\mathbb{Z}_q$. Therefore $\tilde{g}_1 = 1_{\mathbb{G}_1}$ with probability $1/q$, and otherwise $\tilde{g}_1$ is a generator of $\mathbb{G}_1$. Hence, $\tilde{g}_1$ is a generator except with negligible probability, matching the real setup. If $\tilde{g}_1 = 1_{\mathbb{G}_1}$, $\mathcal{B}$ can check this during the setup phase and simply resample s to guarantee that $\tilde{g}_1$ is a generator.

Identity queries. For an identity id, $\mathcal{B}$ samples $y_{\mathsf{id}} \xleftarrow{\$} \mathbb{Z}_q^*$ and returns $\mathsf{pk}_{\mathsf{id}} = (X_2)^{y_{\mathsf{id}}} = g_2^{xy_{\mathsf{id}}}$. In the real scheme, the public key is generated as $g_2^{\mathsf{sk}_{\mathsf{id}}}$ for a uniformly sampled secret key $\mathsf{sk}_{\mathsf{id}} \xleftarrow{\$} \mathbb{Z}_q^*$. Hence, it is enough to show that the exponent xy_{id} is uniformly distributed over $\mathbb{Z}_q^*$.

First, we note that if $x = 0$, the given co-CDH* instance is trivial, since the solution is $h^x = h^0 = 1_{\mathbb{G}_1}$. In this case the reduction can immediately output $1_{\mathbb{G}_1}$ and solve the instance. Hence, from now on we assume $x \neq 0$.

Since q is prime and $x \neq 0$, we have that x is in the multiplicative group $\mathbb{Z}_q^*$. Multiplication by x is therefore a permutation on $\mathbb{Z}_q^*$ and for $y_{\mathsf{id}} \leftarrow \mathbb{Z}_q^*$ sampled uniformly, the value xy_{id} is also uniformly distributed over $\mathbb{Z}_q^*$.

Hash queries. Fresh hash queries are answered as $\mathcal{H}(\ell) = h^r$ with $r \xleftarrow{\$} \mathbb{Z}_q$, which are uniform in $\mathbb{G}_1$ since h is a generator. For signing queries, the oracle is programmed as (4.1). Using again the decomposition $h = g_1^b$ for $b \in \mathbb{Z}_q$ and substituting into (4.1) gives

$$\mathcal{H}(\ell) = g_1^t h^{-m} = g_1^t \, (g_1^b)^{-m} = g_1^{t-bm}.$$

Since $t \xleftarrow{\$} \mathbb{Z}_q$ is sampled uniformly and independently of the other terms, the exponent is uniformly distributed in $\mathbb{Z}_q$ and hence $\mathcal{H}(\ell)$ is also uniformly distributed in $\mathbb{G}_1$.

Signing queries. We recall the definition of γ from mklhs in Section 2.7 but with our substituted $\tilde{g}_1$ generator,

$$\gamma = (\mathcal{H}(\ell) \cdot \tilde{g}_1^m)^{\mathsf{sk}}.$$

Substituting the programmed hash from (4.1), $\tilde{g}_1 = g_1^s h$, $g_1^x = X_1$, and $\mathsf{sk} = xy_{\mathsf{id}}$ yields

$$\begin{aligned}
\gamma &= [(g_1^t h^{-m})(g_1^s h)^m]^{xy_{\mathsf{id}}} \\
&= (g_1^{ms+t} h^{m-m})^{xy_{\mathsf{id}}} \\
&= (g_1^{ms+t})^{xy_{\mathsf{id}}} \\
&= \left((g_1^x)^{ms+t}\right)^{y_{\mathsf{id}}} \\
&= \left(X_1^{sm+t}\right)^{y_{\mathsf{id}}},
\end{aligned}$$

which is identical to γ returned by $\mathcal{B}$ in (4.2). Thus, signatures returned by $\mathcal{B}$ are correct and distributed identically to those generated by the real signing algorithm.

Hence, all values returned by $\mathcal{B}$ are distributed as in a real execution of mklhs, and the view of $\mathcal{A}$ in its interaction with $\mathcal{B}$ is identical to its view in the real HomUF-CMA-DHQ experiment, which proves the claim. □

5

A MKHS Construction for Bounded-Rank Quadratic Evaluation

In this chapter, we present an extension of mklhs supporting limited quadratic evaluations in the Type 3 pairing setting. We refer to the resulting scheme as mkqhs-br for *multi-key quadratic homomorphic signature scheme* with *bounded rank*. Our construction builds directly on mklhs_3 by keeping Setup, KeyGen, and Sign the same, while the new quadratic capability is introduced entirely through modifications to Eval and Verify.

5.1 Quadratic Evaluations Through Rank Decompositions

The challenge of quadratic evaluations. In the linear setting of mklhs, an evaluator computes a weighted sum simply by exponentiating and multiplying signatures, since these operations correspond directly to scaling and adding the underlying messages.

Quadratic operations, however, require computing a product $m_i \cdot m_j$, which means multiplying two quantities both encoded in the exponents of their respective signatures, something group arithmetic cannot do. The Eval algorithm could exponentiate by the message directly to obtain $(g^{m_i})^{m_j} = g^{m_i m_j}$, but then m_j appears as a bare scalar exponent rather than being encoded in a group element. Thus, the verifier cannot reconstruct it from the public key and label hashes alone without knowing m_j explicitly, breaking succinctness.

The pairing also offers a possible solution, the bilinear property $e(g_1^{m_i}, g_2^{m_j}) = e(g_1, g_2)^{m_i m_j}$ does produce a product without revealing the messages. However, a naive attempt leads to verification equations where label hashes and secret keys of different signers become tangled, collapsing information that must remain separate for verification to work. Extending mklhs to quadratic functions therefore requires a more careful approach, and we describe ours next.

Rank decomposition for quadratic homomorphism. The main idea behind our constructions is to factor the quadratic part of the computation into products of linear evaluations. As shown in Section 2.3, any matrix $Q \in \mathbb{Z}_q^{n\times n}$ of rank at most R can be decomposed as

$$Q = \sum_{r=1}^{R} \mathbf{u}_r \mathbf{v}_r^\top,$$

where $\mathbf{u}_r, \mathbf{v}_r \in \mathbb{Z}_q^n$ are coefficient vectors. Hence, by Corollary 2.15, any quadratic form also factors as

$$\mathbf{m}^\top Q\, \mathbf{m} = \sum_{r=1}^{R} \Big(\mathbf{u}_r^\top \mathbf{m}\Big)\Big(\mathbf{v}_r^\top \mathbf{m}\Big).$$

Each factor $\mathbf{u}_r^\top \mathbf{m} = \sum_{i=1}^n u_{i,r} m_i$ and $\mathbf{v}_r^\top \mathbf{m} = \sum_{i=1}^n v_{i,r} m_i$ is a linear combination of messages and can therefore be encoded in group elements using the same homomorphism mechanism as in mklhs.

5.2 The Supported Function Class

Motivated by the rank decomposition above, our scheme supports quadratic polynomials of the form

$$f(m_1, \ldots, m_n) = \sum_{i=1}^{n} a_i m_i + \sum_{r=1}^{R} \left(\sum_{i=1}^{n} u_{i,r} m_i\right)\left(\sum_{i=1}^{n} v_{i,r} m_i\right), \tag{5.1}$$

where $m_i, a_i, u_{i,r}, v_{i,r} \in \mathbb{Z}_q$, or equivalently,

$$f(\mathbf{m}) = \mathbf{a}^\top \mathbf{m} + \mathbf{m}^\top Q \mathbf{m},$$

where $\mathbf{m}, \mathbf{a} \in \mathbb{Z}_q^n$, and $Q \in \mathbb{Z}_q^{n\times n}$ is a matrix of rank at most R.

The evaluated signature size of mkqhs-br grows as $O(tR)$ for $t = |\mathsf{id} \in \mathcal{P}|$. We ensure succinctness by restricting admissible functions to those whose quadratic form satisfies $\mathrm{rank}(Q) = R \le O(\log n)$, such that the evaluated signature sizes remain bounded by $\mathsf{poly}(\lambda, t, \log n)$. Hence the name *bounded rank*.

For the admissible function class (5.1), we also require that every label ℓ_i appearing in a labeled program $\mathcal{P}$ contributes non-trivially to f, meaning that for each $\ell_i \in \mathcal{P}$, either $a_i \neq 0$ or $(u_{i,r}, v_{i,r}) \neq (0,0)$ for some $r \in [R]$. We consider this the natural extension of the restriction $a_i \neq 0$ imposed on mklhs_3.

5.3 Expressiveness of the Low-Rank Restriction

We note that while admissible functions are required to have rank at most $R \le O(\log n)$ for our construction, we do not consider this all too restrictive. Many natural quadratic quantities have low-rank structure by construction. For example, the inner product of two n-dimensional vectors $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_q^n$, $\langle \mathbf{x}, \mathbf{y}\rangle = \sum_{i=1}^n x_i y_i$, corresponds to a matrix Q of rank one. Furthermore, the rank restriction aligns naturally with the use of low-rank approximations in statistics and data analysis. In

many large-scale computations, high-rank matrices are replaced by low-rank approximations to reduce computational and storage costs while preserving the dominant structure of the data [31]. Our construction supports verification of computations performed on such low-rank representations.

On the other hand, the rank restriction prevents succinct evaluation of arbitrary quadratic forms. Most notably, direct squares m_i^2 across all inputs are not supported without loss of succinctness. For instance as in Example 2.16, the sum of squares $\sum_{i=1}^{n} m_i^2$ corresponds to the identity matrix $Q = I_n$, which has full rank n and would cause the signature size to scale linearly with n. We address the challenge of supporting direct squares while maintaining succinctness in Chapter 6 via an additional extension.

Despite the bounded rank restriction, our construction remains correct and secure for any rank, it is only the succinctness guarantee that breaks down when R grows beyond $O(\log n)$. Hence, we emphasize that the low-rank restriction is only a by-product of the succinctness requirement and not a limitation of the approach.

5.4 The mkqhs-br Construction

In the presentation of our construction, boldface symbols denote column vectors of length R over $\mathbb{Z}_q$, *i.e.*, for a vector $\boldsymbol{\mu}$, we mean $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_R)^\top \in \mathbb{Z}_q^R$. For $\boldsymbol{\rho}, \boldsymbol{\mu} \in \mathbb{Z}_q^R$, we write their inner product as $\langle \boldsymbol{\rho}, \boldsymbol{\mu} \rangle = \sum_{r=1}^{R} \rho_r \mu_r$. For each $i \in [n]$, we also write $\mathbf{u}_i = (u_{i,1}, \ldots, u_{i,R})^\top$ and $\mathbf{v}_i = (v_{i,1}, \ldots, v_{i,R})^\top$, corresponding to the coefficients of the quadratic form term of the supported polynomial class. Vectors may also be over $\mathbb{G}_1$, so by $\boldsymbol{\gamma}$, we mean $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_R)^\top \in \mathbb{G}_1^R$. For vectors $\boldsymbol{\gamma} \in \mathbb{G}_1^R$ and $\boldsymbol{\rho} \in \mathbb{Z}_q^R$, we denote their element-wise exponentiation by $\boldsymbol{\gamma}^{\boldsymbol{\rho}} = \prod_{r=1}^{R} \gamma_r^{\rho_r}$.

The Eval and Verify algorithms of mkqhs-br are given in Figure 5.2. While the Setup, KeyGen and Sign algorithms are identical to those of mklhs_3 in Figure 4.1 and Figure 2.3, we recall them here in Figure 5.1 for completeness.

$\mathsf{Setup}(1^\lambda)$	$\mathsf{KeyGen}(\mathsf{pp})$	$\mathsf{Sign}(\mathsf{sk}_{\mathsf{id}}, \ell, m)$
$\mathcal{G} \leftarrow \mathsf{BilinGroup}_3(\lambda)$	$\mathsf{id} \xleftarrow{\$} \mathsf{ID}$	$\gamma = (H(\ell) \cdot g_1^m)^{\mathsf{sk}_{\mathsf{id}}}$
define sets:	$\mathsf{sk}_{\mathsf{id}} \xleftarrow{\$} \mathbb{Z}_q^*$	$\mu = m$
$\mathsf{ID}, \mathcal{T} \subseteq \{0,1\}^*$	$\mathsf{pk}_{\mathsf{id}} = g_2^{\mathsf{sk}_{\mathsf{id}}} \in \mathbb{G}_2$	**return** $\sigma = (\mathsf{id}, \gamma, \mu)$
$\mathcal{M} \subseteq \mathbb{Z}_q$	**return** $(\mathsf{sk}_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}}, \mathsf{id})$	
fix hash:		
$H : \{0,1\}^* \to \mathbb{G}_1$		
$\mathsf{pp} \leftarrow (\mathcal{G}, \mathsf{ID}, \mathcal{M}, \mathcal{T}, H)$		
return pp		

Figure 5.1: The Setup, KeyGen, and Sign algorithms of mklhs_3 and mkqhs-br.

$\mathsf{Eval}(\mathcal{P}, \{\sigma_i\}_{i=1}^n)$

1 : **parse** $\mathcal{P} = (f, \{\ell_i\}_{i=1}^n)$
2 : **parse** $f = (\{a_i, \mathbf{u}_i, \mathbf{v}_i\}_{i=1}^n)$
3 : **parse** $\mathbf{u}_i = (u_{i,1}, \ldots, u_{i,R})^\top$
4 : **parse** $\mathbf{v}_i = (v_{i,1}, \ldots, v_{i,R})^\top$
5 : **parse** $\sigma_i = (\mathsf{id}_i, \gamma_i, \mu_i)$
6 : $\gamma^{(a)} = \prod_{i=1}^n \gamma_i^{a_i}$
7 : **for** $r \in [R]$
8 : $\quad \gamma_r^{(u)} = \prod_{i=1}^n \gamma_i^{u_{i,r}}, \;\; \gamma_r^{(v)} = \prod_{i=1}^n \gamma_i^{v_{i,r}}$
9 : $\boldsymbol{\gamma}^{(u)} = (\gamma_1^{(u)}, \ldots, \gamma_R^{(u)})^\top$
10 : $\boldsymbol{\gamma}^{(v)} = (\gamma_1^{(v)}, \ldots, \gamma_R^{(v)})^\top$
11 : $\tilde{\gamma} = \left(\gamma^{(a)}, \boldsymbol{\gamma}^{(u)}, \boldsymbol{\gamma}^{(v)}\right)$
12 : **for** $\mathsf{id} \in \mathcal{P}$
13 : $\quad \mathcal{I}_{\mathsf{id}} = \{\, i \in [n] \mid \ell_i = (\mathsf{id}, \cdot) \,\}$
14 : $\quad \mu_{\mathsf{id}}^{(a)} = \sum_{i \in \mathcal{I}_{\mathsf{id}}} a_i \mu_i$
15 : $\quad \boldsymbol{\mu}_{\mathsf{id}}^{(u)} = \sum_{i \in \mathcal{I}_{\mathsf{id}}} \mu_i \mathbf{u}_i, \;\; \boldsymbol{\mu}_{\mathsf{id}}^{(v)} = \sum_{i \in \mathcal{I}_{\mathsf{id}}} \mu_i \mathbf{v}_i$
16 : $\quad \tilde{\mu}_{\mathsf{id}} = (\mu_{\mathsf{id}}^{(a)}, \boldsymbol{\mu}_{\mathsf{id}}^{(u)}, \boldsymbol{\mu}_{\mathsf{id}}^{(v)})$
17 : **return** $\tilde{\sigma} = (\tilde{\gamma}, \{\tilde{\mu}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}})$

$\mathsf{Verify}(\mathcal{P}, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}}, \tilde{m}, \tilde{\sigma})$

1 : **parse** $\mathcal{P} = (f, \{\ell_i\}_{i=1}^n), \;\; f = (\{a_i, \mathbf{u}_i, \mathbf{v}_i\}_{i=1}^n)$
2 : **parse** $\tilde{\sigma} = (\tilde{\gamma}, \{\tilde{\mu}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}}), \;\; \tilde{\gamma} = \left(\gamma^{(a)}, \boldsymbol{\gamma}^{(u)}, \boldsymbol{\gamma}^{(v)}\right)$
3 : **for** $\mathsf{id} \in \mathcal{P}$
4 : $\quad$ **parse** $\tilde{\mu}_{\mathsf{id}} = (\mu_{\mathsf{id}}^{(a)}, \boldsymbol{\mu}_{\mathsf{id}}^{(u)}, \boldsymbol{\mu}_{\mathsf{id}}^{(v)})$
5 : $\quad \mathcal{I}_{\mathsf{id}} = \{\, i \in [n] \mid \ell_i = (\mathsf{id}, \cdot) \,\}$
6 : $\mu^{(a)} = \sum_{\mathsf{id} \in \mathcal{P}} \mu_{\mathsf{id}}^{(a)}$
7 : $\boldsymbol{\mu}^{(u)} = \sum_{\mathsf{id} \in \mathcal{P}} \boldsymbol{\mu}_{\mathsf{id}}^{(u)}, \quad \boldsymbol{\mu}^{(v)} = \sum_{\mathsf{id} \in \mathcal{P}} \boldsymbol{\mu}_{\mathsf{id}}^{(v)}$
8 : $\mathsf{ver}_1 \leftarrow \left[\tilde{m} == \mu^{(a)} + \langle \boldsymbol{\mu}^{(u)}, \boldsymbol{\mu}^{(v)} \rangle\right]$
9 : $\mathsf{ver}_2 \leftarrow \left[e(\gamma^{(a)}, g_2) == \prod_{\mathsf{id} \in \mathcal{P}} e\left(g_1^{\mu_{\mathsf{id}}^{(a)}} \cdot \prod_{i \in \mathcal{I}_{\mathsf{id}}} H(\ell_i)^{a_i}, \mathsf{pk}_{\mathsf{id}}\right)\right]$
10 : $(\boldsymbol{\rho}, \boldsymbol{\rho}') \xleftarrow{\$} (\mathbb{Z}_q^*)^{R \times 2}$
11 : $\Gamma_\rho = (\boldsymbol{\gamma}^{(u)})^{\boldsymbol{\rho}} (\boldsymbol{\gamma}^{(v)})^{\boldsymbol{\rho}'}$
12 : $\mathsf{ver}_3 \leftarrow \left[e(\Gamma_\rho, g_2) == \prod_{\mathsf{id} \in \mathcal{P}} e\left(g_1^{\langle \boldsymbol{\rho}, \boldsymbol{\mu}_{\mathsf{id}}^{(u)} \rangle + \langle \boldsymbol{\rho}', \boldsymbol{\mu}_{\mathsf{id}}^{(v)} \rangle} \cdot \prod_{i \in \mathcal{I}_{\mathsf{id}}} H(\ell_i)^{\langle \boldsymbol{\rho}, \mathbf{u}_i \rangle + \langle \boldsymbol{\rho}', \mathbf{v}_i \rangle}, \mathsf{pk}_{\mathsf{id}}\right)\right]$
13 : **return** $[\mathsf{ver}_1 \wedge \mathsf{ver}_2 \wedge \mathsf{ver}_3]$

Figure 5.2: Eval and Verify for mkqhs-br. Its Setup, KeyGen, and Sign algorithms are as mklhs_3 in Figure 5.1. Green text indicates additions compared to mklhs_3.

Construction overview. Unlike mklhs_3, where $\tilde{\gamma}$ is a single element in $\mathbb{G}_1$, in mkqhs-br, $\tilde{\gamma}$ contains multiple components. Concretely, $\tilde{\gamma} = \left(\gamma^{(a)}, \boldsymbol{\gamma}^{(u)}, \boldsymbol{\gamma}^{(v)}\right)$ where $\gamma^{(a)}$ represents the linear form in (5.1) and $(\boldsymbol{\gamma}^{(u)}, \boldsymbol{\gamma}^{(v)})$ corresponds to the rank components of the quadratic form term in (5.1). Notably, all individual group components represents linear forms and are built using the same aggregation procedure as in mklhs_3.

Similarly, the per-identity message aggregates $\tilde{\mu}_{\mathsf{id}}$ are expanded to include both the linear message aggregate $\mu_{\mathsf{id}}^{(a)}$ and two vectors $\boldsymbol{\mu}_{\mathsf{id}}^{(u)}$ and $\boldsymbol{\mu}_{\mathsf{id}}^{(v)}$. The latter track how each signer contributes to the u- and v-parts of each rank product in (5.1).

Lastly, verification proceeds in three different checks.

Algebraic check (ver_1). This confirms that the claimed output $\tilde{m}$ equals the sum of the linear and quadratic message aggregates per identity.

Linear check (ver_2). Identical to the pairing check in mklhs_3, it verifies that the linear evaluations were performed correctly against the public keys $\mathsf{pk}_{\mathsf{id}}$.

Quadratic check (ver_3). This verifies that the quadratic evaluations were performed correctly against the public keys $\{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}$. To avoid $2R$ separate pairing evaluations per identity, the verifier samples random challenges $\boldsymbol{\rho}, \boldsymbol{\rho}' \xleftarrow{\$} (\mathbb{Z}_q^*)^R$ to collapse verification of $(\boldsymbol{\gamma}^{(u)}, \boldsymbol{\gamma}^{(v)})$ into a single pairing evaluation per identity.

5.5 Correctness of mkqhs-br

We show that mkqhs-br satisfies evaluation and authentication correctness as in Definition 2.27. We recall that evaluation correctness requires that given a labeled program $\mathcal{P} = (f, \{\ell_i\}_{i=1}^n)$ and honestly signed messages $\{m_i\}_{i=1}^n$, evaluating the corresponding signatures should produce a signature $\tilde{\sigma}$ that verifies successfully against the output $\tilde{m} = f(m_1, \ldots, m_n)$ with overwhelming probability.

Authentication correctness then follows as a special case of evaluation correctness for $n = 1$, by setting $\mathcal{P}$ to the identity program $\mathcal{I}_\ell$, where $a_1 = 1$ and $R = 0$.

Suppose that the messages $\{m_i\}_{i=1}^n$ have been honestly signed, so that for each $i \in [n]$, the signature of m_i is $\sigma_i = (\mathsf{id}_i, \gamma_i, \mu_i)$, where

$$\gamma_i = (H(\ell_i) \cdot g_1^{m_i})^{\mathsf{sk}_{\mathsf{id}_i}} \quad \text{and} \quad \mu_i = m_i.$$

Let $\tilde{\sigma} \leftarrow \mathsf{Eval}(\mathcal{P}, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}, \{\sigma_i\}_{i\in[n]})$ be the evaluated signature. By mkqhs-br.Eval in Figure 5.2, $\tilde{\sigma}$ contains the $\mathbb{G}_1$ elements

$$\gamma^{(a)} = \prod_{i=1}^n \gamma_i^{a_i}, \qquad \gamma_r^{(u)} = \prod_{i=1}^n \gamma_i^{u_{i,r}}, \qquad \gamma_r^{(v)} = \prod_{i=1}^n \gamma_i^{v_{i,r}},$$

together with, for each $\mathsf{id} \in \mathcal{P}$, the $\mathbb{Z}_q$ values

$$\mu_{\mathsf{id}}^{(a)} = \sum_{i\in\mathcal{I}_{\mathsf{id}}} a_i m_i, \qquad \boldsymbol{\mu}_{\mathsf{id}}^{(u)} = \sum_{i\in\mathcal{I}_{\mathsf{id}}} m_i\, \mathbf{u}_i, \qquad \boldsymbol{\mu}_{\mathsf{id}}^{(v)} = \sum_{i\in\mathcal{I}_{\mathsf{id}}} m_i\, \mathbf{v}_i,$$

where we recall that $\mathcal{I}_{\mathsf{id}} = \{i \in [n] \mid \ell_i = (\mathsf{id}, \cdot)\}$.

We show that all three checks in Figure 5.2 deterministically pass for such honest evaluations.

Algebraic check (ver_1). Since $\{\mathcal{I}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}$ partitions $[n]$, summing the per-identity aggregates over $\mathcal{P}$ yields the global aggregates

$$\begin{aligned}
\mu^{(a)} &= \sum_{\mathsf{id}\in\mathcal{P}} \mu_{\mathsf{id}}^{(a)} &&= \sum_{\mathsf{id}\in\mathcal{P}} \sum_{i\in\mathcal{I}_{\mathsf{id}}} a_i m_i &&= \sum_{i=1}^{n} a_i m_i, \\
\boldsymbol{\mu}^{(u)} &= \sum_{\mathsf{id}\in\mathcal{P}} \boldsymbol{\mu}_{\mathsf{id}}^{(u)} &&= \sum_{\mathsf{id}\in\mathcal{P}} \sum_{i\in\mathcal{I}_{\mathsf{id}}} m_i\, \mathbf{u}_i &&= \sum_{i=1}^{n} m_i\, \mathbf{u}_i, \\
\boldsymbol{\mu}^{(v)} &= \sum_{\mathsf{id}\in\mathcal{P}} \boldsymbol{\mu}_{\mathsf{id}}^{(v)} &&= \sum_{\mathsf{id}\in\mathcal{P}} \sum_{i\in\mathcal{I}_{\mathsf{id}}} m_i\, \mathbf{v}_i &&= \sum_{i=1}^{n} m_i\, \mathbf{v}_i,
\end{aligned}$$

whose components satisfy $\mu_r^{(u)} = \sum_{i=1}^{n} u_{i,r} m_i$ and $\mu_r^{(v)} = \sum_{i=1}^{n} v_{i,r} m_i$ for every $r \in [R]$. Hence

$$\begin{aligned}
\mu^{(a)} + \langle \boldsymbol{\mu}^{(u)}, \boldsymbol{\mu}^{(v)} \rangle &= \sum_{i=1}^{n} a_i m_i + \sum_{r=1}^{R} \Big(\sum_{i=1}^{n} u_{i,r} m_i \Big) \Big(\sum_{i=1}^{n} v_{i,r} m_i \Big) \\
&= f(m_1, \ldots, m_n) = \tilde{m},
\end{aligned}$$

so ver_1 always passes.

Linear pairing check (ver_2). The correctness of the verification check ver_2 follows directly from the correctness of mklhs_3. Both ver_2 in $\mathsf{mkqhs\text{-}br}$ and ver_2 in mklhs_3 perform identical checks, since $\gamma^{(a)} \in \mathbb{G}_1$ and $\mu_{\mathsf{id}}^{(a)} \in \mathbb{Z}_q$ in $\mathsf{mkqhs\text{-}br}$ are structured identically to $\tilde{\gamma}$ and $\tilde{\mu}_{\mathsf{id}}$ in mklhs_3.

Quadratic pairing check (ver_3). To prove the correctness of the third verification check, we begin by expanding the group element Γ_ρ defined on line 11 in $\mathsf{mkqhs\text{-}br.Verify}$ from Figure 5.2,

$$\begin{aligned}
\Gamma_\rho &= \prod_{r=1}^{R} (\gamma_r^{(u)})^{\rho_r} (\gamma_r^{(v)})^{\rho'_r} \\
&= \prod_{r=1}^{R} \left(\prod_{i=1}^{n} \gamma_i^{u_{i,r}} \right)^{\rho_r} \left(\prod_{i=1}^{n} \gamma_i^{v_{i,r}} \right)^{\rho'_r} \\
&= \prod_{i=1}^{n} \gamma_i^{\sum_{r=1}^{R} (\rho_r u_{i,r} + \rho'_r v_{i,r})} \\
&= \prod_{i=1}^{n} \gamma_i^{\langle \boldsymbol{\rho}, \mathbf{u}_i \rangle + \langle \boldsymbol{\rho}', \mathbf{v}_i \rangle}.
\end{aligned}$$

Next, we substitute the honest signatures $\gamma_i = (H(\ell_i) \cdot g_1^{m_i})^{\mathsf{sk}_{\mathsf{id}_i}}$ and partition $[n]$ into subsets $\mathcal{I}_{\mathsf{id}}$ for each unique identity $\mathsf{id} \in \mathcal{P}$, yielding:

$$\begin{aligned}\Gamma_\rho &= \prod_{\mathsf{id}\in\mathcal{P}} \prod_{i\in\mathcal{I}_{\mathsf{id}}} (H(\ell_i) \cdot g_1^{m_i})^{\mathsf{sk}_{\mathsf{id}}(\langle \boldsymbol{\rho}, \mathbf{u}_i\rangle + \langle \boldsymbol{\rho}', \mathbf{v}_i\rangle)} \\ &= \prod_{\mathsf{id}\in\mathcal{P}} \left(g_1^{\sum_{i\in\mathcal{I}_{\mathsf{id}}} m_i(\langle \boldsymbol{\rho}, \mathbf{u}_i\rangle + \langle \boldsymbol{\rho}', \mathbf{v}_i\rangle)} \cdot \prod_{i\in\mathcal{I}_{\mathsf{id}}} H(\ell_i)^{\langle \boldsymbol{\rho}, \mathbf{u}_i\rangle + \langle \boldsymbol{\rho}', \mathbf{v}_i\rangle} \right)^{\mathsf{sk}_{\mathsf{id}}}.\end{aligned}$$

Using the linearity of the inner product, we can simplify the exponent of g_1 as

$$\begin{aligned}\sum_{i\in\mathcal{I}_{\mathsf{id}}} m_i \left(\langle \boldsymbol{\rho}, \mathbf{u}_i\rangle + \langle \boldsymbol{\rho}', \mathbf{v}_i\rangle\right) &= \left\langle \boldsymbol{\rho}, \sum_{i\in\mathcal{I}_{\mathsf{id}}} m_i \mathbf{u}_i \right\rangle + \left\langle \boldsymbol{\rho}', \sum_{i\in\mathcal{I}_{\mathsf{id}}} m_i \mathbf{v}_i \right\rangle \\ &= \langle \boldsymbol{\rho}, \boldsymbol{\mu}_{\mathsf{id}}^{(u)}\rangle + \langle \boldsymbol{\rho}', \boldsymbol{\mu}_{\mathsf{id}}^{(v)}\rangle.\end{aligned}$$

Substituting this back into our expression for $\Gamma_{\boldsymbol{\rho}}$ yields

$$\Gamma_\rho = \prod_{\mathsf{id}\in\mathcal{P}} \left(g_1^{\langle \boldsymbol{\rho}, \boldsymbol{\mu}_{\mathsf{id}}^{(u)}\rangle + \langle \boldsymbol{\rho}', \boldsymbol{\mu}_{\mathsf{id}}^{(v)}\rangle} \cdot \prod_{i\in\mathcal{I}_{\mathsf{id}}} H(\ell_i)^{\langle \boldsymbol{\rho}, \mathbf{u}_i\rangle + \langle \boldsymbol{\rho}', \mathbf{v}_i\rangle} \right)^{\mathsf{sk}_{\mathsf{id}}}.$$

Finally, computing the pairing of $\Gamma_{\boldsymbol{\rho}}$ with g_2 and applying the definition $\mathsf{pk}_{\mathsf{id}} = g_2^{\mathsf{sk}_{\mathsf{id}}}$, we obtain

$$\begin{aligned}e(\Gamma_\rho, g_2) &= \prod_{\mathsf{id}\in\mathcal{P}} e\left(\left(g_1^{\langle \boldsymbol{\rho}, \boldsymbol{\mu}_{\mathsf{id}}^{(u)}\rangle + \langle \boldsymbol{\rho}', \boldsymbol{\mu}_{\mathsf{id}}^{(v)}\rangle} \cdot \prod_{i\in\mathcal{I}_{\mathsf{id}}} H(\ell_i)^{\langle \boldsymbol{\rho}, \mathbf{u}_i\rangle + \langle \boldsymbol{\rho}', \mathbf{v}_i\rangle} \right)^{\mathsf{sk}_{\mathsf{id}}}, g_2 \right) \\ &= \prod_{\mathsf{id}\in\mathcal{P}} e\left(g_1^{\langle \boldsymbol{\rho}, \boldsymbol{\mu}_{\mathsf{id}}^{(u)}\rangle + \langle \boldsymbol{\rho}', \boldsymbol{\mu}_{\mathsf{id}}^{(v)}\rangle} \cdot \prod_{i\in\mathcal{I}_{\mathsf{id}}} H(\ell_i)^{\langle \boldsymbol{\rho}, \mathbf{u}_i\rangle + \langle \boldsymbol{\rho}', \mathbf{v}_i\rangle}, \mathsf{pk}_{\mathsf{id}} \right),\end{aligned}$$

which exactly matches the right-hand side of check ver_3 in Figure 5.2. Thus, the check passes.

Since all three checks pass deterministically, regardless of the random choice of $(\boldsymbol{\rho}, \boldsymbol{\rho}')$ in mkqhs-br.Verify, the scheme satisfies evaluation correctness. Authentication correctness now follows as the special case $n = 1$ with f given by the identity program $\mathcal{I}_\ell$ with $a_1 = 1$ and $R = 0$.

5.6 Security Theorem and Proof Outline

Theorem 5.1. *Assuming that the co-CDH* problem is hard,* mkqhs-br *in Figure 5.2 is secure under* HomUF-CMA-DHQ *in the random oracle model. Formally, let $\mathcal{A}$ be a PPT adversary in the* HomUF-CMA-DHQ *security experiment. Then its advantage is bounded by*

$$\mathsf{Adv}^{\text{HomUF-CMA-DHQ}}_{\mathcal{A},\,\text{mkqhs-br}}(\lambda) \leq 5 \cdot \mathsf{Adv}^{\text{co-CDH}^*}_{\mathcal{B}}(\lambda),$$

where $\mathcal{B}$ is a PPT algorithm that solves the co-CDH problem with advantage* $\mathsf{Adv}^{\text{co-CDH}^*}_{\mathcal{B}}(\lambda)$.

Proof Outline. The proof follows a similar structure as Theorem 4.1. The simulation of the HomUF-CMA-DHQ experiment is identical, with correctness given by Claim 4.3.

Type-1 forgeries reduce to Type-3 forgeries when folding dataset identifiers into the tag space by the same argument as in Subsection 4.2.1.

Type-2 forgeries are now split into two sub-cases. If the linear aggregates are inconsistent for some identity, $\mathcal{B}$ extract a co-CDH* solution from $\gamma^{(a)}$ exactly as in Theorem 4.1. Otherwise, the quadratic aggregates must be inconsistent, and $\mathcal{B}$ instead extracts from the rank components $(\boldsymbol{\gamma}^{(u)}, \boldsymbol{\gamma}^{(v)})$. For this case, we define new mismatch terms $\delta^{(u)}_{\mathsf{id},r}$ and $\delta^{(v)}_{\mathsf{id},r}$ for each identity and rank, taking the same role as δ_{id} in the linear case. Finally, bounding the abort probability with Schwartz–Zippel results in an upper bound of $2/(q-1)$ rather than $1/(q-1)$, accounting for the new randomness from $(\boldsymbol{\rho}, \boldsymbol{\rho}') \xleftarrow{\$} (\mathbb{Z}_q^*)^R$.

Type-3 forgeries are similarly split into two sub-cases. If there exists an unsigned label ℓ_i^* with index i and $a_i^* \neq 0$, $\mathcal{B}$ extracts a co-CDH* solution from $\gamma^{(a)*}$ exactly as in Theorem 4.1. Otherwise, admissibility guarantees that there exists some unsigned index i with $a_i^* = 0$ and $(u_{i,r}^*, v_{i,r}^*) \neq (0,0)$ for some $r \in [R]$, and $\mathcal{B}$ instead extracts from Γ_ρ^*. The main difference from the Type-2 quadratic case is that the unsigned labels were programmed in the hash phase as $\mathcal{H}(\ell_i^*) = h^{r_i}$, so the extraction has to account for two different hash expressions. The abort probability is again bounded by Schwartz–Zippel, resulting in upper bound of $2/q$ rather than $1/q$ from additional randomness from $(\boldsymbol{\rho}, \boldsymbol{\rho}') \xleftarrow{\$} (\mathbb{Z}_q^*)^R$

Finally, the total advantage is calculated by combining the bounds from each forgery-type.

5.7 Proving Security

Proof. We distinguish between the three types of forgeries from Definition 3.3. We define a reduction $\mathcal{B}$ with identical simulation to the one of Subsection 4.2.2 and whose correctness follows from Claim 4.3.

We start by noting that with the same reasoning as in Theorem 4.1, all Type-1 forgeries reduce to Type-3 forgeries, so we disregard them in our analysis.

5.7.1 co-CDH* Extraction From Type-2 Forgeries

Let $\mathcal{A}$ output a successful Type-2 forgery $(\mathcal{P}^*, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}^*}, \tilde{m}^*, \tilde{\sigma}^*)$ after its interaction with $\mathcal{B}$, where $\mathcal{P}^* = (f^*, \ell_1^*, \ldots, \ell_n^*)$ and $f^* = (\{a_i^*, \mathbf{u}_i^*, \mathbf{v}_i^*\})$. By definition of a Type-2 forgery, we have

$$\mathsf{Verify}(\mathcal{P}^*, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}^*}, \tilde{m}^*, \tilde{\sigma}^*) = 1 \quad \text{and} \quad \tilde{m}^* \neq f^*(m_1, \ldots, m_n),$$

where m_i is the message that $\mathcal{A}$ queried during the sign query phase for label ℓ_i^*. By the same reasoning as for Theorem 4.1, for every label ℓ_i^* appearing in the program $\mathcal{P}^*$, $\mathcal{H}(\ell_i^*)$ was programmed during the signing phase and *not* the hash phase.

$\mathcal{B}$ first parses $\tilde{\sigma}^* = \left(\left(\gamma^{(a)*}, \left(\gamma_r^{(u)*}, \gamma_r^{(v)*}\right)_{r=1}^{R}\right), \left\{\mu_{\mathsf{id}}^{(a)*}, \left(\mu_{\mathsf{id},r}^{(u)*}, \mu_{\mathsf{id},r}^{(v)*}\right)_{r=1}^{R}\right\}_{\mathsf{id}\in\mathcal{P}^*}\right)$, and successful verification implies

$$\tilde{m}^* = \mu^{(a)*} + \sum_{r=1}^{R} \mu_r^{(u)*}\mu_r^{(v)*}, \tag{5.2}$$

where we write

$$\mu^{(a)*} = \sum_{\mathsf{id}\in\mathcal{P}^*} \mu_{\mathsf{id}}^{(a)*}, \quad \mu_r^{(u)*} = \sum_{\mathsf{id}\in\mathcal{P}^*} \mu_{\mathsf{id},r}^{(u)*}, \quad \mu_r^{(v)*} = \sum_{\mathsf{id}\in\mathcal{P}^*}^{t} \mu_{\mathsf{id},r}^{(v)*}.$$

We also define their honestly computed counterparts as

$$\mu^{(a)\circ} := \sum_{i=1}^{n} a_i^* m_i, \quad \mu_r^{(u)\circ} := \sum_{i=1}^{n} u_{i,r}^* m_i, \quad \mu_r^{(v)\circ} = \sum_{i=1}^{n} v_{i,r}^* m_i.$$

Since the forgery is Type-2,

$$\tilde{m}^* \neq \mu^{(a)\circ} + \sum_{r=1}^{R} \mu_r^{(u)\circ}\mu_r^{(v)\circ}. \tag{5.3}$$

Therefore, at least one of the following must hold for a Type-2 forgery:

Case 1: linear forgery. There exists some identity $\mathsf{id} \in \mathcal{P}^*$ such that

$$\mu_{\mathsf{id}}^{(a)*} \neq \sum_{i\in\mathcal{I}_{\mathsf{id}}} a_i^* m_i.$$

In this case, $\mathcal{B}$ proceeds exactly as in Theorem 4.1 for mklhs_3, where the co-CDH* solution is extracted from $\tilde{\gamma}^{(a)}$ instead. By Theorem 4.1, we get

$$\mathsf{Adv}_{\mathcal{B}}^{\text{co-CDH}^*}(\lambda) \;\geq\; \left(1 - \frac{1}{q-1}\right) \cdot \Pr[\mathcal{A} \text{ outputs a valid Type-2 forgery}].$$

Case 2: quadratic forgery. Otherwise, all linear aggregates are correct, *i.e.*,

$$\mu_{\mathsf{id}}^{(a)*} = \sum_{i\in\mathcal{I}_{\mathsf{id}}} a_i^* m_i, \quad \text{for all } \mathsf{id} \in \mathcal{P}^*.$$

Hence, $\mu^{(a)*} = \mu^{(a)\circ}$, and equations (5.2) and (5.3) imply

$$\sum_{r=1}^{R} \mu_r^{(u)*}\mu_r^{(v)*} \neq \sum_{r=1}^{R} \mu_r^{(u)\circ}\mu_r^{(u)\circ}. \tag{5.4}$$

The rest of this subsection proceeds in the quadratic forgery setting.

Identifying inconsistent identities. For each identity id in $\mathcal{P}$, we define the *mismatch terms* for rank component r as

$$\delta^{(u)}_{\mathsf{id},r} := \mu^{(u)*}_{\mathsf{id},r} - \sum_{i\in\mathcal{I}_{\mathsf{id}}} u^*_{i,r} m_i, \quad \delta^{(v)}_{\mathsf{id},r} := \mu^{(v)*}_{\mathsf{id},r} - \sum_{i\in\mathcal{I}_{\mathsf{id}}} v^*_{i,r} m_i. \tag{5.5}$$

We also note that each mismatch term is computable from values stored during the signing phase, since for every $i \in \mathcal{I}_{\mathsf{id}}$, the tuple $(\cdot, \ell^*_i, m_i, \cdot)$ is recorded in L_σ.

Claim 5.2. *There exists at least one identity* $\mathsf{id} \in \mathcal{P}^*$ *and rank component* $r \in [R]$ *such that* $\delta^{(u)*}_{\mathsf{id},r} \neq 0$ *or* $\delta^{(v)*}_{\mathsf{id},r} \neq 0$.

Proof of Claim 5.2. Suppose for contradiction that for every identity $\mathsf{id} \in \mathcal{P}^*$ and every rank component $r \in [R]$, $\delta^{(u)}_{\mathsf{id},r} = 0$ and $\delta^{(v)}_{\mathsf{id},r} = 0$, *i.e.*,

$$\mu^{(u)*}_{\mathsf{id},r} = \sum_{i\in\mathcal{I}_{\mathsf{id}}} u^*_{i,r} m_i \quad \text{and} \quad \mu^{(v)*}_{\mathsf{id},r} = \sum_{i\in\mathcal{I}_{\mathsf{id}}} v^*_{i,r} m_i.$$

For any rank index r, summing $\mu^{(u)*}_{\mathsf{id},r}$ over id yields

$$\mu^{(u)*}_r = \sum_{\mathsf{id}\in\mathcal{P}^*} \mu^{(u)*}_{\mathsf{id},r} = \sum_{\mathsf{id}\in\mathcal{P}^*} \sum_{i\in\mathcal{I}_{\mathsf{id}}} u^*_{i,r} m_i = \sum_{i=1}^{n} u^*_{i,r} m_i = \mu^{(u)\circ}_r.$$

By symmetry, we also get $\mu^{(v)*}_r = \mu^{(v)\circ}_r$. Therefore, for every $r \in [R]$, we have $\mu^{(u)*}_r \mu^{(v)*}_r = \mu^{(u)\circ}_r \mu^{(v)\circ}_r$, and summing over r yields

$$\sum_{r=1}^{R} \mu^{(u)*}_r \mu^{(v)*}_r = \sum_{r=1}^{R} \mu^{(u)\circ}_r \mu^{(v)\circ}_r,$$

contradicting the Type-2 quadratic forgery case requirement from (5.4). Therefore, there exists at least one identity $\mathsf{id} \in \mathcal{P}^*$ and rank component $r \in [R]$ such that $\delta^{(u)*}_{\mathsf{id},r} \neq 0$ or $\delta^{(v)*}_{\mathsf{id},r} \neq 0$.

□

Using the mismatch terms, we also define the set of *consistent* identities as

$$\mathcal{C} := \left\{ \mathsf{id} \in \mathcal{P} \;\middle|\; \forall r \in [R] : \left(\delta^{(u)}_{\mathsf{id},r}, \delta^{(v)}_{\mathsf{id},r}\right) = (0,0) \right\}.$$

By Claim 5.2, there exists some $\mathsf{id} \in \mathcal{P}$ such that $\mathsf{id} \notin \mathcal{C}$. Hence, the set of *inconsistent* identities $\mathcal{P} \setminus \mathcal{C}$ is non-empty.

Finding an expression for Γ^*_ρ. Continuing with the extraction in the quadratic forgery setting, $\mathcal{B}$ samples

$$\left((\rho_1, \ldots, \rho_R)^\top, (\rho'_1, \ldots, \rho'_R)^\top\right) \xleftarrow{\$} (\mathbb{Z}^*_q)^{R\times 2}$$

as in line 11 of mkqhs-br.Verify from Figure 5.2, and calculates

$$\Gamma^*_\rho = \prod_{r=1}^{R} (\gamma_r^{(u)*})^{\rho_r} (\gamma_r^{(v)*})^{\rho'_r}.$$

For each identity $\mathsf{id} \in \mathcal{P}^*$, we define the placeholder variable

$$\hat{A}_{\mathsf{id}} := \tilde{g}_1^{\sum_{r=1}^{R} \left(\rho_r \mu^{(u)*}_{\mathsf{id},r} + \rho'_r \mu^{(v)*}_{\mathsf{id},r}\right)} \cdot \prod_{i \in \mathcal{I}_{\mathsf{id}}} \mathcal{H}(\ell_i^*)^{\sum_{r=1}^{R} \left(\rho_r u^*_{i,r} + \rho'_r v^*_{i,r}\right)}. \tag{5.6}$$

From the verification equation on line 12 in mkqhs-br.Verify, and using that $\mathsf{pk}_{\mathsf{id}} = X_2^{y_{\mathsf{id}}}$, we obtain

$$e(\Gamma^*_\rho, g_2) = \prod_{\mathsf{id} \in \mathcal{P}^*} e(\hat{A}_{\mathsf{id}}, X_2^{y_{\mathsf{id}}}).$$

Using bilinearity and $X_2 = g_2^x$, this becomes

$$\begin{aligned} e(\Gamma^*_\rho, g_2) &= \prod_{\mathsf{id} \in \mathcal{P}^*} e(\hat{A}_{\mathsf{id}}, g_2^{x y_{\mathsf{id}}}) \\ &= \prod_{\mathsf{id} \in \mathcal{P}^*} e(\hat{A}_{\mathsf{id}}^{x y_{\mathsf{id}}}, g_2) \\ &= e\left(\prod_{\mathsf{id} \in \mathcal{P}^*} \hat{A}_{\mathsf{id}}^{x y_{\mathsf{id}}}, g_2 \right), \end{aligned}$$

and by the non-degeneracy of the pairing,

$$\Gamma^*_\rho = \prod_{\mathsf{id} \in \mathcal{P}^*} \hat{A}_{\mathsf{id}}^{x y_{\mathsf{id}}}. \tag{5.7}$$

Substituting the programmed hashes. Next, we expand each $\hat{A}_{\mathsf{id}}$ from (5.6), using $\tilde{g}_1 = g_1^s h$ and the programmed hash values $\mathcal{H}(\ell) = g_1^t h^{-m}$ from (4.1):

$$\begin{aligned} \hat{A}_{\mathsf{id}} &= (g_1^s h)^{\sum_{r=1}^{R} \left(\rho_r \mu^{(u)*}_{\mathsf{id},r} + \rho'_r \mu^{(v)*}_{\mathsf{id},r}\right)} \cdot \prod_{i \in \mathcal{I}_{\mathsf{id}}} (g_1^{t_i} h^{-m_i})^{\sum_{r=1}^{R} \left(\rho_r u^*_{i,r} + \rho'_r v^*_{i,r}\right)} \\ &= g_1^{s \sum_{r=1}^{R} \left(\rho_r \mu^{(u)*}_{\mathsf{id},r} + \rho'_r \mu^{(v)*}_{\mathsf{id},r}\right) + \sum_{i \in \mathcal{I}_{\mathsf{id}}} t_i \sum_{r=1}^{R} \left(\rho_r u^*_{i,r} + \rho'_r v^*_{i,r}\right)} \\ &\quad \cdot h^{\sum_{r=1}^{R} \left(\rho_r \mu^{(u)*}_{\mathsf{id},r} + \rho'_r \mu^{(v)*}_{\mathsf{id},r}\right) - \sum_{i \in \mathcal{I}_{\mathsf{id}}} m_i \sum_{r=1}^{R} \left(\rho_r u^*_{i,r} + \rho'_r v^*_{i,r}\right)}. \end{aligned}$$

By rearranging the variables and comparing with the mismatch terms from Equation (5.5), we can rewrite the exponent of h as

$$\sum_{r=1}^{R} \left(\rho_r \delta^{(u)}_{\mathsf{id},r} + \rho'_r \delta^{(v)}_{\mathsf{id},r}\right).$$

We also define the corresponding exponent of g_1 for identity id as

$$\hat{a}_{\mathsf{id}} := s \sum_{r=1}^{R} \left(\rho_r \mu^{(u)*}_{\mathsf{id},r} + \rho'_r \mu^{(v)*}_{\mathsf{id},r}\right) + \sum_{i \in \mathcal{I}_{\mathsf{id}}} t_i \sum_{r=1}^{R} \left(\rho_r u^*_{i,r} + \rho'_r v^*_{i,r}\right),$$

computable by $\mathcal{B}$ from the stored values t_i from $(\ell_i^*, \mathcal{H}(\ell_i^*), \cdot, t_i) \in L_H$ and s from the setup phase. Hence,

$$\hat{A}_{\mathsf{id}} = g_1^{\hat{a}_{\mathsf{id}}} \cdot h^{\sum_{r=1}^{R}\left(\rho_r \delta_{\mathsf{id},r}^{(u)} + \rho_r' \delta_{\mathsf{id},r}^{(v)}\right)}.$$

Extracting the co-CDH* solution. For identities $\mathsf{id} \in \mathcal{C}$, $\delta_{\mathsf{id},r}^{(u)} = \delta_{\mathsf{id},r}^{(v)} = 0$, and the h-term becomes $1_{\mathbb{G}_1}$. Splitting (5.7) into consistent and inconsistent identities yields

$$\begin{aligned}
\Gamma_\rho^* &= \prod_{\mathsf{id}\in\mathcal{P}^*} \left(g_1^{\hat{a}_{\mathsf{id}}} h^{\sum_{r=1}^{R}\left(\rho_r \delta_{\mathsf{id},r}^{(u)} + \rho_r' \delta_{\mathsf{id},r}^{(v)}\right)} \right)^{x y_{\mathsf{id}}} \\
&= \prod_{\mathsf{id}\in\mathcal{C}} \left(g_1^{\hat{a}_{\mathsf{id}}} \right)^{x y_{\mathsf{id}}} \cdot \prod_{\mathsf{id}\in(\mathcal{P}^*\setminus\mathcal{C})} \left(g_1^{\hat{a}_{\mathsf{id}}} h^{\sum_{r=1}^{R}\left(\rho_r \delta_{\mathsf{id},r}^{(u)} + \rho_r' \delta_{\mathsf{id},r}^{(v)}\right)} \right)^{x y_{\mathsf{id}}} \\
&= \prod_{\mathsf{id}\in\mathcal{P}^*} \left(g_1^{\hat{a}_{\mathsf{id}}} \right)^{x y_{\mathsf{id}}} \cdot \prod_{\mathsf{id}\in(\mathcal{P}^*\setminus\mathcal{C})} \left(h^{\sum_{r=1}^{R}\left(\rho_r \delta_{\mathsf{id},r}^{(u)} + \rho_r' \delta_{\mathsf{id},r}^{(v)}\right)} \right)^{x y_{\mathsf{id}}} \\
&= \prod_{\mathsf{id}\in\mathcal{P}^*} X_1^{\hat{a}_{\mathsf{id}} y_{\mathsf{id}}} \cdot (h^x)^{\sum_{\mathsf{id}\in(\mathcal{P}^*\setminus\mathcal{C})} y_{\mathsf{id}} \sum_{r=1}^{R}\left(\rho_r \delta_{\mathsf{id},r}^{(u)} + \rho_r' \delta_{\mathsf{id},r}^{(v)}\right)}.
\end{aligned}$$

Next, we define the values

$$\hat{B} := \prod_{\mathsf{id}\in\mathcal{P}^*} X_1^{y_{\mathsf{id}} \hat{a}_{\mathsf{id}}},$$

and

$$\hat{c} := \sum_{\mathsf{id}\in(\mathcal{P}^*\setminus\mathcal{C})} y_{\mathsf{id}} \sum_{r=1}^{R} \left(\rho_r \delta_{\mathsf{id},r}^{(u)} + \rho_r' \delta_{\mathsf{id},r}^{(v)}\right),$$

which are both computable by $\mathcal{B}$. From the expression for Γ_ρ^* derived above, we obtain $\Gamma_\rho^* = \hat{B} \cdot (h^x)^{\hat{c}}$. Hence,

$$\frac{\Gamma_\rho^*}{\hat{B}} = (h^x)^{\hat{c}}.$$

If $\hat{c} = 0$ the reduction aborts. Otherwise, since $\hat{c} \neq 0$, $\mathcal{B}$ computes and outputs

$$\omega := \left(\frac{\Gamma_\rho^*}{\hat{B}}\right)^{\hat{c}^{-1}} = h^x,$$

which solves the co-CDH* instance.

Bounding the abort probability. It remains to bound the probability of abort, *i.e.*, the probability that $\hat{c} = 0$. We may view $\hat{c}$ as a polynomial of degree at most 2 in the random variables $\rho_1, \rho_1', \ldots, \rho_r, \rho_r' \xleftarrow{\$} \mathbb{Z}_q^*$ and $\{y_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}^*}$, with $y_{\mathsf{id}} \xleftarrow{\$} \mathbb{Z}_q^*$.

We show that $\hat{c}$ is not the zero polynomial. By Claim 5.2, there exists some $\mathsf{id} \notin \mathcal{C}$ and some $r \in [R]$ such that $\delta_{\mathsf{id},r}^{(u)} \neq 0$ or $\delta_{\mathsf{id},r}^{(v)} \neq 0$. If $\delta_{\mathsf{id},r}^{(u)} \neq 0$, then the monomial $y_{\mathsf{id}}\rho_r$ appears in $\hat{c}$ with non-zero coefficient $\delta_{\mathsf{id},r}^{(u)}$. By symmetry, if $\delta_{\mathsf{id},r}^{(v)} \neq 0$, then the monomial $y_{\mathsf{id}}\rho_r'$ appears with non-zero coefficient $\delta_{\mathsf{id},r}^{(v)}$. Hence, at least one coefficient

is non-zero, so $\hat{c}$ is not the zero polynomial. Therefore, by the Schwartz–Zippel lemma with $|\mathbb{Z}_q^*| = q-1$,

$$\Pr[\hat{c} = 0] \leq \frac{2}{q-1}.$$

Concluding, our reduction transforms Type-2 forgeries into a solution to the co-CDH* problem unless it aborts, which gives us

$$\mathsf{Adv}_{\mathcal{B}}^{\text{co-CDH}^*}(\lambda) \;\geq\; \left(1 - \frac{2}{q-1}\right) \cdot \Pr[\mathcal{A} \text{ outputs a valid Type-2 forgery}].$$

5.7.2 co-CDH* Extraction From Type-3 Forgeries

Let $\mathcal{A}$, after its interaction with $\mathcal{B}$, output a successful Type-3 forgery

$$(\mathcal{P}^*, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}^*}, \tilde{m}^*, \tilde{\sigma}^*), \quad \text{where } \mathcal{P}^* = (f^*, \ell_1^*, \ldots, \ell_n^*) \text{ and } f^* = \Big(\{a_i^*, \mathbf{u}_i^*, \mathbf{v}_i^*\}_{i=1}^n\Big).$$

We partition the labels in $\mathcal{P}^*$ into those that were signed during the game and those that were not. Concretely, we define

$$\mathcal{U} := \{i \in [n] \mid (\cdot, \ell_i^*, \cdot, \cdot) \notin L_\sigma\},$$

as the set of indices corresponding to *unsigned* labels in $\mathcal{P}^*$. Since the forgery is Type-3, we have $\mathcal{U} \neq \varnothing$. For each $\mathsf{id} \in \mathcal{P}^*$, we also let $\mathcal{I}_{\mathsf{id}} = \{\, i \in [n] \mid \ell_i^* = (\mathsf{id}, \cdot)\,\}$ and $\mathcal{U}_{\mathsf{id}} := \mathcal{I}_{\mathsf{id}} \cap \mathcal{U}$.

By the admissibility of f^* stated at the start of this chapter, every index $i \in \mathcal{U}$ satisfies either $a_i^* \neq 0$ or $(u_{i,r}^*, v_{i,r}^*) \neq (0,0)$ for some $r \in [R]$. Hence, exactly one of the following two cases holds for every Type-3 forgery:

Case 1: linear forgery. For every $i \in \mathcal{U}$, $a_i^* \neq 0$. In this case, $\mathcal{B}$ extracts the co-CDH* solution from $\gamma^{(a)*}$ using the Type-3 extraction of Theorem 4.1, applied with $\mu_{\mathsf{id}}^{(a)*}$ in the role of $\tilde{\mu}_{\mathsf{id}}^*$. The condition $a_i^* \neq 0$ for every $i \in \mathcal{U}$ matches the admissibility criterion in Theorem 4.1, and the extraction of h^x is identical. Hence,

$$\mathsf{Adv}_{\mathcal{B}}^{\text{co-CDH}^*}(\lambda) \;\geq\; \left(1 - \frac{1}{q}\right) \cdot \Pr[\mathcal{A} \text{ outputs a valid Type-3 forgery in Case 1}].$$

Case 2: quadratic forgery. There exists some $i \in \mathcal{U}$ with $a_i^* = 0$. By admissibility, this index i is guaranteed to satisfy $(u_{i,r}^*, v_{i,r}^*) \neq (0,0)$ for some $r \in [R]$. The rest of this proof proceeds with extraction in this case.

We now show how $\mathcal{B}$ extracts the co-CDH* solution from Γ_ρ^* on line 11 of mkqhs-br.Verify from Figure 5.2 in the quadratic forgery case. To ease the notation, we write

$$\begin{aligned} \theta_i^* &:= \langle \boldsymbol{\rho}, \mathbf{u}_i^* \rangle + \langle \boldsymbol{\rho}', \mathbf{v}_i^* \rangle = \sum_{r=1}^{R} \Big(\rho_r u_{i,r}^* + \rho_r' v_{i,r}^*\Big), \\ \Theta_{\mathsf{id}}^* &:= \langle \boldsymbol{\rho}, \boldsymbol{\mu}_{\mathsf{id}}^{(u)*} \rangle + \langle \boldsymbol{\rho}', \boldsymbol{\mu}_{\mathsf{id}}^{(v)*} \rangle = \sum_{r=1}^{R} \Big(\rho_r \mu_{\mathsf{id},r}^{(u)*} + \rho_r' \mu_{\mathsf{id},r}^{(v)*}\Big). \end{aligned} \tag{5.8}$$

For each $\mathsf{id} \in \mathcal{P}^*$, we define the placeholder value

$$\hat{A}_{\mathsf{id}} := \tilde{g}_1^{\Theta^*_{\mathsf{id}}} \cdot \prod_{i \in \mathcal{I}_{\mathsf{id}}} \mathcal{H}(\ell_i^*)^{\theta_i^*}. \tag{5.9}$$

Following the same derivation as in the Type-2 proof, the verification check ver_3 with $\mathsf{pk}_{\mathsf{id}} = X_2^{y_{\mathsf{id}}} = g_2^{x y_{\mathsf{id}}}$, and both bilinearity and non-degeneracy of the pairing yields

$$\Gamma^*_\rho = \prod_{\mathsf{id} \in \mathcal{P}^*} \hat{A}_{\mathsf{id}}^{x y_{\mathsf{id}}}. \tag{5.10}$$

Substituting the programmed hashes. For all $i \in \mathcal{I}_{\mathsf{id}} \setminus \mathcal{U}_{\mathsf{id}}$, we recall that the hash values were programmed during the sign query phase as $\mathcal{H}(\ell_i^*) = g_1^{t_i} h^{-m_i}$ from (4.1), where m_i is the message signed for label ℓ_i^*. For $i \in \mathcal{U}_{\mathsf{id}}$, the hash values were instead programmed in the delayed hash phase, or sampled for the first time at verification, as $\mathcal{H}(\ell_i^*) = h^{r_i}$.

Substituting these hashes and $\tilde{g}_1 = g_1^s h$ into (5.9) yields,

$$\begin{aligned}
\hat{A}_{\mathsf{id}} &= (g_1^s h)^{\Theta^*_{\mathsf{id}}} \cdot \prod_{i \in \mathcal{I}_{\mathsf{id}} \setminus \mathcal{U}_{\mathsf{id}}} (g_1^{t_i} h^{-m_i})^{\theta_i^*} \cdot \prod_{i \in \mathcal{U}_{\mathsf{id}}} (h^{r_i})^{\theta_i^*} \\
&= g_1^{s\Theta^*_{\mathsf{id}} + \sum_{i \in \mathcal{I}_{\mathsf{id}} \setminus \mathcal{U}_{\mathsf{id}}} \theta_i^* t_i} \cdot h^{\Theta^*_{\mathsf{id}} - \sum_{i \in \mathcal{I}_{\mathsf{id}} \setminus \mathcal{U}_{\mathsf{id}}} \theta_i^* m_i + \sum_{i \in \mathcal{U}_{\mathsf{id}}} \theta_i^* r_i}.
\end{aligned}$$

We therefore define the per-identity exponents

$$\begin{aligned}
\hat{a}_{\mathsf{id}} &:= s\Theta^*_{\mathsf{id}} + \sum_{i \in \mathcal{I}_{\mathsf{id}} \setminus \mathcal{U}_{\mathsf{id}}} \theta_i^* t_i, \\
\hat{b}_{\mathsf{id}} &:= \Theta^*_{\mathsf{id}} - \sum_{i \in \mathcal{I}_{\mathsf{id}} \setminus \mathcal{U}_{\mathsf{id}}} \theta_i^* m_i + \sum_{i \in \mathcal{U}_{\mathsf{id}}} \theta_i^* r_i,
\end{aligned} \tag{5.11}$$

giving $\hat{A}_{\mathsf{id}} = g_1^{\hat{a}_{\mathsf{id}}} h^{\hat{b}_{\mathsf{id}}}$. Both exponents are computable by $\mathcal{B}$, since s is sampled during setup and t_i, r_i are recorded in L_H.

Extracting the co-CDH* solution. Substituting $\hat{A}_{\mathsf{id}} = g_1^{\hat{a}_{\mathsf{id}}} h^{\hat{b}_{\mathsf{id}}}$ into (5.10) gives

$$\begin{aligned}
\Gamma^*_\rho &= \prod_{\mathsf{id} \in \mathcal{P}^*} \left(g_1^{\hat{a}_{\mathsf{id}}} h^{\hat{b}_{\mathsf{id}}} \right)^{x y_{\mathsf{id}}} \\
&= \prod_{\mathsf{id} \in \mathcal{P}^*} X_1^{y_{\mathsf{id}} \hat{a}_{\mathsf{id}}} \cdot (h^x)^{\sum_{\mathsf{id} \in \mathcal{P}^*} y_{\mathsf{id}} \hat{b}_{\mathsf{id}}}.
\end{aligned}$$

We define

$$\hat{B} := \prod_{\mathsf{id} \in \mathcal{P}^*} X_1^{y_{\mathsf{id}} \hat{a}_{\mathsf{id}}} \quad \text{and} \quad \hat{c} := \sum_{\mathsf{id} \in \mathcal{P}^*} y_{\mathsf{id}} \hat{b}_{\mathsf{id}}, \tag{5.12}$$

both computable by $\mathcal{B}$ with y_{id} stored in L_{ID}. Then

$$\Gamma^*_\rho = \hat{B} \cdot (h^x)^{\hat{c}}.$$

If $\hat{c} = 0$, $\mathcal{B}$ aborts. Otherwise, since $\hat{c} \neq 0$, $\mathcal{B}$ computes and outputs

$$\omega := \left(\frac{\Gamma^*_\rho}{\hat{B}} \right)^{\hat{c}^{-1}} = h^x,$$

which solves the co-CDH* instance.

Bounding the abort probability. Expanding $\hat{c}$ from (5.12) using (5.8) and (5.11) yields

$$\hat{c} = \sum_{\mathsf{id}\in\mathcal{P}^*} y_{\mathsf{id}} \Bigg(\underbrace{\sum_{r=1}^{R} \Big(\rho_r \mu^{(u)*}_{\mathsf{id},r} + \rho'_r \mu^{(v)*}_{\mathsf{id},r} \Big)}_{\Theta^*_{\mathsf{id}}} - \sum_{i\in\mathcal{I}_{\mathsf{id}}\setminus\mathcal{U}_{\mathsf{id}}} \Big(\underbrace{\sum_{r=1}^{R} \Big(\rho_r u^*_{i,r} + \rho'_r v^*_{i,r} \Big)}_{\theta^*_i} \Big) m_i + \sum_{i\in\mathcal{U}_{\mathsf{id}}} \Big(\underbrace{\sum_{r=1}^{R} \Big(\rho_r u^*_{i,r} + \rho'_r v^*_{i,r} \Big)}_{\theta^*_i} \Big) r_i \Bigg).$$

We bound the probability that $\hat{c} = 0$ by conditioning successively on the random variables $(y_{\mathsf{id}})_{\mathsf{id}\in\mathcal{P}^*} \xleftarrow{\$} (\mathbb{Z}_q^*)^{|\mathsf{id}\in\mathcal{P}^*|}$, $(\rho_r, \rho'_r)_{r\in[R]} \xleftarrow{\$} (\mathbb{Z}_q^*)^{2R}$, and $(r_i)_{i\in\mathcal{U}} \xleftarrow{\$} \mathbb{Z}_q^{|\mathcal{U}|}$.

First, we fix some $\bar{\imath} \in \mathcal{U}$ with $a^*_{\bar{\imath}} = 0$, and by the Case 2 guarantee, we may also fix some $\bar{r} \in [R]$ with $(u^*_{\bar{\imath},\bar{r}}, v^*_{\bar{\imath},\bar{r}}) \neq (0,0)$. Let $\bar{\mathsf{id}}$ be an identity such that $\ell^*_{\bar{\imath}} = (\bar{\mathsf{id}}, \cdot)$ and $\bar{\imath} \in \mathcal{U}_{\bar{\mathsf{id}}}$, which also guarantees the existence of the random variable $r_{\bar{\imath}}$ in L_H.

Viewed as a linear polynomial in $(\rho_r, \rho'_r)_{r\in[R]} \xleftarrow{\$} (\mathbb{Z}_q^*)^{2R}$, the inner sum

$$\theta^*_{\bar{\imath}} = \sum_{r=1}^{R} \Big(\rho_r u^*_{\bar{\imath},r} + \rho'_r v^*_{\bar{\imath},r} \Big)$$

has at least one non-zero coefficient by our choice of $(\bar{\imath}, \bar{r})$, namely either $u^*_{\bar{\imath},\bar{r}}$ as the coefficient of $\rho_{\bar{r}}$ or $v^*_{\bar{\imath},\bar{r}}$ as the coefficient of $\rho'_{\bar{r}}$. Hence, this inner sum is not the zero polynomial, and by the Schwartz–Zippel lemma with $|\mathbb{Z}_q^*| = q-1$,

$$\Pr[\theta^*_{\bar{\imath}} = 0] \leq \frac{1}{q-1}.$$

We now condition on the event $\theta^*_{\bar{\imath}} \neq 0$ and fix $(y_{\mathsf{id}})_{\mathsf{id}\in\mathcal{P}^*}$ and $(\rho_r, \rho'_r)_{r\in[R]}$ to arbitrary outcomes such that this event holds. Viewed this way, $\hat{c}$ is a linear polynomial in $(r_i)_{i\in\mathcal{U}}$ sampled from $\mathbb{Z}_q$, and the coefficient of $r_{\bar{\imath}}$ is $y_{\bar{\mathsf{id}}}\theta^*_{\bar{\imath}} \neq 0$ since $y_{\bar{\mathsf{id}}} \xleftarrow{\$} \mathbb{Z}_q^*$. Hence, $\hat{c}$ is not the zero polynomial, and by the Schwartz–Zippel lemma with $|\mathbb{Z}_q| = q$,

$$\Pr[\hat{c} = 0 \mid \theta^*_{\bar{\imath}} \neq 0] \leq \frac{1}{q}.$$

Finally, by the trivial bound $\Pr[\hat{c} = 0 \mid \theta^*_{\bar{\imath}} = 0] \leq 1$ and the law of total probability (Theorem 2.17), we get

$$\begin{aligned}\Pr[\hat{c} = 0] &= \Pr[\theta^*_{\bar{\imath}} \neq 0] \cdot \Pr[\hat{c} = 0 \mid \theta^*_{\bar{\imath}} \neq 0] \;+\; \Pr[\theta^*_{\bar{\imath}} = 0] \cdot \Pr[\hat{c} = 0 \mid \theta^*_{\bar{\imath}} = 0] \\ &\leq \left(1 - \frac{1}{q-1}\right) \cdot \frac{1}{q} + \frac{1}{q-1} \cdot 1 = \frac{2}{q}.\end{aligned}$$

Consequently,

$$\mathsf{Adv}^{\text{co-CDH}^*}_{\mathcal{B}}(\lambda) \;\geq\; \left(1 - \frac{2}{q}\right) \cdot \Pr[\mathcal{A} \text{ outputs a valid Type-3 forgery in Case 2}].$$

5.7.3 Combining the Bounds

We now combine the bounds from the previous subsections into a single expression for the total advantage $\mathsf{Adv}^{\mathsf{HomUF\text{-}CMA\text{-}DHQ}}_{\mathcal{A},\,\mathsf{mkqhs\text{-}br}}(\lambda)$.

By the same argument as in the proof of Theorem 4.1, Type-1 forgeries are also disregarded in the advantage bound. The advantage of the adversary, after applying the union bound (Theorem 2.18), therefore satisfies

$$\mathsf{Adv}^{\mathsf{HomUF\text{-}CMA\text{-}DHQ}}_{\mathcal{A},\,\mathsf{mkqhs\text{-}br}}(\lambda) \leq \Pr[\text{Type-2 forgery}] + \Pr[\text{Type-3 forgery}].$$

For each forgery type, it is either in the linear or the quadratic case, so applying the union bound once more yields

$$\begin{aligned}\Pr[\text{Type-2 forgery}] &\leq \Pr[\text{Type-2, Case 1}] + \Pr[\text{Type-2, Case 2}],\\ \Pr[\text{Type-3 forgery}] &\leq \Pr[\text{Type-3, Case 1}] + \Pr[\text{Type-3, Case 2}].\end{aligned}$$

Substituting the bounds derived in the previous subsections yields

$$\begin{aligned}\mathsf{Adv}^{\mathsf{HomUF\text{-}CMA\text{-}DHQ}}_{\mathcal{A},\,\mathsf{mkqhs\text{-}br}}(\lambda) &\leq \left(\frac{1}{1-\frac{1}{q-1}} + \frac{1}{1-\frac{2}{q-1}} + \frac{1}{1-\frac{1}{q}} + \frac{1}{1-\frac{2}{q}}\right) \cdot \mathsf{Adv}^{\text{co-CDH}^*}_{\mathcal{B}}(\lambda)\\ &= \left(\frac{q-1}{q-2} + \frac{q-1}{q-3} + \frac{q}{q-1} + \frac{q}{q-2}\right) \cdot \mathsf{Adv}^{\text{co-CDH}^*}_{\mathcal{B}}(\lambda).\end{aligned}$$

Since the sum of the four fractions is at most 5 for all primes $q \geq 11$, we conclude

$$\mathsf{Adv}^{\mathsf{HomUF\text{-}CMA\text{-}DHQ}}_{\mathcal{A},\,\mathsf{mkqhs\text{-}br}}(\lambda) \;\leq\; 5 \cdot \mathsf{Adv}^{\text{co-CDH}^*}_{\mathcal{B}}(\lambda).$$

This completes the proof of Theorem 5.1. □

6

Signature Compression and the Message-Square Extension

This chapter presents two extensions addressing different limitations of the baseline construction mkqhs-br from Chapter 5.

Section 6.1 introduces mkqhs-$\widetilde{\text{br}}$, a compressed variant of mkqhs-br that reduces the evaluated signature size from $O(tR)$ to $O(t + R)$. This makes the scheme practical for low-rank statistical approximations with $R > 1$ and many signers, where the multiplicative signature size of mkqhs-br would otherwise become prohibitive.

Section 6.2 introduces a *message-squares* extension of mkqhs-br we fittingly refer to as mkqhs-br-m^2. The extensions adds signing of m^2 alongside m in the signing algorithm, extending the supported polynomial class to include direct square terms $b_i m_i^2$ for $b_i \in \mathbb{Z}_q^*$. This enables evaluation of quadratic quantities, including variance and least-squares loss.

Section 6.3 uses that the two extensions are independent of each other and may therefore be applied simultaneously to mkqhs-br, resulting in a combined construction that inherits both the extended function class of mkqhs-br-m^2 and the compressed signature size of mkqhs-$\widetilde{\text{br}}$.

6.1 A Compressed Construction: mkqhs-$\widetilde{\text{br}}$

The baseline construction mkqhs-br achieves support for bounded-rank quadratic evaluation, but the size of its evaluated signatures grow as $O(tR)$. The high cost comes from the per-identity vectors $\boldsymbol{\mu}_{\mathsf{id}}^{(u)}, \boldsymbol{\mu}_{\mathsf{id}}^{(v)} \in \mathbb{Z}_q^R$, which together contribute $2tR$ scalar elements to the signature. For applications such as low-rank approximations of statistics over many signers, this multiplicative dependence on t and R may quickly result in very large evaluated signatures.

In this section, we present a compressed variant of mkqhs-br, we refer to as mkqhs-$\widetilde{\text{br}}$, and whose algorithms are illustrated in Figure 6.1. The mkqhs-$\widetilde{\text{br}}$ variant reduces the asymptotic evaluated signature size from $O(tR)$ to $O(t + R)$ compared to mkqhs-br, which we argue makes it more practical for low-rank statistical applications involving many distinct users, where the rank R is often larger.

The main change from mkqhs-br is that the verifier no longer needs the full per-

identity vectors $\boldsymbol{\mu}_{\mathsf{id}}^{(u)}, \boldsymbol{\mu}_{\mathsf{id}}^{(v)}$, since looking back at ver_3 in Figure 5.2, these vectors only appear in the inner products $\langle \boldsymbol{\rho}, \boldsymbol{\mu}_{\mathsf{id}}^{(u)} \rangle$ and $\langle \boldsymbol{\rho}', \boldsymbol{\mu}_{\mathsf{id}}^{(v)} \rangle$ using the verifiers random challenge $(\boldsymbol{\rho}, \boldsymbol{\rho}') \xleftarrow{\$} (\mathbb{Z}_q^*)^{R\times 2}$. We may therefore have Eval compute these inner products instead, given we can guarantee that the random challenges are fixed before the computation.

We achieve this by replacing the verifier-sampled challenge $(\boldsymbol{\rho}, \boldsymbol{\rho}') \xleftarrow{\$} (\mathbb{Z}_q^*)^{R\times 2}$ with a deterministic challenge $(\boldsymbol{\rho}, \boldsymbol{\rho}') \xleftarrow{\$} H_\rho(\cdot)$ computed from a hash function

$$H_\rho : \{0,1\}^* \to (\mathbb{Z}_q^*)^{R\times 2},$$

queried on public elements of the evaluated signature. Since both Eval and Verify have access to these public elements, both can derive the same challenge without any interaction.

The evaluator then compresses each pair $(\boldsymbol{\mu}_{\mathsf{id}}^{(u)}, \boldsymbol{\mu}_{\mathsf{id}}^{(v)})$ into a single scalar

$$\tilde{\mu}_{\mathsf{id}}^{(u,v)} \;:=\; \langle \boldsymbol{\rho}, \boldsymbol{\mu}_{\mathsf{id}}^{(u)} \rangle + \langle \boldsymbol{\rho}', \boldsymbol{\mu}_{\mathsf{id}}^{(v)} \rangle,$$

and, for each $\mathsf{id} \in \mathcal{P}$, only includes $\tilde{\mu}_{\mathsf{id}}^{(u,v)} \in \mathbb{Z}_q$ in place of the two length-R vectors $\boldsymbol{\mu}_{\mathsf{id}}^{(u)}, \boldsymbol{\mu}_{\mathsf{id}}^{(v)}$. The global aggregates $\boldsymbol{\mu}^{(u)} = \sum_{\mathsf{id}} \boldsymbol{\mu}_{\mathsf{id}}^{(u)}$ and $\boldsymbol{\mu}^{(v)} = \sum_{\mathsf{id}} \boldsymbol{\mu}_{\mathsf{id}}^{(v)}$ are still sent, as they are needed to recover the quadratic part of the claimed output $\tilde{m}$. Crucially, these R-vectors appear only once in the signature rather than once per identity, and therefore contribute only $O(R)$ group elements rather than $O(tR)$.

6.1.1 Changes From mkqhs-br

The Setup, KeyGen, and Sign algorithms of $\widetilde{\mathsf{mkqhs\text{-}br}}$ are identical to those of $\mathsf{mkqhs\text{-}br}$ in Figure 5.2. The modified Eval and Verify algorithms are given in Figure 6.1.

In $\widetilde{\mathsf{mkqhs\text{-}br}}$, the evaluated signature $\tilde{\sigma} = \left(\tilde{\gamma}, \{\tilde{\mu}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}, \boldsymbol{\mu}^{(u)}, \boldsymbol{\mu}^{(v)}\right)$ consists of the same group elements $\tilde{\gamma} = (\gamma^{(a)}, \boldsymbol{\gamma}^{(u)}, \boldsymbol{\gamma}^{(v)})$ as in $\mathsf{mkqhs\text{-}br}$, but with differing scalar elements. Indeed, $\tilde{\mu}_{\mathsf{id}} = (\mu_{\mathsf{id}}^{(a)}, \tilde{\mu}_{\mathsf{id}}^{(u,v)}) \in \mathbb{Z}_q^2$ for each $\mathsf{id} \in \mathcal{P}$, where the scalar $\tilde{\mu}_{\mathsf{id}}^{(u,v)}$ and global aggregates $\boldsymbol{\mu}^{(u)}, \boldsymbol{\mu}^{(v)} \in \mathbb{Z}_q^R$ replace the two per-identity vectors $\boldsymbol{\mu}_{\mathsf{id}}^{(u)}, \boldsymbol{\mu}_{\mathsf{id}}^{(v)} \in \mathbb{Z}_q^R$ used in $\mathsf{mkqhs\text{-}br}$. In total, the signature contains $2R+1$ group elements and $2t+2R$ scalar elements, giving overall size $O(t+R)$. When $R \le O(\log n)$, the evaluated signature size is bounded by $O(t+\log n)$ and the scheme remains succinct.

Verification now also proceeds in four checks instead of three.

Algebraic check (ver_1). This confirms that the claimed $\tilde{m}$ equals $\mu^{(a)} + \langle \boldsymbol{\mu}^{(u)}, \boldsymbol{\mu}^{(v)} \rangle$. This is the analogue of ver_1 in $\mathsf{mkqhs\text{-}br}$, but here the inner product is computed directly from the sent global vectors $\boldsymbol{\mu}^{(u)}, \boldsymbol{\mu}^{(v)}$ instead of being recovered from per-identity vectors $\boldsymbol{\mu}_{\mathsf{id}}^{(u)}, \boldsymbol{\mu}_{\mathsf{id}}^{(v)}$.

Linear pairing check (ver_2). Identical to ver_2 of $\mathsf{mkqhs\text{-}br}$, it verifies that the linear part of the evaluation was performed correctly against the public keys.

$\mathsf{Eval}(\mathcal{P}, \{\sigma_i\}_{i=1}^n)$

parse $\mathcal{P} = (f, \{\ell_i\}_{i=1}^n)$
parse $f = (\{a_i, \mathbf{u}_i, \mathbf{v}_i\}_{i=1}^n)$
parse $\mathbf{u}_i = (u_{i,1}, \ldots, u_{i,R})^\top$
parse $\mathbf{v}_i = (v_{i,1}, \ldots, v_{i,R})^\top$
parse $\sigma_i = (\mathsf{id}_i, \gamma_i, \mu_i)$
$\gamma^{(a)} = \prod_{i=1}^n \gamma_i^{a_i}$
for $r \in [R]$
  $\gamma_r^{(u)} = \prod_{i=1}^n \gamma_i^{u_{i,r}},\ \gamma_r^{(v)} = \prod_{i=1}^n \gamma_i^{v_{i,r}}$
$\boldsymbol{\gamma}^{(u)} = (\gamma_1^{(u)}, \ldots, \gamma_R^{(u)})^\top$
$\boldsymbol{\gamma}^{(v)} = (\gamma_1^{(v)}, \ldots, \gamma_R^{(v)})^\top$
$\tilde{\gamma} = \left(\gamma^{(a)}, \boldsymbol{\gamma}^{(u)}, \boldsymbol{\gamma}^{(v)}\right)$
for $\mathsf{id} \in \mathcal{P}$
  $\mathcal{I}_{\mathsf{id}} = \{i \in [n] \mid \ell_i = (\mathsf{id}, \cdot)\}$
  $\mu_{\mathsf{id}}^{(a)} = \sum_{i \in \mathcal{I}_{\mathsf{id}}} a_i \mu_i$
  $\boldsymbol{\mu}_{\mathsf{id}}^{(u)} = \sum_{i \in \mathcal{I}_{\mathsf{id}}} \mu_i \mathbf{u}_i,\ \boldsymbol{\mu}_{\mathsf{id}}^{(v)} = \sum_{i \in \mathcal{I}_{\mathsf{id}}} \mu_i \mathbf{v}_i$
$\boldsymbol{\mu}^{(u)} = \sum_{\mathsf{id} \in \mathcal{P}} \boldsymbol{\mu}_{\mathsf{id}}^{(u)},\ \boldsymbol{\mu}^{(v)} = \sum_{\mathsf{id} \in \mathcal{P}} \boldsymbol{\mu}_{\mathsf{id}}^{(v)}$
$(\boldsymbol{\rho}, \boldsymbol{\rho}') \leftarrow H_\rho\left(\mathcal{P}, \tilde{\gamma}, \{\mu_{\mathsf{id}}^{(a)}\}_{\mathsf{id} \in \mathcal{P}}, \boldsymbol{\mu}^{(u)}, \boldsymbol{\mu}^{(v)}\right)$
for $\mathsf{id} \in \mathcal{P}$
  $\tilde{\mu}_{\mathsf{id}}^{(u,v)} = \langle \boldsymbol{\rho}, \boldsymbol{\mu}_{\mathsf{id}}^{(u)} \rangle + \langle \boldsymbol{\rho}', \boldsymbol{\mu}_{\mathsf{id}}^{(v)} \rangle$
  $\tilde{\mu}_{\mathsf{id}} = \left(\mu_{\mathsf{id}}^{(a)}, \tilde{\mu}_{\mathsf{id}}^{(u,v)}\right)$
return $\tilde{\sigma} = \left(\tilde{\gamma}, \{\tilde{\mu}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}}, \boldsymbol{\mu}^{(u)}, \boldsymbol{\mu}^{(v)}\right)$

$\mathsf{Verify}(\mathcal{P}, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}}, \tilde{m}, \tilde{\sigma})$

parse $\mathcal{P} = (f, \{\ell_i\}_{i=1}^n),\ f = (\{a_i, \mathbf{u}_i, \mathbf{v}_i\}_{i=1}^n)$,
parse $\tilde{\sigma} = \left(\tilde{\gamma}, \{\tilde{\mu}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}}, \boldsymbol{\mu}^{(u)}, \boldsymbol{\mu}^{(v)}\right),\ \tilde{\gamma} = \left(\gamma^{(a)}, \boldsymbol{\gamma}^{(u)}, \boldsymbol{\gamma}^{(v)}\right)$
for $\mathsf{id} \in \mathcal{P}$
  parse $\tilde{\mu}_{\mathsf{id}} = \left(\mu_{\mathsf{id}}^{(a)}, \tilde{\mu}_{\mathsf{id}}^{(u,v)}\right)$
  $\mathcal{I}_{\mathsf{id}} = \{i \in [n] \mid \ell_i = (\mathsf{id}, \cdot)\}$
$(\boldsymbol{\rho}, \boldsymbol{\rho}') \leftarrow H_\rho\left(\mathcal{P}, \tilde{\gamma}, \{\mu_{\mathsf{id}}^{(a)}\}_{\mathsf{id} \in \mathcal{P}}, \boldsymbol{\mu}^{(u)}, \boldsymbol{\mu}^{(v)}\right)$
$\mu^{(a)} = \sum_{\mathsf{id} \in \mathcal{P}} \mu_{\mathsf{id}}^{(a)}$
$\mathsf{ver}_1 \leftarrow \left[\tilde{m} == \mu^{(a)} + \langle \boldsymbol{\mu}^{(u)}, \boldsymbol{\mu}^{(v)} \rangle\right]$
$\mathsf{ver}_2 \leftarrow \left[e(\gamma^{(a)}, g_2) == \prod_{\mathsf{id} \in \mathcal{P}} e\left(g_1^{\mu_{\mathsf{id}}^{(a)}} \cdot \prod_{i \in \mathcal{I}_{\mathsf{id}}} H(\ell_i)^{a_i}, \mathsf{pk}_{\mathsf{id}}\right)\right]$
$\Gamma_\rho = (\boldsymbol{\gamma}^{(u)})^{\boldsymbol{\rho}} (\boldsymbol{\gamma}^{(v)})^{\boldsymbol{\rho}'}$
$\mathsf{ver}_3 \leftarrow \left[e(\Gamma_\rho, g_2) == \prod_{\mathsf{id} \in \mathcal{P}} e\left(g_1^{\tilde{\mu}_{\mathsf{id}}^{(u,v)}} \cdot \prod_{i \in \mathcal{I}_{\mathsf{id}}} H(\ell_i)^{\langle \boldsymbol{\rho}, \mathbf{u}_i \rangle + \langle \boldsymbol{\rho}', \mathbf{v}_i \rangle}, \mathsf{pk}_{\mathsf{id}}\right)\right]$
$\mathsf{ver}_4 \leftarrow \left[\sum_{\mathsf{id} \in \mathcal{P}} \tilde{\mu}_{\mathsf{id}}^{(u,v)} == \langle \boldsymbol{\rho}, \boldsymbol{\mu}^{(u)} \rangle + \langle \boldsymbol{\rho}', \boldsymbol{\mu}^{(v)} \rangle\right]$
return $[\mathsf{ver}_1 \wedge \mathsf{ver}_2 \wedge \mathsf{ver}_3 \wedge \mathsf{ver}_4]$

Figure 6.1: The Eval and Verify algorithms for mkqhs-$\widetilde{\text{br}}$. Setup, KeyGen, and Sign are identical to mkqhs-br, given in in Figure 5.2. Green text indicates changes from Figure 5.2.

Quadratic pairing check (ver_3). This is the counterpart of ver_3 in mkqhs-br. It uses the same collapsed pairing equation in $\mathbb{G}_T$, but with the per-identity exponent of g_1 now coming directly from $\tilde{\mu}_{\mathsf{id}}^{(u,v)}$ instead of being reconstructed as $\langle \boldsymbol{\rho}, \boldsymbol{\mu}_{\mathsf{id}}^{(u)} \rangle + \langle \boldsymbol{\rho}', \boldsymbol{\mu}_{\mathsf{id}}^{(v)} \rangle$.

Compression-consistency check (ver_4). This is new to mkqhs-$\widetilde{\text{br}}$. It verifies that the compressed per-identity scalars $\tilde{\mu}_{\mathsf{id}}^{(u,v)}$ are consistent with both the global vectors $\boldsymbol{\mu}^{(u)}, \boldsymbol{\mu}^{(v)}$ and the random challenge $(\boldsymbol{\rho}, \boldsymbol{\rho}')$. Concretely, it checks that

$$\sum_{\mathsf{id} \in \mathcal{P}} \tilde{\mu}_{\mathsf{id}}^{(u,v)} \;=\; \langle \boldsymbol{\rho}, \boldsymbol{\mu}^{(u)} \rangle + \langle \boldsymbol{\rho}', \boldsymbol{\mu}^{(v)} \rangle.$$

Without this check, the evaluator could choose each $\tilde{\mu}_{\mathsf{id}}^{(u,v)}$ freely, decoupling the per-identity scalars from the global aggregates.

6.1.2 Sketch of Correctness

We sketch evaluation correctness and note that authentication correctness follows as the special case $n = 1$ with f given by the identity program $\mathcal{I}_\ell$ as in Section 5.5.

Checks ver_1, ver_2, and ver_3 in Figure 6.1 pass deterministically by the same arguments as in the correctness proof of mkqhs-br in Section 5.5. This is because the global vectors $\boldsymbol{\mu}^{(u)}, \boldsymbol{\mu}^{(v)}$ are honestly computed as the sums of the per-identity vectors $\{\boldsymbol{\mu}_{\mathsf{id}}^{(u)}, \boldsymbol{\mu}_{\mathsf{id}}^{(v)}\}_{\mathsf{id} \in \mathcal{P}}$, and the new per-identity quadratic aggregate is

$$\tilde{\mu}_{\mathsf{id}}^{(u,v)} = \langle \boldsymbol{\rho}, \boldsymbol{\mu}_{\mathsf{id}}^{(u)} \rangle + \langle \boldsymbol{\rho}', \boldsymbol{\mu}_{\mathsf{id}}^{(v)} \rangle.$$

The check ver_4 follows from the definition of $\tilde{\mu}_{\mathsf{id}}^{(u,v)}$ in mkqhs-$\widetilde{\text{br}}$.Eval, since substitution and rearrangement of the left hand side of ver_4 yields

$$\begin{aligned}
\sum_{\mathsf{id} \in \mathcal{P}} \tilde{\mu}_{\mathsf{id}}^{(u,v)} &= \sum_{\mathsf{id} \in \mathcal{P}} \Big(\langle \boldsymbol{\rho}, \boldsymbol{\mu}_{\mathsf{id}}^{(u)} \rangle + \langle \boldsymbol{\rho}', \boldsymbol{\mu}_{\mathsf{id}}^{(v)} \rangle \Big) \\
&= \sum_{\mathsf{id} \in \mathcal{P}} \sum_{r=1}^{R} \Big(\rho_r \mu_{\mathsf{id},r}^{(u)} + \rho'_r \mu_{\mathsf{id},r}^{(v)} \Big) \\
&= \sum_{r=1}^{R} \rho_r \, \mu_r^{(u)} + \sum_{r=1}^{R} \rho'_r \, \mu_r^{(v)} \\
&= \langle \boldsymbol{\rho}, \boldsymbol{\mu}^{(u)} \rangle + \langle \boldsymbol{\rho}', \boldsymbol{\mu}^{(v)} \rangle.
\end{aligned}$$

Lastly, correctness also requires mkqhs-$\widetilde{\text{br}}$.Eval and mkqhs-$\widetilde{\text{br}}$.Verify to derive the same challenge $(\boldsymbol{\rho}, \boldsymbol{\rho}')$, which holds because both compute it from H_ρ on the same inputs.

Since all four checks pass deterministically, the scheme satisfies both authentication and evaluation correctness.

6.1.3 Security Theorem and Proof Outline

Theorem 6.1. *Assuming that the co-CDH* problem is hard, the compressed scheme* $\widetilde{\mathsf{mkqhs\text{-}br}}$ *in Figure 6.1 is secure under* HomUF-CMA-DHQ *in the random oracle model. Formally, let* $\mathcal{A}$ *be a PPT adversary in the* HomUF-CMA-DHQ *security experiment. Then its advantage is bounded by*

$$\mathsf{Adv}^{\mathsf{HomUF\text{-}CMA\text{-}DHQ}}_{\mathcal{A},\,\widetilde{\mathsf{mkqhs\text{-}br}}}(\lambda) \le 5 \cdot \mathsf{Adv}^{\text{co-CDH}^*}_{\mathcal{B}}(\lambda),$$

where $\mathcal{B}$ *is a PPT algorithm that solves the co-CDH* problem with advantage* $\mathsf{Adv}^{\text{co-CDH}^*}_{\mathcal{B}}(\lambda)$.

The proof follows that of Theorem 5.1 closely, and we defer the full proof to Appendix A. The structure of the proof is similar, where forgeries are distinguished between linear and quadratic type, and each is reduced to extracting a co-CDH* solution from $\gamma^{(a)}$ or Γ_ρ respectively. The abort probabilities are bounded via the Schwartz–Zippel lemma.

The main differences relative to the mkqhs-br proof are:

Simulation of H_ρ. The hash H_ρ is modelled as a random oracle $\mathcal{H}_\rho$, returning uniformly random values in $(\mathbb{Z}_q^*)^{R\times 2}$ while maintaining consistency across repeated queries. Additionally, since H_ρ is not used inside Sign, $\mathcal{H}_\rho$ is not subject to the delayed-hash restriction of the HomUF-CMA-DHQ experiment. Since this is the only difference in the simulation, Claim 4.3 still applies and the simulation is indistinguishable from a real execution of HomUF-CMA-DHQ.

Type-2 Quadratic Forgeries. In the mkqhs-br proof of Theorem 5.1, extraction from Type-2 quadratic forgeries relies on the per-component mismatch terms $(\delta^{(u)}_{\mathsf{id},r}, \delta^{(v)}_{\mathsf{id},r})$. In $\widetilde{\mathsf{mkqhs\text{-}br}}$, the per-identity vectors are no longer part of the evaluated signature, so we instead define a single scalar mismatch per identity

$$\delta_{\mathsf{id}} := \tilde{\mu}^{(u,v)*}_{\mathsf{id}} - \sum_{r=1}^{R}\left(\rho_r \sum_{i\in\mathcal{I}_{\mathsf{id}}} u^*_{i,r} m_i + \rho'_r \sum_{i\in\mathcal{I}_{\mathsf{id}}} v^*_{i,r} m_i\right). \tag{6.1}$$

The ver_4 check then forces any inconsistency in the per-identity mismatch terms of (6.1) to propagate into the global aggregates $\boldsymbol{\mu}^{(u)*}, \boldsymbol{\mu}^{(v)*}$ with overwhelming probability. Since this is the main difference in the security proof, we provide its proof in Claim 6.2 below.

Type-3 Quadratic Forgeries. In the mkqhs-br proof of Theorem 5.1, extraction from Type-3 quadratic forgeries uses $\hat{A}_{\mathsf{id}}$ built from $\Theta^*_{\mathsf{id}} = \langle \boldsymbol{\rho}, \boldsymbol{\mu}^{(u)*}_{\mathsf{id}}\rangle + \langle \boldsymbol{\rho}', \boldsymbol{\mu}^{(v)*}_{\mathsf{id}}\rangle$. In $\widetilde{\mathsf{mkqhs\text{-}br}}$, the per-identity vectors are not part of the evaluated signature, but the compressed scalar $\tilde{\mu}^{(u,v)*}_{\mathsf{id}}$ plays exactly the same role as $\langle \boldsymbol{\rho}, \boldsymbol{\mu}^{(u)*}_{\mathsf{id}}\rangle + \langle \boldsymbol{\rho}', \boldsymbol{\mu}^{(v)*}_{\mathsf{id}}\rangle$. Hence, setting $\Theta^*_{\mathsf{id}} := \tilde{\mu}^{(u,v)*}_{\mathsf{id}}$ in the definition of $\hat{A}_{\mathsf{id}}$, $\hat{a}_{\mathsf{id}}$, and $\hat{b}_{\mathsf{id}}$, the extraction and abort-probability argument goes through word-for-word as in the Type-3 quadratic case of Theorem 5.1.

Claim 6.2. *In the quadratic Type-2 forgery case of* $\widetilde{\mathsf{mkqhs\text{-}br}}$*, there exists at least one identity* $\mathsf{id} \in \mathcal{P}^*$ *such that* $\delta_{\mathsf{id}} \neq 0$ *except with probability at most* $\frac{1}{q-1}$.

Proof of Claim 6.2. As in Theorem 5.1, we recall that a quadratic Type-2 forgery implies

$$\sum_{r=1}^{R} \mu_r^{(u)*} \mu_r^{(v)*} \neq \sum_{r=1}^{R} \mu_r^{(u)\circ} \mu_r^{(u)\circ}. \tag{6.2}$$

where we denote the honest aggregates

$$\mu_r^{(u)\circ} = \sum_{i=1}^{n} u_{i,r}^* m_i, \quad \mu_r^{(v)\circ} = \sum_{i=1}^{n} v_{i,r}^* m_i.$$

Assume for contradiction that $\delta_{\mathsf{id}} = 0$ for every $\mathsf{id} \in \mathcal{P}^*$. Summing over all identities gives

$$\sum_{\mathsf{id} \in \mathcal{P}^*} \delta_{\mathsf{id}} = 0,$$

and expanding using (6.1),

$$\begin{aligned} 0 &= \sum_{\mathsf{id} \in \mathcal{P}^*} \tilde{\mu}_{\mathsf{id}}^{(u,v)*} - \sum_{\mathsf{id} \in \mathcal{P}^*} \sum_{r=1}^{R} \left(\rho_r \sum_{i \in \mathcal{I}_{\mathsf{id}}} u_{i,r}^* m_i + \rho'_r \sum_{i \in \mathcal{I}_{\mathsf{id}}} v_{i,r}^* m_i \right) \\ &= \sum_{\mathsf{id} \in \mathcal{P}^*} \tilde{\mu}_{\mathsf{id}}^{(u,v)*} - \sum_{r=1}^{R} \left(\rho_r \sum_{\mathsf{id} \in \mathcal{P}^*} \sum_{i \in \mathcal{I}_{\mathsf{id}}} u_{i,r}^* m_i + \rho'_r \sum_{\mathsf{id} \in \mathcal{P}^*} \sum_{i \in \mathcal{I}_{\mathsf{id}}} v_{i,r}^* m_i \right) \\ &= \sum_{\mathsf{id} \in \mathcal{P}^*} \tilde{\mu}_{\mathsf{id}}^{(u,v)*} - \sum_{r=1}^{R} \rho_r \mu_r^{(u)\circ} + \rho'_r \mu_r^{(v)\circ}. \end{aligned} \tag{6.3}$$

Furthermore, since $\tilde{\sigma}^*$ verifies, the check ver_4 on line 12 in $\mathsf{mkqhs\text{-}\widetilde{br}.Verify}$ from Figure 6.1, implies

$$\sum_{\mathsf{id} \in \mathcal{P}} \tilde{\mu}_{\mathsf{id}}^{(u,v)} = \langle \boldsymbol{\rho}, \boldsymbol{\mu}^{(u)} \rangle + \langle \boldsymbol{\rho}', \boldsymbol{\mu}^{(v)} \rangle = \sum_{r=1}^{R} \rho_r \mu_r^{(u)*} + \rho'_r \mu_r^{(v)*}.$$

Substituting into (6.3) and rearranging, we obtain

$$\sum_{r=1}^{R} \rho_r \left(\mu_r^{(u)*} - \mu_r^{(u)\circ} \right) + \sum_{r=1}^{R} \rho'_r \left(\mu_r^{(v)*} - \mu_r^{(v)\circ} \right) = 0. \tag{6.4}$$

We may view the left hand side of equation (6.4) as a non-zero linear polynomial in the random variables $\{\rho_r, \rho'_r\}_{r=1}^{R}$ sampled uniformly from $\mathbb{Z}_q^*$ by $\mathcal{H}_\rho$. We also note that since $\{\mu_r^{(u)*}, \mu_r^{(v)*}\}_{r=1}^{R}$ appear as input to the random oracle $\mathcal{H}_\rho$, the adversary must commit to these values before the challenge $(\boldsymbol{\rho}, \boldsymbol{\rho}')$ is determined. The polynomial is therefore evaluated at a point that is uniformly random and independent of the adversary's forgery.

We claim that not all coefficients in this polynomial are zero. Indeed, if $\mu_r^{(u)*} = \mu_r^{(u)\circ}$ and $\mu_r^{(v)*} = \mu_r^{(v)\circ}$ for all $r \in [R]$, then

$$\sum_{r=1}^{R} \mu_r^{(u)*} \mu_r^{(v)*} = \sum_{r=1}^{R} \mu_r^{(u)\circ} \mu_r^{(v)\circ},$$

contradicting (6.2). Therefore, by the Schwartz–Zippel lemma, with $|\mathbb{Z}_q^*| = q - 1$ equation (6.4) holds with probability at most $1/(q-1)$. Since this bound comes from the assumption that $\delta_{\mathsf{id}} = 0$ for all $\mathsf{id} \in \mathcal{P}$, we get

$$\Pr\Big[\forall \mathsf{id} \in \mathcal{P}^* : \delta_{\mathsf{id}} = 0\Big] \leq \frac{1}{q-1},$$

which proves the claim. □

6.2 Succinct Message-Squares: mkqhs-br-m^2

Many practical use-cases of MKHS arise in statistics and machine learning, where support for quadratic polynomial evaluation is necessary for common computations such as variance and least-squares loss. We show how these examples point at a limitation of the bounded-rank quadratic class supported in Chapter 5, and subsequently address it by extending mkqhs-br.

6.2.1 Limitations of the Bounded Rank Function Class

We recall that the quadratic part of the evaluated polynomials in Chapter 5 and Section 6.1 had to be expressible as $\mathbf{m}^\top Q \mathbf{m}$ with $\operatorname{rank}(Q) \leq R = O(\log n)$. This captures low-rank cross terms, but fails at capturing sums of quadratic terms $\sum_{i=1}^n m_i^2$.

To give a concrete example, variance over the data points $m_1, \ldots, m_n$ is expressed as

$$\begin{aligned}\mathsf{Var}(m_1, \ldots, m_n) &= \frac{1}{n}\sum_{i=1}^n m_i^2 - \frac{1}{n^2}\left(\sum_{i=1}^n m_i\right)^2 \\ &= \mathbf{m}^\top \left(\frac{1}{n} I_n - \frac{1}{n^2}\mathbf{1}\mathbf{1}^\top\right)\mathbf{m} \\ &= \mathbf{m}^\top V_n \mathbf{m},\end{aligned}$$

where $\mathbf{1} = (1, \ldots, 1)^\top$. The matrix V_n has $\operatorname{rank}(V_n) \geq n - 1$ since $\operatorname{rank}(I_n) = n$ and $\operatorname{rank}(\mathbf{1}\mathbf{1}^\top) = 1$. Hence, the previous class of admissible functions also fail at variance calculations.

To handle these sorts of expressions while preserving succinctness, we modify mkqhs-br by also signing message-squares m_i^2 in Sign. We refer to the resulting construction as mkqhs-br-m^2.

6.2.2 The New Admissible Function Class

The message-squares extension has the effect of extending the supported polynomial class to explicitly include square terms, yielding polynomials of the form

$$f(m_1, \ldots, m_n) = \sum_{i=1}^n \left(a_i m_i + b_i m_i^2\right) + \sum_{r=1}^R \left(\sum_{i=1}^n u_{i,r} m_i\right)\left(\sum_{i=1}^n v_{i,r} m_i\right). \tag{6.5}$$

For the function class (6.5), we once again require that every label ℓ_i appearing in a labeled program $\mathcal{P}$ must contribute non-trivially to f, meaning that for each $\ell_i \in \mathcal{P}$, at least one of $(a_i, b_i) \neq (0, 0)$, or $(u_{i,r}, v_{i,r}) \neq (0, 0)$ for some $r \in [R]$ must hold. This is the natural extension of the non-triviality restriction imposed in Chapter 5, but now accounting for the additional square coefficient b_i.

6.2.3 Examples of Computable Statistical Functions

Table 6.1 shows how a range of practical statistical functions can be expressed in the new admissible function class (6.5), all with $R \leq 1$, confirming that succinctness is preserved.

Polynomial	**Function**	a_i	b_i	R	$u_{i,1}$	$v_{i,1}$
$\sum_{i=1}^{n} m_i^2$	Squared norm	0	1	0	–	–
$\frac{1}{n}\sum_{i=1}^{n} m_i^2 - \frac{1}{n^2}\left(\sum_{i=1}^{n} m_i\right)^2$	Variance	0	$\frac{1}{n}$	1	1	$-\frac{1}{n^2}$
$\sum_{i=1}^{n}(m_i - c_i)^2$	Euclidean distance	$-2c_i$	1	0	–	–
$\frac{1}{n}\sum_{i=1}^{n}(m_i - y_i)^2$	Least-squares loss (MSE)	$-\frac{2y_i}{n}$	$\frac{1}{n}$	0	–	–

Table 6.1: Representation of polynomial functions in the extended class (6.5). The vectors $\mathbf{c} = (c_1, \ldots, c_n)$ and $\mathbf{y} = (y_1, \ldots, y_n)$ are public so additive constants depending only on these are omitted as they are computable by the verifier.

An important statistical use case is evaluating the squared Euclidean distance between two d-dimensional vectors, $\mathbf{x}$ and $\mathbf{y}$. For $n = 2d$ messages, we define $\mathbf{x} = (m_1, \ldots, m_d)^\top$ and $\mathbf{y} = (m_{d+1}, \ldots, m_{2d})^\top$, yielding:

$$\|\mathbf{x} - \mathbf{y}\|^2 = \sum_{i=1}^{d}(m_i - m_{i+d})^2.$$

For $d = 2$, the quadratic part admits a rank one decomposition with the help of the square terms. Indeed, letting f be an admissible function from (6.5) with $a_i = 0$ for all i and

$$\mathbf{b} = (0, 2, 0, 2)^\top, \quad R = 1, \quad \mathbf{u}_1 = (1,\ 1,\ -1,\ 1)^\top, \quad \mathbf{v}_1 = (1,\ -1,\ -1,\ -1)^\top,$$

we may verify that

$$\begin{aligned} f(m_1, m_2, m_3, m_4) &= 2m_2^2 + 2m_4^2 + (m_1 + m_2 - m_3 + m_4)(m_1 - m_2 - m_3 - m_4) \\ &= m_1^2 + m_2^2 + m_3^2 + m_4^2 - 2m_1m_3 - 2m_2m_4 \\ &= (m_1 - m_3)^2 + (m_2 - m_4)^2 = \|\mathbf{x} - \mathbf{y}\|^2. \end{aligned} \tag{6.6}$$

We generalise this to arbitrary dimension d in the following proposition.

Proposition 6.3. *Let* $\mathbf{x} = (m_1, \ldots, m_d)^\top$ *and* $\mathbf{y} = (m_{d+1}, \ldots, m_{2d})^\top$ *be two* d*-dimensional vectors in* $\mathbb{Z}_q^d$*. Then their squared Euclidean distance*

$$\|\mathbf{x} - \mathbf{y}\|^2 = \sum_{i=1}^{d} (m_i - m_{i+d})^2 \tag{6.7}$$

lies in the polynomial class (6.5) *with minimal* R *satisfying* $R \leq \lceil d/2 \rceil$*. Such a representation with* $R = \lceil d/2 \rceil$ *is obtained by applying the coefficients in Table 6.2 to* (6.5)*.*

Squared Euclidean Distance: $\sum_{i=1}^{d} (m_i - m_{i+d})^2, \quad R = \lceil d/2 \rceil$

Case	**Coeff.**	**Index (i)**
For each $r \in \{1, \ldots, \lfloor d/2 \rfloor\}$	$b_i = 2$	$2r,\ 2r + d$
	$u_{i,r} = 1$	$2r - 1,\ 2r,\ 2r + d$
	$u_{i,r} = -1$	$2r - 1 + d$
	$v_{i,r} = 1$	$2r - 1$
	$v_{i,r} = -1$	$2r,\ 2r - 1 + d,\ 2r + d$
If d is odd ($R = \lceil d/2 \rceil \neq \lfloor d/2 \rfloor$)	$u_{i,R} = v_{i,R} = 1$	d
	$u_{i,R} = v_{i,R} = -1$	$2d$

Table 6.2: Coefficient assignment for the squared Euclidean distance expressed in the admissible function class (6.5). All unlisted coefficients are zero.

The proof of Proposition 6.3 is deferred to Appendix B.

We regard the ability to express squared Euclidean distance with small R as an indication of high expressivity for the mkqhs-br-m^2 function class. Succinctness is preserved for dimensions up to $d = 10$, where the resulting $n = 20$ and $R = 5$ satisfy the condition $R \leq \lceil \log_2 n \rceil$. However, for $d \geq 11$, the rank exceeds this threshold. We consider this supported dimension range sufficient for many practical applications. In particular, the ability to succinctly evaluate squared Euclidean distances allows for verifiable computation of pairwise distances, allowing applications in nearest neighbour classification and k-means clustering.

6.2.4 Construction

We now present the succinct *message-squares* variant of the mkqhs-br construction from Chapter 5. All changes stem from one modification where Sign also signs m^2 in addition to signing the message m.

Concretely, Setup and KeyGen remain unchanged, except that Setup now fixes two hash functions $H_1, H_2 : \{0,1\}^* \to \mathbb{G}_1$, as shown in Figure 6.2. Using these hashes, the Sign algorithm signs both m and m^2 under the same label:

$$\gamma = (H_1(\ell) \cdot g_1^{m})^{\mathsf{sk}_{\mathsf{id}}} \quad \text{and} \quad \gamma' = (H_2(\ell) \cdot g_1^{m^2})^{\mathsf{sk}_{\mathsf{id}}}.$$

Eval and Verify are then modified to handle γ' by treating it like an additional component in the linear signature aggregate $\gamma^{(a)}$ from mkqhs-br. Specifically, $\gamma^{(a)}$ is replaced with

$$\gamma^{(a,b)} = \prod_{i=1}^{n} \gamma_i^{a_i} (\gamma_i')^{b_i},$$

reflecting the computation $\sum_{i=1}^{n}(a_i m_i + b_i m_i^2)$, and likewise, $\mu_{\mathsf{id}}^{(a)}$ gets replaced with

$$\mu_{\mathsf{id}}^{(a,b)} = \sum_{i \in \mathcal{I}_{\mathsf{id}}} (a_i \mu_i + b_i \mu_i^2).$$

Moreover, ver_1 and ver_2 are then adjusted to verify the same relation as the previous constructions, but with the linear part extended to include the square terms.

Setup(1^λ)

1 : $\mathcal{G} \leftarrow \mathsf{BilinGroup}_3(\lambda)$
2 : **define sets:**
3 : $\quad \mathsf{ID}, \mathcal{T} \subseteq \{0,1\}^*$
4 : $\quad \mathcal{M} \subseteq \mathbb{Z}_q$
5 : **fix hashes:**
6 : $\quad H_1, H_2 : \{0,1\}^* \to \mathbb{G}_1$
7 : $\mathsf{pp} \leftarrow (\mathcal{G}, \mathsf{ID}, \mathcal{M}, \mathcal{T}, H_1, H_2)$
8 : **return** pp

KeyGen(pp)

1 : $\mathsf{id} \xleftarrow{\$} \mathsf{ID}$
2 : $\mathsf{sk}_{\mathsf{id}} \xleftarrow{\$} \mathbb{Z}_q^*$
3 : $\mathsf{pk}_{\mathsf{id}} = g_2^{\mathsf{sk}_{\mathsf{id}}} \in \mathbb{G}_2$
4 : **return** $(\mathsf{sk}_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}}, \mathsf{id})$

Sign$(\mathsf{sk}_{\mathsf{id}}, \ell, m)$

1 : $\gamma = (H_1(\ell) \cdot g_1^{m})^{\mathsf{sk}_{\mathsf{id}}}$
2 : $\gamma' = (H_2(\ell) \cdot g_1^{m^2})^{\mathsf{sk}_{\mathsf{id}}}$
3 : $\mu = m$
4 : **return** $\sigma = (\mathsf{id}, \gamma, \gamma', \mu)$

Figure 6.2: The Setup, KeyGen and Sign algorithms for the message-squares extension. Green text indicates changes from Figure 5.1.

We also emphasize that the cross-term mechanism remains identical to mkqhs-br, so the evaluated signature is unchanged in size at $O(tR)$. However, signing m^2 alongside m introduces an additional trust assumption on the signer, which is discussed further in Chapter 7.

6.2.5 Correctness of mkqhs-br-m^2

We show that mkqhs-br-m^2 satisfies evaluation correctness. Authentication correctness follows as a special case for $n = 1$ by setting $a_1 = 1$, $b_1 = 0$ and $R = 0$.

Suppose that the messages $\{m_i\}_{i=1}^n$ have been honestly signed. For each $i \in [n]$, the signature then consists of $\sigma_i = (\mathsf{id}_i, \gamma_i, \gamma_i', \mu_i)$, where

$$\gamma_i = (H_1(\ell_i) g_1^{m_i})^{\mathsf{sk}_{\mathsf{id}_i}}, \quad \gamma_i' = (H_2(\ell_i) g_1^{m_i^2})^{\mathsf{sk}_{\mathsf{id}_i}}, \quad \text{and} \quad \mu_i = m_i.$$

$\mathsf{Eval}(\mathcal{P}, \{\sigma_i\}_{i=1}^n)$

1 : **parse** $\mathcal{P} = (f, \{\ell_i\}_{i=1}^n)$
2 : **parse** $f = (\{a_i, b_i, \mathbf{u}_i, \mathbf{v}_i\}_{i=1}^n)$
3 : **parse** $\mathbf{u}_i = (u_{i,1}, \ldots, u_{i,R})^\top$
4 : **parse** $\mathbf{v}_i = (v_{i,1}, \ldots, v_{i,R})^\top$
5 : **parse** $\sigma_i = (\mathsf{id}_i, \gamma_i, \gamma_i', \mu_i)$
6 : $\gamma^{(a,b)} = \prod_{i=1}^n \gamma_i^{a_i} (\gamma_i')^{b_i}$
7 : **for** $r \in [R]$
8 : $\quad \gamma_r^{(u)} = \prod_{i=1}^n \gamma_i^{u_{i,r}},\ \gamma_r^{(v)} = \prod_{i=1}^n \gamma_i^{v_{i,r}}$
9 : $\boldsymbol{\gamma}^{(u)} = (\gamma_1^{(u)}, \ldots, \gamma_R^{(u)})^\top$
10 : $\boldsymbol{\gamma}^{(v)} = (\gamma_1^{(v)}, \ldots, \gamma_R^{(v)})^\top$
11 : $\tilde{\gamma} = \left(\gamma^{(a,b)}, \boldsymbol{\gamma}^{(u)}, \boldsymbol{\gamma}^{(v)}\right)$
12 : **for** $\mathsf{id} \in \mathcal{P}$
13 : $\quad \mathcal{I}_{\mathsf{id}} = \{\, i \in [n] \mid \ell_i = (\mathsf{id}, \cdot) \,\}$
14 : $\quad \mu_{\mathsf{id}}^{(a,b)} = \sum_{i \in \mathcal{I}_{\mathsf{id}}} (a_i \mu_i + b_i \mu_i^2)$
15 : $\quad \boldsymbol{\mu}_{\mathsf{id}}^{(u)} = \sum_{i \in \mathcal{I}_{\mathsf{id}}} \mu_i \, \mathbf{u}_i,\ \boldsymbol{\mu}_{\mathsf{id}}^{(v)} = \sum_{i \in \mathcal{I}_{\mathsf{id}}} \mu_i \, \mathbf{v}_i$
16 : $\quad \tilde{\mu}_{\mathsf{id}} = (\mu_{\mathsf{id}}^{(a,b)}, \boldsymbol{\mu}_{\mathsf{id}}^{(u)}, \boldsymbol{\mu}_{\mathsf{id}}^{(v)})$
17 : **return** $\tilde{\sigma} = (\tilde{\gamma}, \{\tilde{\mu}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}})$

$\mathsf{Verify}(\mathcal{P}, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}}, \tilde{m}, \tilde{\sigma})$

1 : **parse** $\mathcal{P} = (f, \{\ell_i\}_{i=1}^n),\ f = (\{a_i, b_i, \mathbf{u}_i, \mathbf{v}_i\}_{i=1}^n)$
2 : **parse** $\tilde{\sigma} = (\tilde{\gamma}, \{\tilde{\mu}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}}),\ \tilde{\gamma} = \left(\gamma^{(a,b)}, \boldsymbol{\gamma}^{(u)}, \boldsymbol{\gamma}^{(v)}\right)$
3 : **for** $\mathsf{id} \in \mathcal{P}$
4 : $\quad$ **parse** $\tilde{\mu}_{\mathsf{id}} = (\mu_{\mathsf{id}}^{(a,b)}, \boldsymbol{\mu}_{\mathsf{id}}^{(u)}, \boldsymbol{\mu}_{\mathsf{id}}^{(v)})$
5 : $\quad \mathcal{I}_{\mathsf{id}} = \{\, i \in [n] \mid \ell_i = (\mathsf{id}, \cdot) \,\}$
6 : $\mu^{(a,b)} = \sum_{\mathsf{id} \in \mathcal{P}} \mu_{\mathsf{id}}^{(a,b)}$
7 : $\boldsymbol{\mu}^{(u)} = \sum_{\mathsf{id} \in \mathcal{P}} \boldsymbol{\mu}_{\mathsf{id}}^{(u)},\ \boldsymbol{\mu}^{(v)} = \sum_{\mathsf{id} \in \mathcal{P}} \boldsymbol{\mu}_{\mathsf{id}}^{(v)}$
8 : $\mathsf{ver}_1 \leftarrow \left[\tilde{m} == \mu^{(a,b)} + \langle \boldsymbol{\mu}^{(u)}, \boldsymbol{\mu}^{(v)} \rangle\right]$
9 : $\mathsf{ver}_2 \leftarrow \left[e(\gamma^{(a,b)}, g_2) == \prod_{\mathsf{id} \in \mathcal{P}} e\left(g_1^{\mu_{\mathsf{id}}^{(a,b)}} \cdot \prod_{i \in \mathcal{I}_{\mathsf{id}}} H_1(\ell_i)^{a_i} H_2(\ell_i)^{b_i}, \mathsf{pk}_{\mathsf{id}}\right)\right]$
10 : $(\boldsymbol{\rho}, \boldsymbol{\rho}') \xleftarrow{\$} (\mathbb{Z}_q^*)^{R \times 2}$
11 : $\Gamma_\rho = (\boldsymbol{\gamma}^{(u)})^{\boldsymbol{\rho}} (\boldsymbol{\gamma}^{(v)})^{\boldsymbol{\rho}'}$
12 : $\mathsf{ver}_3 \leftarrow \left[e(\Gamma_\rho, g_2) == \prod_{\mathsf{id} \in \mathcal{P}} e\left(g_1^{\langle \boldsymbol{\rho}, \boldsymbol{\mu}_{\mathsf{id}}^{(u)} \rangle + \langle \boldsymbol{\rho}', \boldsymbol{\mu}_{\mathsf{id}}^{(v)} \rangle} \cdot \prod_{i \in \mathcal{I}_{\mathsf{id}}} H_1(\ell_i)^{\langle \boldsymbol{\rho}, \mathbf{u}_i \rangle + \langle \boldsymbol{\rho}', \mathbf{v}_i \rangle}, \mathsf{pk}_{\mathsf{id}}\right)\right]$
13 : **return** $[\mathsf{ver}_1 \wedge \mathsf{ver}_2 \wedge \mathsf{ver}_3]$

Figure 6.3: The Eval and Verify algorithms for mkqhs-br-m^2. Setup, KeyGen, and Sign are given in Figure 6.2. Green text indicates changes from Figure 5.2.

Algebraic check (ver_1). Summing the per-identity message-square aggregates $\mu_{\mathsf{id}}^{(a,b)}$ over all identities in $\mathcal{P}$ yields the global aggregate

$$\mu^{(a,b)} = \sum_{\mathsf{id}\in\mathcal{P}} \mu_{\mathsf{id}}^{(a,b)} = \sum_{\mathsf{id}\in\mathcal{P}} \sum_{i\in\mathcal{I}_{\mathsf{id}}} (a_i m_i + b_i m_i^2) = \sum_{i=1}^{n} a_i m_i + b_i m_i^2.$$

Combined with the global quadratic aggregates $\boldsymbol{\mu}^{(u)}$ and $\boldsymbol{\mu}^{(v)}$ (derived as in Section 5.5), the verifier checks

$$\begin{aligned}\mu^{(a,b)} + \langle \boldsymbol{\mu}^{(u)}, \boldsymbol{\mu}^{(v)} \rangle &= \sum_{i=1}^{n} (a_i m_i + b_i m_i^2) + \sum_{r=1}^{R} \left(\sum_{i=1}^{n} u_{i,r} m_i \right) \left(\sum_{i=1}^{n} v_{i,r} m_i \right) \\ &= f(m_1, \ldots, m_n) = \tilde{m}.\end{aligned}$$

Thus, ver_1 passes for honest evaluations.

Linear pairing check (ver_2). Expanding $\gamma^{(a,b)}$ using the honest signing equations and grouping by identity yields

$$\begin{aligned}\gamma^{(a,b)} &= \prod_{i=1}^{n} \gamma_i^{a_i} (\gamma_i')^{b_i} \\ &= \prod_{i=1}^{n} \left((H_1(\ell_i) g_1^{m_i})^{a_i} \cdot (H_2(\ell_i) g_1^{m_i^2})^{b_i} \right)^{\mathsf{sk}_{\mathsf{id}_i}} \\ &= \prod_{\mathsf{id}\in\mathcal{P}} \left(\prod_{i\in\mathcal{I}_{\mathsf{id}}} H_1(\ell_i)^{a_i} H_2(\ell_i)^{b_i} \cdot g_1^{(a_i m_i + b_i m_i^2)} \right)^{\mathsf{sk}_{\mathsf{id}}} \\ &= \prod_{\mathsf{id}\in\mathcal{P}} \left(g_1^{\mu_{\mathsf{id}}^{(a,b)}} \cdot \prod_{i\in\mathcal{I}_{\mathsf{id}}} H_1(\ell_i)^{a_i} H_2(\ell_i)^{b_i} \right)^{\mathsf{sk}_{\mathsf{id}}}.\end{aligned}$$

By the bilinearity of e and $\mathsf{pk}_{\mathsf{id}} = g_2^{\mathsf{sk}_{\mathsf{id}}}$, we obtain

$$\begin{aligned}e(\gamma^{(a,b)}, g_2) &= \prod_{\mathsf{id}\in\mathcal{P}} e\left(\left(g_1^{\mu_{\mathsf{id}}^{(a,b)}} \cdot \prod_{i\in\mathcal{I}_{\mathsf{id}}} H_1(\ell_i)^{a_i} H_2(\ell_i)^{b_i} \right)^{\mathsf{sk}_{\mathsf{id}}}, g_2 \right) \\ &= \prod_{\mathsf{id}\in\mathcal{P}} e\left(g_1^{\mu_{\mathsf{id}}^{(a,b)}} \cdot \prod_{i\in\mathcal{I}_{\mathsf{id}}} H_1(\ell_i)^{a_i} H_2(\ell_i)^{b_i}, \mathsf{pk}_{\mathsf{id}} \right),\end{aligned}$$

which matches the verification equation of ver_2 in Figure 6.3. Hence, the check always passes.

Quadratic pairing check (ver_3). The check ver_3 passes deterministically as in the correctness proof of mkqhs-br from Section 5.5, since ver_3 remains identical with H_1 taking the place of H.

6.2.6 Security Theorem and Proof Outline

Theorem 6.4. *Assuming that the co-CDH* problem is hard, the scheme* mkqhs-br-m^2 *in Figure 6.3 is secure under* HomUF-CMA-DHQ *in the random oracle model. Formally, let $\mathcal{A}$ be a PPT adversary in the* HomUF-CMA-DHQ *security experiment. Then its advantage is bounded by*

$$\mathsf{Adv}^{\mathsf{HomUF\text{-}CMA\text{-}DHQ}}_{\mathcal{A},\,\mathsf{mkqhs\text{-}br\text{-}m^2}}(\lambda) \;\leq\; 5 \cdot \mathsf{Adv}^{\text{co-CDH}^*}_{\mathcal{B}}(\lambda),$$

where $\mathcal{B}$ is a PPT algorithm that solves the co-CDH problem with advantage* $\mathsf{Adv}^{\text{co-CDH}^*}_{\mathcal{B}}(\lambda)$.

The proof follows that of Theorem 5.1 closely, and we defer the full proof to Appendix C. The structure of the proof is similar, where forgeries are distinguished between linear and quadratic type, and each is reduced to extracting a co-CDH* solution from $\gamma^{(a,b)*}$ or Γ^*_ρ respectively. The respective abort probabilities are bounded via the Schwartz–Zippel lemma. Quadratic extraction is identical to that of mkqhs-br, since the message-squares extension leaves ver_3 in Figure 6.3 structurally unchanged.

The main differences relative to the mkqhs-br proof are:

Simulation of H_2. The reduction now programs two random oracles instead of one. During the sign query phase, $\mathcal{B}$ samples independent $t, t' \xleftarrow{\$} \mathbb{Z}_q$ and programs both oracles simultaneously on the queried label as

$$\mathcal{H}_1(\ell) = g_1^t h^{-m}, \qquad \mathcal{H}_2(\ell) = g_1^{t'} h^{-m^2}.$$

This allows $\mathcal{B}$ to compute the new signature component γ' as $\gamma' = (X_1^{m^2 s + t'})^{y_{\mathsf{id}}}$, analogously to γ and without knowledge of x. For fresh hash queries on unsigned labels, $\mathcal{B}$ instead programs $\mathcal{H}_1(\ell) = h^r$ and $\mathcal{H}_2(\ell) = h^{r'}$ for $r, r' \xleftarrow{\$} \mathbb{Z}_q$. By applying Claim 4.3 independently to each oracle, the view of $\mathcal{A}$ remains identically distributed to a real execution of HomUF-CMA-DHQ.

Type-2 forgeries. In the mklhs_3 proof of Theorem 4.1, extraction relies on the per-identity mismatch δ_{id} and the intermediary variable $\hat{A}_{\mathsf{id}}$. In mkqhs-br-m^2, both quantities are extended to absorb the message-squares structure:

$$\delta_{\mathsf{id}} = \mu^{(a,b)*}_{\mathsf{id}} - \sum_{i \in \mathcal{I}_{\mathsf{id}}} \left(a_i^* m_i + b_i^* m_i^2 \right),$$

$$\hat{A}_{\mathsf{id}} = \tilde{g}_1^{\mu^{(a,b)*}_{\mathsf{id}}} \cdot \prod_{i \in \mathcal{I}_{\mathsf{id}}} \mathcal{H}_1(\ell_i^*)^{a_i^*}\, \mathcal{H}_2(\ell_i^*)^{b_i^*}.$$

The Type-2 forgery case-split then branches on whether some $\delta_{\mathsf{id}} \neq 0$ (linear case, handled exactly as in the mklhs_3 proof of Theorem 4.1) or all $\delta_{\mathsf{id}} = 0$ (quadratic case, handled exactly as in the mkqhs-br proof of Theorem 5.1).

Type-3 forgeries. The admissibility condition for mkqhs-br-m^2 described in Subsection 6.2.2 requires that every $i \in \mathcal{U}$ satisfies at least one of $(a_i^*, b_i^*) \neq (0,0)$ or $(u_{i,r}^*, v_{i,r}^*) \neq (0,0)$ for some $r \in [R]$. The Type-3 forgery case-split therefore branches on whether $(a_i^*, b_i^*) \neq (0,0)$ for *every* $i \in \mathcal{U}$ (linear case) or whether

some $i \in \mathcal{U}$ has $a_i^* = b_i^* = 0$ and $(u_{i,r}^*, v_{i,r}^*) \neq (0,0)$ for that i (quadratic case, handled exactly as in the mkqhs-br proof of Theorem 5.1).

In the linear case, extraction proceeds as in the Type-3 proof of Theorem 4.1, with the difference that each unsigned index $i \in \mathcal{U}_{\mathsf{id}}$ now contributes $a_i^* r_i + b_i^* r_i'$ to $\hat{b}_{\mathsf{id}}$ (rather than $a_i^* r_i$ alone), reflecting the two programmed hashes. A negligible abort probability follows from admissibility, which states that $(a_i^*, b_i^*) \neq (0,0)$ for every $i \in \mathcal{U}$.

Combining the four cases with the union bound yields the claimed advantage of $5 \cdot \mathsf{Adv}_{\mathcal{B}}^{\text{co-CDH}^*}(\lambda)$ for all primes $q \geq 11$.

6.3 The Combined Construction: mkqhs-$\widetilde{\text{br}}$-m^2

The two extensions presented in this chapter are both modifications of mkqhs-br that act on independent parts of the construction. The mkqhs-br-m^2 extension of Section 6.2 modifies only Sign, ver$_1$, and ver$_2$, while mkqhs-$\widetilde{\text{br}}$ of Section 6.1 modifies only ver$_3$. Since their changes do not interact, they can be applied simultaneously to mkqhs-br, yielding a construction that achieves the extended polynomial class (6.5) of mkqhs-br-m^2 and the compressed evaluated signature size $O(t + R)$ of mkqhs-$\widetilde{\text{br}}$.

Concretely, the combined construction is obtained by applying the message-squares signing of Figure 6.2 together with the compressed Eval and Verify of Figure 6.1, but replacing $\gamma^{(a)}$ and $\mu_{\mathsf{id}}^{(a)}$ throughout with $\gamma^{(a,b)}$ and $\mu_{\mathsf{id}}^{(a,b)}$ respectively. We refer to the combined construction as mkqhs-$\widetilde{\text{br}}$-m^2.

Correctness follows immediately from the correctness of mkqhs-$\widetilde{\text{br}}$ and mkqhs-br-m^2 applied independently to their respective components in mkqhs-$\widetilde{\text{br}}$-m^2. Security under HomUF-CMA-DHQ follows by combining the proofs of Theorems 6.1 and 6.4. Linear forgeries are handled by the simulation and extraction arguments of mkqhs-br-m^2, and quadratic forgeries are handled by the mismatch argument of mkqhs-$\widetilde{\text{br}}$. The resulting advantage bound is therefore

$$\mathsf{Adv}^{\mathsf{HomUF\text{-}CMA\text{-}DHQ}}_{\mathcal{A},\,\text{mkqhs-}\widetilde{\text{br}}\text{-m}^2}(\lambda) \;\leq\; 5 \cdot \mathsf{Adv}_{\mathcal{B}}^{\text{co-CDH}^*}(\lambda),$$

identical to mkqhs-$\widetilde{\text{br}}$ and mkqhs-br-m^2.

7

Trust, Efficiency, and Implementation

We have now introduced all four constructions of this thesis and established their correctness and security. This chapter addresses three practical questions that remain.

Section 7.1 examines a new trust assumption introduced by the message-squares extension in mkqhs-br-m^2 and mkqhs-$\widetilde{\text{br}}$-m^2, and proposes a one-time batch consistency check that certifies an entire dataset of signatures with a single evaluation. Section 7.2 compares the four constructions in evaluated signature size and dominant computational cost, concluding by recommending mkqhs-$\widetilde{\text{br}}$-m^2 as the default for multi-key statistical applications. Section 7.3 describes a reference implementation of the four constructions.

7.1 Dishonest Signers

In mklhs, mkqhs-br and mkqhs-$\widetilde{\text{br}}$, the Sign algorithm produces a single group element $\gamma = (H(\ell) \cdot g_1^m)^{\mathsf{sk}}$ encoding the message $\mu = m$. In mkqhs-br-m^2 and mkqhs-$\widetilde{\text{br}}$-m^2, it additionally produces $\gamma' = (H_2(\ell) \cdot g_1^{m^2})^{\mathsf{sk}}$, encoding the square m^2 under the same label and key. This introduces a new degree of freedom where a signer could provide a γ' inconsistent with the message encoded in γ. We argue that this additional trust requirement is acceptable in practice, since any such inconsistency is detected at verification whenever the evaluator is honest. We also propose a concrete one-time consistency check that can certify consistency of a batch of signatures at once.

Detection with an honest evaluator. mkqhs-br-m^2.Eval in Figure 6.3 computes the per-identity $\mu_{\mathsf{id}}^{(a,b)} = \sum_{i \in \mathcal{I}_{\mathsf{id}}} (a_i \mu_i + b_i \mu_i^2)$ from the claimed messages μ_i, not from the γ_i' components. If a signer provides γ_i' encoding some $w_i \neq \mu_i^2$ while keeping $\mu_i = m_i$, a mismatch arises between $\mu_{\mathsf{id}}^{(a,b)}$ and $\gamma^{(a,b)}$ in the evaluated signature. The honest evaluator computes $\mu_{\mathsf{id}}^{(a,b)}$ from the scalar μ_i, and thus uses the correct m_i^2, while the group element $\gamma^{(a,b)} = \prod_{i \in [n]} \gamma_i^{a_i} (\gamma_i')^{b_i}$ incorporates the dishonest w_i via γ_i'. The resulting mismatch causes ver_2 to fail whenever $b_i \neq 0$, so a signer who provides an inconsistent γ_i' cannot pass verification against an honest evaluator on any labeled program that uses square terms.

Colluding signer and evaluator. If the evaluator is also dishonest, it could adjust $\mu_{\mathsf{id}}^{(a,b)}$ to match the inconsistent γ_i', allowing ver_2 to pass despite the claimed message not being honestly computed. However, this threat already exists in mklhs, mkqhs-br and mkqhs-$\widetilde{\text{br}}$. A dishonest signer can produce $\sigma^* = (\gamma^*, \mu^*)$ with $\gamma^* = (H(\ell) \cdot g_1^m)^{\mathsf{sk}}$ and $\mu^* \neq m$, and a colluding evaluator can adjust the aggregated signature accordingly. Hence, a colluding signer and evaluator can always produce a consistent but incorrect output regardless of the construction, and the message-squares extension introduces no additional attack surface. We note that this kind of collusion is outside the scope of both HomUF-CMA and HomUF-CMA-DHQ, which model all signers as honest and only consider dishonest evaluators.

A batch consistency check. We have established that a verifier may detect if some γ_i' is inconsistent in an honestly evaluated program, but such an inconsistency would pass unnoticed with $b_i = 0$ until γ_i' is used in a program with $b_i \neq 0$. The failing verification also does not indicate exactly which signature is malformed.

As an extra integrity check, we propose a procedure allowing anyone who wishes to do future evaluations on a dataset of individual signatures $\{\sigma_i\}_{i=1}^n = \{(\mathsf{id}_i, \gamma_i, \gamma_i', \mu_i)\}_{i=1}^n$, to run the one-time consistency check illustrated in Figure 7.1. The check produces a single evaluated signature $\tilde{\sigma}$ that certifies consistency of all γ_i' at once.

$\mathsf{ConsCheck}(\{\sigma_i\}_{i=1}^n, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}})$

1 : **parse** $\sigma_i = (\mathsf{id}_i, \gamma_i, \gamma_i', \mu_i)$ for each $i \in [n]$

2 : $(c_1, \ldots, c_n)^\top \xleftarrow{\$} (\mathbb{Z}_q^*)^n$

3 : Define labeled program $\mathcal{P}_c = (f_c, \{\ell_i\}_{i=1}^n)$ with $R = n$ and for each $i \in [n]$:

4 : $\quad a_i = 0, \quad b_i = -c_i, \quad \mathbf{u}_i = c_i\mathbf{e}_i \in \mathbb{Z}_q^n, \quad \mathbf{v}_i = \mathbf{e}_i \in \mathbb{Z}_q^n$

5 : $\tilde{\sigma} \leftarrow \mathsf{mkqhs\text{-}br\text{-}m}^2.\mathsf{Eval}(\mathcal{P}_c, \{\sigma_i\}_{i=1}^n)$

6 : **return** $\mathsf{mkqhs\text{-}br\text{-}m}^2.\mathsf{Verify}(\mathcal{P}_c, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}, 0, \tilde{\sigma})$

Figure 7.1: Batch consistency check for $\mathsf{mkqhs\text{-}br\text{-}m}^2$. By $\mathbf{e}_i \in \mathbb{Z}_q^n$ we mean the i-th standard basis vector.

If the consistency check fails, indicating that at least one signature in the dataset is inconsistent, a recursive binary search can isolate the malformed signature. Concretely, by repeatedly partitioning the dataset of n signatures into two halves, running the consistency check on each half, and recursively checking any half that fails, the inconsistent signatures can be located with $O(\log n)$ checks.

We acknowledge that our proposed check requires the evaluation of a labeled program with $R = n$, breaking the succinctness requirement of $R \leq O(\log n)$ for our constructions. We argue that this is an acceptable cost, since the check is intended as a one-time pre-evaluation step on a dataset, after which evaluations of arbitrary programs on the verified data may be executed safely. Hence, we consider the cost of the dataset check to be amortised across all subsequent evaluations on its signatures.

Lemma 7.1 (Batch Consistency for mkqhs-br-m^2). *Let* $\{\sigma_i\}_{i=1}^n = \{(\mathsf{id}_i, \gamma_i, \gamma_i', \mu_i)\}_{i=1}^n$ *be signatures from* mkqhs-br-m^2*, where* $\gamma_i' = (H_2(\ell_i) g_1^{w_i})^{\mathsf{sk}_{\mathsf{id}_i}}$ *for some* $w_i \in \mathbb{Z}_q$*. The procedure* ConsCheck *in Figure 7.1 then satisfies:*

Completeness. *If* $w_i = \mu_i^2$ *for all* $i \in [n]$*, then* ConsCheck *returns* 1 *deterministically.*

Soundness. *If* $w_i \neq \mu_i^2$ *for some* $i \in [n]$*, then* ConsCheck *returns* 0 *except with negligible probability.*

Proof. Let $\mathbf{c} = (c_1, \ldots, c_n)^\top \xleftarrow{\$} (\mathbb{Z}_q^*)^n$ be the vector of challenge coefficients sampled uniformly by the party running the check. The check runs Eval and Verify on the labeled program $\mathcal{P}$ with coefficients

$$a_i = 0, \qquad b_i = -c_i, \qquad R = n, \qquad \mathbf{u}_r = c_r \mathbf{e}_r, \qquad \mathbf{v}_r = \mathbf{e}_r,$$

where $\mathbf{e}_r \in \mathbb{Z}_q^n$ is the r-th standard basis vector. The corresponding evaluated function from (6.5) is then

$$f(\mathbf{m}) = -\sum_{i=1}^n c_i m_i^2 + \sum_{r=1}^n \Big(c_r \mathbf{e}_r^\top \mathbf{m}\Big)\Big(\mathbf{e}_r^\top \mathbf{m}\Big) = -\sum_{i=1}^n c_i m_i^2 + \sum_{r=1}^n c_r m_r^2 = 0. \tag{7.1}$$

The expected output is therefore always $\tilde{m} = 0$ regardless of the choice of $\mathbf{m}$.

Hence, if all signers behave honestly with $\gamma_i' = (H_2(\ell_i) g_1^{\mu_i^2})^{\mathsf{sk}_{\mathsf{id}_i}}$, the correctness of mkqhs-br-m^2 ensures that all three verification checks pass against $\tilde{m} = 0$ deterministically. This proves completeness.

Next, suppose some signer provides $\gamma_i' = (H_2(\ell_i) g_1^{w_i})^{\mathsf{sk}_{\mathsf{id}_i}}$ with $w_i \neq \mu_i^2$ for at least one index i. Substituting this dishonest γ_i' and the coefficients $b_i = -c_i$, $a_i = 0$ into the expression for $\gamma^{(a,b)}$ on line 6 in mkqhs-br-m^2.Eval (Figure 6.3) and grouping by identity, the evaluator computes

$$\begin{aligned}
\gamma^{(a,b)} &= \prod_{i=1}^n \gamma_i^{a_i} (\gamma_i')^{b_i} = \prod_{i=1}^n (\gamma_i')^{-c_i} \\
&= \prod_{i=1}^n \Big((H_2(\ell_i) g_1^{w_i})^{\mathsf{sk}_{\mathsf{id}_i}}\Big)^{-c_i} \\
&= \prod_{\mathsf{id} \in \mathcal{P}} \left(g_1^{-\sum_{i \in \mathcal{I}_{\mathsf{id}}} c_i w_i} \cdot \prod_{i \in \mathcal{I}_{\mathsf{id}}} H_2(\ell_i)^{-c_i} \right)^{\mathsf{sk}_{\mathsf{id}}}.
\end{aligned}$$

For each identity, the evaluator also computes $\mu_{\mathsf{id}}^{(a,b)}$ from the sent scalars μ_i (which the signer has claimed as their messages) as

$$\mu_{\mathsf{id}}^{(a,b)} = \sum_{i \in \mathcal{I}_{\mathsf{id}}} (a_i \mu_i + b_i \mu_i^2) = -\sum_{i \in \mathcal{I}_{\mathsf{id}}} c_i \mu_i^2.$$

After sending $\gamma^{(a,b)}$ and each $\mu_{\mathsf{id}}^{(a,b)}$ to the verifier, the check ver_2 from Figure 6.3 takes the form

$$e(\gamma^{(a,b)}, g_2) = e\left(\prod_{\mathsf{id}\in\mathcal{P}} \left(g_1^{-\sum_{i\in\mathcal{I}_{\mathsf{id}}} c_i w_i} \cdot \prod_{i\in\mathcal{I}_{\mathsf{id}}} H_2(\ell_i)^{-c_i}\right)^{\mathsf{sk}_{\mathsf{id}}},\ g_2\right)$$
$$\stackrel{?}{=} \prod_{\mathsf{id}\in\mathcal{P}} e\left(g_1^{-\sum_{i\in\mathcal{I}_{\mathsf{id}}} c_i \mu_i^2} \cdot \prod_{i\in\mathcal{I}_{\mathsf{id}}} H_2(\ell_i)^{-c_i},\ \mathsf{pk}_{\mathsf{id}}\right).$$

Applying bilinearity, together with $\mathsf{pk}_{\mathsf{id}} = g_2^{\mathsf{sk}_{\mathsf{id}}}$ and cancelling the matching $H_2(\ell_i)^{-c_i}$ factors on both sides, reduces ver_2 to

$$e\left(g_1^{-\sum_{\mathsf{id}\in\mathcal{P}} \mathsf{sk}_{\mathsf{id}} \sum_{i\in\mathcal{I}_{\mathsf{id}}} c_i w_i}, g_2\right) \stackrel{?}{=} e\left(g_1^{-\sum_{\mathsf{id}\in\mathcal{P}} \mathsf{sk}_{\mathsf{id}} \sum_{i\in\mathcal{I}_{\mathsf{id}}} c_i \mu_i^2}, g_2\right).$$

By non-degeneracy of the pairing and the partitioning of $[n]$ by $\mathcal{I}_{\mathsf{id}}$, this requires

$$\sum_{i=1}^{n} \mathsf{sk}_{\mathsf{id}_i}\, c_i\,(w_i - \mu_i^2) = 0. \tag{7.2}$$

We view the left-hand side of (7.2) as a linear polynomial in $\mathbf{c} \in (\mathbb{Z}_q^*)^n$. The coefficient of c_i is $\mathsf{sk}_{\mathsf{id}_i}(w_i - \mu_i^2)$, which is non-zero whenever $w_i \neq \mu_i^2$, since $\mathsf{sk}_{\mathsf{id}_i} \xleftarrow{\$} \mathbb{Z}_q^*$. Hence, the polynomial is not identically zero since there exists at least on inconsistent γ', and the Schwartz–Zippel lemma over $\mathbb{Z}_q^*$ bounds the probability that (7.2) holds by $1/(q-1)$.

Hence, if any signer provides an inconsistent γ_i', the consistency check will fail with overwhelming probability $1 - 1/(q-1)$, *i.e.*, when the check passes the verifier is assured that all γ_i' are consistent with their corresponding μ_i^2 except with negligible probability, proving soundness. □

7.2 Efficiency Evaluation

This section analyses the efficiency of the four constructions presented up to this point, relative to each other and to mklhs. We consider two aspects, namely the size of signatures returned by the schemes, and the dominant computational costs caused by each of their algorithms.

7.2.1 Signature Size Tradeoffs

Individual signature size. The output of Sign in all constructions is a constant-size individual signature σ_i that does not depend on number of messages n, number of signers t, or the rank R of the quadratic form. In mkqhs-br and mkqhs-$\widetilde{\text{br}}$, each signer outputs $\sigma_i = (\mathsf{id}_i, \gamma_i, \mu_i)$, consisting of one group element $\gamma_i \in \mathbb{G}_1$ and one scalar $\mu_i \in \mathbb{Z}_q$. The message-squares extension in mkqhs-br-m^2 and mkqhs-$\widetilde{\text{br}}$-m^2 adds one additional element $\gamma_i' \in \mathbb{G}_1$, causing a constant cost of one additional $\mathbb{G}_1$ element per signature.

Evaluated signature size. The compression introduced in mkqhs-$\widetilde{\text{br}}$ is not always beneficial, since for small R and few signers, sending the global vectors $\boldsymbol{\mu}^{(u)}, \boldsymbol{\mu}^{(v)}$ once may not compensate for dropping the per-identity vectors $\boldsymbol{\mu}_{\mathsf{id}}^{(u)}, \boldsymbol{\mu}_{\mathsf{id}}^{(v)}$.

In mkqhs-br (and mkqhs-br-m^2), the evaluated signature carries $2tR + t$ scalar elements in total, while in mkqhs-$\widetilde{\text{br}}$ (and mkqhs-$\widetilde{\text{br}}$-m^2) it carries $2t + 2R$ scalar elements. Therefore, mkqhs-$\widetilde{\text{br}}$ has fewer scalar elements than mkqhs-br if and only if $2t + 2R < 2tR + t$, which simplifies to

$$t > \frac{2R}{2R - 1}.$$

This condition is satisfied when $R \geq 2$ and $t \geq 2$, or when $R = 1$ and $t \geq 3$, as illustrated in Table 7.1. For $R = 0$, there is no quadratic component, and both mkqhs-br and mkqhs-$\widetilde{\text{br}}$ act identically to mklhs with neither being more efficient.

R \ t	1	2	3	$\cdots$
0	=	=	=	$\cdots$
1	br	=	$\widetilde{\text{br}}$	$\cdots$
2	br	$\widetilde{\text{br}}$	$\widetilde{\text{br}}$	$\cdots$
3	br	$\widetilde{\text{br}}$	$\widetilde{\text{br}}$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Table 7.1: Recommended scheme by smallest evaluated signature size for rank R and signers t. Dark shading indicates $\widetilde{\text{br}}$ variants are preferred ($t > \frac{2R}{2R-1}$), no shading indicates that br variants are preferred, and light indicates a tie.

7.2.2 Computational Complexity

We analyse the dominant computational costs of the four constructions, summarised in Table 7.2, and compare against the baseline mklhs construction.

We track three computation types ordered by cost: pairings, exponentiations in $\mathbb{G}_1$, and hash-to-$\mathbb{G}_1$ evaluations. This reflects the hierarchy of operation costs in pairing-based implementations, where pairings are the most expensive by a large margin, exponentiations in $\mathbb{G}_1$ are the second most expensive, and hash-to-$\mathbb{G}_1$ evaluations are cheaper than exponentiations but still meaningfully expensive. The remaining operations are omitted as negligible by comparison [29, Table 2].

Signing. Signing cost is constant in all parameters. The message-squares extension adds a single exponentiation in $\mathbb{G}_1$ and a single hash-to-$\mathbb{G}_1$, but since this is a small constant overhead, it does not affect the asymptotic cost and we do not consider it meaningful.

Evaluation. Evaluation cost grows with the rank R, since the evaluator must compute the $2R$ quadratic group elements $\boldsymbol{\gamma}^{(u)}$ and $\boldsymbol{\gamma}^{(v)}$. For the bounded-rank setting $R \leq O(\log n)$, this becomes $O(n \log n)$, which we consider modest. The message-squares extension adds one extra exponentiation per input signature.

Verification. The pairing count is the dominant cost and is equal across all four bounded-rank constructions at $2(t+1)$, *i.e.*, double the $t+1$ pairings of mklhs. This reflects the price of our method of supporting quadratic evaluations, since a second pairing equation is used to authenticate the quadratic part of the admissible function. Crucially, the pairing count does not grow with n or R,

only with the number of signers t, which we consider makes verification still practical for large datasets.

The costs of exponentiating in $\mathbb{G}_1$ for verification are linear in n, t and R for all constructions. The message-squares variants cause an extra n exponentiations and n hash evaluations relative to their non-squares counterparts, since ver_2 must process both H_1 and H_2 per label. This is meaningful but a fixed additive cost, not an asymptotic one.

		mklhs	br/$\widetilde{\text{br}}$	br-m^2/$\widetilde{\text{br}}$-m^2
Sign	exp. in $\mathbb{G}_1$	1	1	2
	hash-to-$\mathbb{G}_1$	1	1	2
Eval	exp. in $\mathbb{G}_1$	n	$n(2R+1)$	$n(2R+2)$
Verify	pairings	$t+1$	$2(t+1)$	$2(t+1)$
	exp. in $\mathbb{G}_1$	$t+n$	$2(t+R+n)$	$2t+2R+3n$
	hash-to-$\mathbb{G}_1$	n	n	$2n$

Table 7.2: Dominant computational costs per construction, with mklhs as the linear baseline. Costs are identical within each column pair (*i.e.* between mkqhs-br and mkqhs-$\widetilde{\text{br}}$, and between mkqhs-br-m^2 and mkqhs-$\widetilde{\text{br}}$-m^2). Light shading indicates increase in cost relative to mklhs and dark shading indicates a further increase beyond that.

7.2.3 Recommended Construction

We argue that the two analyses above motivate mkqhs-$\widetilde{\text{br}}$-m^{2}as the default choice for statistical applications. Among the four constructions, mkqhs-$\widetilde{\text{br}}$-m^2 dominates by inheriting both the compressed $O(t+R)$ signature size of mkqhs-$\widetilde{\text{br}}$ and the extended function class of mkqhs-br-m^2, with their respective overheads over mkqhs-br being independent and small.

On the signature size side, the compression in mkqhs-$\widetilde{\text{br}}$ and mkqhs-$\widetilde{\text{br}}$-m^2 reduces the evaluated signature size from $O(tR)$ to $O(t+R)$, which Table 7.1 shows is preferable whenever $R \geq 1$ and $t \geq 2$. Since MKHS are deployed by definition in settings where $t \geq 2$, and many statistical applications motivating our constructions have $R \geq 1$, this makes mkqhs-$\widetilde{\text{br}}$ and mkqhs-$\widetilde{\text{br}}$-m^2 preferable when it comes to succinctness. On the computational complexity side, the pairing count is the dominant cost and is equal across all four quadratic constructions, while the computational overhead of the message-squares extensions are small and additive in comparison. We therefore conclude that mkqhs-$\widetilde{\text{br}}$-m^2 offers the best balance of expressiveness, succinctness, and computational efficiency among the four constructions, and recommend it as the default for multi-key statistical applications.

7.3 Reference Implementation

To complement the theoretical contributions of this thesis, we provide a reference implementation of the constructions developed in Chapters 5 and 6. The implementation is written in Rust, built on the `arkworks` cryptographic library [5], and instantiated over the BLS12-381 Type-3 pairing-friendly curve, matching the Type-3 pairing setting used throughout the thesis. It is publicly available on GitHub [6] under the Apache-2.0 license.

Scope. The implementation is intended as a research artifact accompanying the thesis. It has not been audited and is not intended for production use. The library includes the baseline mklhs scheme of Aranha and Pagnin [1] alongside the four quadratic constructions developed in this thesis, with each scheme implemented as a separate Rust module

Worked examples. The repository contains two runnable examples that use the mkqhs-$\widetilde{\text{br}}$-m^2 scheme on the Efron–Hastie diabetes dataset [32], demonstrating end-to-end signing, evaluation, and verification on real data:

Variance. Verifiable computation of the sample variance of the diabetes target variable across $t = 10$ signers, instantiating the variance row of Table 6.1.

Squared Euclidean distance. Verifiable computation of the squared Euclidean distance between two randomly sampled patients on three data dimensions (age, BMI, blood pressure), instantiating Proposition 6.3 at dimension $d = 3$.

Both examples confirm that the constructions are practically usable on statistical inputs of the kind in Section 6.2.3.

Ethical considerations. The Efron–Hastie diabetes dataset [32] is a publicly available and fully anonymised benchmark of 442 patients with 10 data dimensions, widely used in statistical learning. No individuals can be identified from the data, and our use is restricted to demonstrating the correctness and feasibility of the constructions on statistical computations only.

8

Weakly-Secure HomUF-CMA Transformation

We have so far analysed the security of our schemes in the HomUF-CMA-DHQ experiment introduced in Chapter 3, which captures the delayed hash query restriction used in the proof technique of Aranha and Pagnin [1]. In this chapter we revisit the same schemes under a different security notion, the *weakly-secure* variant of HomUF-CMA introduced by Fiore *et al.* alongside the original formulation of HomUF-CMA [15]. Weakly-secure HomUF-CMA (Weak-HomUF-CMA) requires the adversary to commit, prior to the setup phase, to the collection of messages and tags it will request signatures on during the course of the game. The identities and dataset identifiers remain adaptive however, as does the ordering of the adversary's queries.

The interest in Weak-HomUF-CMA is twofold. First, it is the standard weaker notion of HomUF-CMA in the literature, and makes the security of our schemes directly comparable to other constructions. Secondly, Fiore *et al.* in their extended paper [33, Appendix A, Theorem 5] give a generic transformation that turn any Weak-HomUF-CMA secure scheme to a HomUF-CMA secure one, albeit requiring cryptographic primitives not yet known to be implementable.

8.1 The Weak-HomUF-CMA Security Game

Definition 8.1 (Weakly-Secure HomUF-CMA Game – Weak-HomUF-CMA [15]). The Weak-HomUF-CMA security experiment is identical to the HomUF-CMA security experiment in Defintion 2.28, except with an additional commitment phase occurring before the setup phase:

Commitment Phase. Before the setup phase, the adversary $\mathcal{A}$ commits to the total numbers Q_{id} and Q_{Δ} of distinct identities and datasets it will query during the game. For every pair $(i,k) \in [Q_{\mathsf{id}}] \times [Q_{\Delta}]$, $\mathcal{A}$ also commits to a collection $\mathcal{T}_{i,k} \subseteq \mathcal{T}$ of tags together with a set of corresponding messages $\{m_\tau\}_{\tau \in \mathcal{T}_{i,k}}$. Importantly, $\mathcal{A}$ is not required to specify the identities or datasets identifiers at this stage. The commitment is then sent to the challenger $\mathcal{C}$.

The experiment outputs 1 if the tuple returned by $\mathcal{A}$ is a HomUF-CMA forgery (as defined in Definition 2.29), and 0 otherwise.

Definition 8.2 (Weak-HomUF-CMA Unforgeable). A MKHS scheme is said to be Weak-HomUF-CMA *unforgeable* if, for every PPT adversary $\mathcal{A}$, its advantage in winning the Weak-HomUF-CMA game is negligible in the security parameter of the scheme. Formally,

$$\begin{aligned}\mathsf{Adv}^{\text{Weak-HomUF-CMA}}_{\mathcal{A},\,\text{MKHS}}(\lambda) &= \Pr[\mathcal{A} \text{ wins the Weak-HomUF-CMA game for } \mathsf{MKHS}(\lambda)] \\ &\leq \mathsf{negl}(\lambda).\end{aligned}$$

8.2 Weak-Secure Transformation of our Constructions

We now describe a transformation of all MKHS scheme considered in this thesis, and show that the transformed schemes are *single-dataset* Weak-HomUF-CMA *secure* in the random oracle model with non-adaptive corruption queries.

We recall that with non-adaptive corruption queries we mean that the adversary of the HomUF-CMA/Weak-HomUF-CMA security game can not corrupt any identity id that has already been used in a previous signing query.

The transformation is done by prefixing every hash query made inside Sign and Verify with the signing identity's public key. Concretely, every internal call to any hash $H_j : \{0,1\}^* \to \mathbb{G}_1$ within Sign and Verify on an input $\ell = (\mathsf{id}, \tau)$ is replaced according to

$$H_j(\ell) \;\longmapsto\; H_j(\mathsf{pk}_{\mathsf{id}},\, \ell),$$

where $\mathsf{pk}_{\mathsf{id}} = g_2^{\mathsf{sk}_{\mathsf{id}}}$ is the public key of the signing identity id. For mklhs, mkqhs-br and mkqhs-$\widetilde{\text{br}}$ this affects the single hash H, while for mkqhs-br-m^2 and mkqhs-$\widetilde{\text{br}}$-m^2 it applies to both H_1 and H_2.

With this modification, a simulation for the Weak-HomUF-CMA game can then program every hash used inside a sign query before the adversary can see the public key and query the oracle on the same input. This change does not alter the algebraic structure of the transformed schemes.

The result below is stated for a single dataset. In the multi-dataset setting, the labels in our constructions are $\ell = (\mathsf{id}, (\Delta, \tau'))$ where the dataset identifier Δ is chosen adaptively by the adversary. Hence, at the time the simulator gets an identity query for some id and generates $\mathsf{pk}_{\mathsf{id}_i}$, it does not yet know which dataset identifiers $\mathcal{A}$ will later use. It therefore cannot pre-program $\mathcal{H}_i(\mathsf{pk}_{\mathsf{id}_i}, \ell)$ for the relevant labels, since those labels are not fully determined until $\mathcal{A}$ reveals the dataset name in later sign queries.

Proposition 8.3. *With the transformation as described above, each scheme* mklhs, mkqhs-br, mkqhs-$\widetilde{\text{br}}$, mkqhs-br-m^2, mkqhs-$\widetilde{\text{br}}$-m^2 *is* Weak-HomUF-CMA *secure under the co-CDH* assumption in the random oracle model on a single dataset with non-adaptive corruption queries, and retain the same constant advantage bound as in their corresponding* HomUF-CMA-DHQ *security theorems.*

Proof. Each of the schemes listed in the proposition are provided with a HomUF-CMA-DHQ proof that uses a reduction $\mathcal{B}$ to solve the co-CDH* problem via the simulation described in Section 4.2.2, with accompanying modifications for each scheme. We argue that the same reduction handles any Weak-HomUF-CMA adversary $\mathcal{A}$ after a single modification to the identity-query simulation step. All other steps of the simulations and the extraction arguments are unchanged.

Removal of corruption queries. By the same argument as in Chapter 3, we remove corruption queries from our simulations using the result of Fiore *et al.* showing that a MKHS scheme is secure against adversaries that do not make corruption queries if and only if it is secure against adversaries that make non-adaptive corruption queries [15, Proposition 1].

Modification to the identity-query simulation. In the HomUF-CMA-DHQ proofs of the schemes, the flag b_H prevents hash queries from occurring before the signing phase is complete. In the Weak-HomUF-CMA experiment, there is no such restriction, and the adversary may issue hash queries at any point. The modification compensates for this by performing all signature creations at the moment a public key is generated.

Concretely, when $\mathcal{A}$ submits an identity query for the i-th identity id_i, the reduction $\mathcal{B}$ generates $\mathsf{pk}_{\mathsf{id}_i}$ exactly as in the HomUF-CMA-DHQ simulation, and then immediately executes the sign query simulation step for every committed pair (τ, m_τ) with $\tau \in \mathcal{T}_{i,1}$. Only after all programming is completed does $\mathcal{B}$ return $\mathsf{pk}_{\mathsf{id}_i}$ to $\mathcal{A}$. In other words, for all $\tau \in \mathcal{T}_{i,1}$ with corresponding m_τ, the signature $\sigma_\tau = \mathsf{Sign}(\mathsf{sk}_{\mathsf{id}_i}, \ell_\tau, m_\tau)$ with $\ell_\tau = (\mathsf{id}_i, \tau)$ gets created by $\mathcal{B}$ before it returns $\mathsf{pk}_{\mathsf{id}}$ to $\mathcal{A}$. This ensures that the random oracle $\mathcal{H}$ gets programmed on each input $(\mathsf{pk}_{\mathsf{id}_i}, \ell_\tau)$ as in the sign query phase of the HomUF-CMA-DHQ simulations.

For mklhs, mkqhs-br, and mkqhs-$\widetilde{\text{br}}$, the sign query simulation programs a single hash value $\mathcal{H}(\mathsf{pk}_{\mathsf{id}_i}, \ell_\tau)$ per label, while for mkqhs-br-m^2 and mkqhs-$\widetilde{\text{br}}$-m^2 it programs both $\mathcal{H}_1(\mathsf{pk}_{\mathsf{id}_i}, \ell_\tau)$ and $\mathcal{H}_2(\mathsf{pk}_{\mathsf{id}_i}, \ell_\tau)$ simultaneously, identical to their respective HomUF-CMA-DHQ proofs.

Ordering argument. By this transformation, every random oracle query that Sign makes for id_i is of the form $\mathcal{H}(\mathsf{pk}_{\mathsf{id}_i}, \ell)$ and therefore has $\mathsf{pk}_{\mathsf{id}_i}$ as its first argument. The public key itself is set by the reduction as $\mathsf{pk}_{\mathsf{id}_i} = (X_2)^{y_{\mathsf{id}_i}} = g_2^{x y_{\mathsf{id}_i}}$ with $y_{\mathsf{id}_i} \xleftarrow{\$} \mathbb{Z}_q^*$, and is revealed to $\mathcal{A}$ only after all committed signing for id_i has been completed. Since $\mathcal{A}$ does not know $\mathsf{pk}_{\mathsf{id}_i}$ before it is returned, and it is returned only once all random oracle programming for id_i's committed labels has occurred, $\mathcal{A}$ cannot issue a hash query on any of those inputs before the corresponding programmed value has been written to L_H.

This replaces the role of the b_H flag. Instead of the experiment stopping hash queries until signing is done, the impossibility of querying a hash that contains a yet unknown public key ensures the same ordering. Hash queries that $\mathcal{A}$ makes on inputs of the form $(\mathsf{pk}_{\mathsf{id}_i}, \ell)$ after receiving $\mathsf{pk}_{\mathsf{id}_i}$ then fall into two categories. Firstly,

queries where $\ell = (\mathsf{id}_i, \tau)$ with $\tau \in \mathcal{T}_{i,1}$ return the pre-programmed values, which are consistent with the corresponding signatures. Secondly, queries where $\tau \notin \mathcal{T}_{i,1}$ are handled by setting $\mathcal{H}(\mathsf{pk}_{\mathsf{id}_i}, \ell) = h^r$ for fresh $r \xleftarrow{\$} \mathbb{Z}_q$, as in the hash-query phase of the HomUF-CMA-DHQ simulations.

Forgery extraction. Once $\mathcal{A}$ produces its forgery, $\mathcal{B}$ determines the forgery type and the corresponding co-CDH* extraction is conducted exactly as in the HomUF-CMA-DHQ proofs. Type-1 forgeries are no longer relevant since there is only one dataset, while for Type-2 and Type-3 forgeries the extraction proceeds unchanged. Every signed label's hash was programmed as $\mathcal{H}(\ell) = g_1^t h^{-m}$ during the modified identity-query step, and every unsigned label's hash was programmed as $\mathcal{H}(\ell) = h^r$ during a hash query, so the expressions for $\hat{A}_{\mathsf{id}}$, $\hat{B}$, and $\hat{c}$ are identical to those in the HomUF-CMA-DHQ proofs. The advantage of $\mathcal{B}$ is therefore the same as in the HomUF-CMA-DHQ theorems for each of the schemes. □

Remark 8.4. We note that the Weak-HomUF-CMA secure versions of our schemes may also be extended to support multiple datasets by applying the generic single-dataset to multi-dataset transformation of Fiore *et al.* [15, Theorem 2]. A further extension to full HomUF-CMA security with non-adaptive corruption queries in the random oracle model may also be done by applying the generic Weak-HomUF-CMA to HomUF-CMA transformation also given by Fiore *et al.* in the extended version of their paper [33, Appendix A, Theorem 5]. Although, the latter transformation requires cryptographic primitives that are not yet known to be implementable.

9

Final Words and Future Directions

9.1 Conclusion

This thesis was motivated by the observation that outsourcing computation is only useful to the extent that the returned results can be trusted. Homomorphic signature schemes address this problem by allowing a verifier to confirm that a returned result was correctly computed from originally signed inputs, without redoing the computation themselves. In the multi-key setting, where inputs come from independent parties each with their own secret key, implemented pairing-based constructions were limited to linear functions. This left a gap between the linear evaluations the schemes could verify and the quadratic functions that appear in practical statistical settings.

The realisation that drove our progress was that quadratic evaluation did not have to be computed directly. Any quadratic form of rank R decomposes into R rank one terms, each being a product of two linear evaluations. Thus, we could reach beyond linear evaluation using the same algebraic mechanisms already present in the linear mklhs scheme.

During the course of this work, we first introduced HomUF-CMA-DHQ, a security notion capturing the exact restrictions used in the original security proof of mklhs. Following, we reproved the security of mklhs in the Type 3 pairing setting with a tighter bound. We then constructed mkqhs-br by extending mklhs, resulting in a pairing-based MKHS supporting bounded-rank quadratic evaluation under the co-CDH* assumption. We then extended it further in two independent directions: (i) mkqhs-$\widetilde{\text{br}}$ which reduces the evaluated signature size, and (ii) mkqhs-br-m^2 which broadens the admissible function class. These were combined into mkqhs-$\widetilde{\text{br}}$-m^2. These four constructions are our main contributions: a family of pairing-based MKHS supporting bounded-rank quadratic evaluation. We continued by discussing efficiency and trust considerations across the four constructions, and provided a reference implementation in Rust demonstrating their practicality. Finally, we showed that all constructions can be lifted to the Weak-HomUF-CMA security notion via a straightforward transformation.

The bounded rank restriction remains the primary limitation, and lifting it is the clearest direction for future work. However, evaluation in implemented MKHS has now gone beyond linear.

9.2 Future Work

The constructions presented in this thesis take a step toward non-linear, pairing-based multi-key homomorphic signatures. Nevertheless, several future directions remain open, ranging from removing the bounded-rank restriction to supporting multi-hop evaluation.

Full-rank succinctness via inner-product argument. A possible approach for achieving full-rank succinctness is to further compress the evaluated signature of $\widetilde{\mathsf{mkqhs\text{-}br}}$ so that all dependence on R becomes logarithmic. We recall from Figure 6.1 that the R-dependence in $\widetilde{\mathsf{mkqhs\text{-}br}}$ comes from the per-rank group elements $\{\gamma_r^{(u)}, \gamma_r^{(v)}\}_{r=1}^{R} \in \mathbb{G}_1^{2R}$ and the scalar vectors $\boldsymbol{\mu}^{(u)}, \boldsymbol{\mu}^{(v)} \in \mathbb{Z}_q^R$.

Firstly, we suggest that the R-dependence from the $\mathbb{G}_1$ elements may be removed altogether by compressing the per-rank pairs $(\gamma_r^{(u)}, \gamma_r^{(v)})$ into a single per-identity group element

$$\tilde{\gamma}_{\mathsf{id}}^{(u,v)} := \prod_{i \in \mathcal{I}_{\mathsf{id}}} \gamma_i^{\langle \boldsymbol{\rho}, \mathbf{u}_i \rangle + \langle \boldsymbol{\rho}', \mathbf{v}_i \rangle} \in \mathbb{G}_1,$$

changing the evaluated signature to having only $t+1$ group elements. The verification can then be modified to accommodate this change while retaining $\mathsf{HomUF\text{-}CMA\text{-}DHQ}$ security. Indeed, we give such an intermediate construction and its security proof in Appendix D.

After this transformation, the only remaining R-dependence comes from $\boldsymbol{\mu}^{(u)}, \boldsymbol{\mu}^{(v)}$. These vectors are used only through the inner products $\langle \boldsymbol{\mu}^{(u)}, \boldsymbol{\mu}^{(v)} \rangle$ and $\langle \boldsymbol{\rho}, \boldsymbol{\mu}^{(u)} \rangle + \langle \boldsymbol{\rho}', \boldsymbol{\mu}^{(v)} \rangle$ in ver_1 and ver_4 respectively. Importantly, the individual elements of the vectors are never accessed, which we suggest makes them a good fit for an *inner-product argument* (IPA).

An IPA lets a prover convince a verifier of an inner-product equation without revealing the underlying vectors [34], [35]. Concretely, given public and independent generator vectors $\mathbf{g}, \mathbf{h} \in \mathbb{G}_1^R$, a prover holding secret vectors $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_q^R$ may convince a verifier that they know $\mathbf{a}, \mathbf{b}$ satisfying

$$c = \langle \mathbf{a}, \mathbf{b} \rangle \in \mathbb{Z}_q \qquad \text{and} \qquad P = \mathbf{g}^{\mathbf{a}} \mathbf{h}^{\mathbf{b}} \in \mathbb{G}_1,$$

by sending c, P, and a proof π of size $O(\log R)$ to the verifier.

We propose that one could further compress the intermediate construction of Appendix D by replacing the pair $(\boldsymbol{\mu}^{(u)}, \boldsymbol{\mu}^{(v)})$ in the evaluated signature with the inner products required by ver_1 and ver_4, together with their corresponding commitments P and proofs π. The reduced evaluated signature size of $O(t+\log R)$ would yield succinct evaluation of all quadratic polynomials by allowing full rank quadratic forms with $R = n$.

What makes this approach non-trivial is the *binding* of P to the rest of the scheme. Since an IPA only guarantees that some pair of vectors opens P with the claimed inner product, an adversary could pick P for vectors that do not have relation to the signed messages. Tying P to the signature elements verified in ver_2 and ver_3 therefore requires additional structure, which we leave as future work.

Full quadratic evaluation via a chain of pairings. The bilinear property of pairings is appealing for computing cross-terms $m_i \cdot m_j$ directly. However, we only explored the naive approach before abandoning the idea, as discussed in Section 5.1. A possible direction is to instead explore chains of pairing-friendly elliptic curves, where the output group of one pairing serves as the input group of the next [36]. Such chaining could potentially enable full quadratic evaluation without the rank decomposition our current construction requires.

Context hiding. Evaluated signatures in our constructions leak information about the underlying input messages beyond what is known from the computation result. *Context hiding* is the property that prevents this. It has already been achieved for pairing-based MKHS supporting linear functions [37], and extending it to our quadratic setting is a natural direction for future work.

Multi-hop. Our constructions support only *single-hop* evaluation, meaning that an evaluated signature cannot itself be used in further homomorphic evaluations. Extending the schemes to support *multi-hop* evaluation would allow chaining computations across different stages by feeding evaluated signatures back into Eval. For example, a verifier could first query for Euclidean distances of freshly signed data. Following, it could query for the averages of these distances. By the multi-hop property, a single evaluated signature is produced certifying the correct computation of both. The main obstacles to full multi-hop support are that chaining quadratic evaluations produces higher-degree polynomials outside of the supported class, and that the rank of the evaluation grows across hops, eventually violating our definition of succinctness. Resolving these limitations is a natural direction for future work.

Bibliography

[1] D. F. Aranha and E. Pagnin, "The simplest multi-key linearly homomorphic signature scheme," in *Progress in Cryptology – LATINCRYPT 2019*, P. Schwabe and N. Thériault, Eds., Cham: Springer International Publishing, 2019, pp. 280–300, ISBN: 978-3-030-30530-7.

[2] D. Catalano, D. Fiore, and I. Tucker, "Additive-homomorphic functional commitments and applications to homomorphic signatures," in *Advances in Cryptology – ASIACRYPT 2022, Part IV*, S. Agrawal and D. Lin, Eds., ser. Lecture Notes in Computer Science, vol. 13794, Taipei, Taiwan: Springer, Cham, Switzerland, Dec. 2022, pp. 159–188. DOI: 10.1007/978-3-031-22972-5_6.

[3] S. D. Galbraith, K. G. Paterson, and N. P. Smart, "Pairings for cryptographers," *Discrete Applied Mathematics*, vol. 156, no. 16, pp. 3113–3121, 2008, Applications of Algebra to Cryptography, ISSN: 0166-218X. DOI: https://doi.org/10.1016/j.dam.2007.12.010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0166218X08000449.

[4] S. Chatterjee and A. Menezes, "On cryptographic protocols employing asymmetric pairings the role of revisited," *Discrete Applied Mathematics*, vol. 159, no. 13, pp. 1311–1322, 2011, ISSN: 0166-218X. DOI: https://doi.org/10.1016/j.dam.2011.04.021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0166218X11001648.

[5] arkworks contributors, ***arkworks*** *zksnark ecosystem*, 2022. [Online]. Available: https://arkworks.rs.

[6] A. Jamal and A. Davoodi, ***mkqhs****: Reference implementation of multi-key (quadratic) homomorphic signatures in Rust*, https://github.com/aramyamal/mkqhs, 2026.

[7] R. Johnson, D. Molnar, D. Song, and D. Wagner, "Homomorphic signature schemes," in *Topics in Cryptology — CT-RSA 2002*, B. Preneel, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 244–262, ISBN: 978-3-540-45760-2.

[8] N. Attrapadung, B. Libert, and T. Peters, "Efficient completely context-hiding quotable and linearly homomorphic signatures," in *Public-Key Cryptography – PKC 2013*, K. Kurosawa and G. Hanaoka, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 386–404, ISBN: 978-3-642-36362-7.

[9] D. Boneh, D. Freeman, J. Katz, and B. Waters, "Signing a linear subspace: Signature schemes for network coding," in *PKC 2009: 12th International Conference on Theory and Practice of Public Key Cryptography*, S. Jarecki and G. Tsudik, Eds., ser. Lecture Notes in Computer Science, vol. 5443, Irvine,

CA, USA: Springer Berlin Heidelberg, Germany, Mar. 2009, pp. 68–87. DOI: 10.1007/978-3-642-00468-1_5.

[10] D. Catalano, D. Fiore, and B. Warinschi, "Efficient network coding signatures in the standard model," in *PKC 2012: 15th International Conference on Theory and Practice of Public Key Cryptography*, M. Fischlin, J. Buchmann, and M. Manulis, Eds., ser. Lecture Notes in Computer Science, vol. 7293, Darmstadt, Germany: Springer Berlin Heidelberg, Germany, May 2012, pp. 680–696. DOI: 10.1007/978-3-642-30057-8_40.

[11] D. Boneh and D. M. Freeman, "Homomorphic signatures for polynomial functions," in *Advances in Cryptology – EUROCRYPT 2011*, K. G. Paterson, Ed., ser. Lecture Notes in Computer Science, vol. 6632, Tallinn, Estonia: Springer Berlin Heidelberg, Germany, May 2011, pp. 149–168. DOI: 10.1007/978-3-642-20465-4_10.

[12] D. Catalano, D. Fiore, and B. Warinschi, "Homomorphic signatures with efficient verification for polynomial functions," in *Advances in Cryptology – CRYPTO 2014, Part I*, J. A. Garay and R. Gennaro, Eds., ser. Lecture Notes in Computer Science, vol. 8616, Santa Barbara, CA, USA: Springer Berlin Heidelberg, Germany, Aug. 2014, pp. 371–389. DOI: 10.1007/978-3-662-44371-2_21.

[13] S. Gorbunov, V. Vaikuntanathan, and D. Wichs, "Leveled fully homomorphic signatures from standard lattices," in *47th Annual ACM Symposium on Theory of Computing*, R. A. Servedio and R. Rubinfeld, Eds., Portland, OR, USA: ACM Press, Jun. 2015, pp. 469–477. DOI: 10.1145/2746539.2746576.

[14] S. Agrawal, D. Boneh, X. Boyen, and D. M. Freeman, "Preventing pollution attacks in multi-source network coding," in *PKC 2010: 13th International Conference on Theory and Practice of Public Key Cryptography*, P. Q. Nguyen and D. Pointcheval, Eds., ser. Lecture Notes in Computer Science, vol. 6056, Paris, France: Springer Berlin Heidelberg, Germany, May 2010, pp. 161–176. DOI: 10.1007/978-3-642-13013-7_10.

[15] D. Fiore, A. Mitrokotsa, L. Nizzardo, and E. Pagnin, "Multi-key homomorphic authenticators," in *Advances in Cryptology – ASIAChomomoRYPT 2016*, J. H. Cheon and T. Takagi, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 499–530, ISBN: 978-3-662-53890-6.

[16] D. Fiore and E. Pagnin, "Matrioska: A compiler for multi-key homomorphic signatures," in *SCN 18: 11th International Conference on Security in Communication Networks*, D. Catalano and R. De Prisco, Eds., ser. Lecture Notes in Computer Science, vol. 11035, Amalfi, Italy: Springer, Cham, Switzerland, Sep. 2018, pp. 43–62. DOI: 10.1007/978-3-319-98113-0_3.

[17] R. W. F. Lai, R. K. H. Tai, H. W. H. Wong, and S. S. M. Chow, "Multi-key homomorphic signatures unforgeable under insider corruption," in *Advances in Cryptology – ASIACRYPT 2018, Part II*, T. Peyrin and S. Galbraith, Eds., ser. Lecture Notes in Computer Science, vol. 11273, Brisbane, Queensland, Australia: Springer, Cham, Switzerland, Dec. 2018, pp. 465–492. DOI: 10.1007/978-3-030-03329-3_16.

[18] G. Anthoine, D. Balbás, and D. Fiore, "Fully-succinct multi-key homomorphic signatures from standard assumptions," in *Advances in Cryptology –*

CRYPTO 2024, Part III, L. Reyzin and D. Stebila, Eds., ser. Lecture Notes in Computer Science, vol. 14922, Santa Barbara, CA, USA: Springer, Cham, Switzerland, Aug. 2024, pp. 317–351. DOI: 10.1007/978-3-031-68382-4_10.

[19] M. Ajtai, "Generating hard instances of lattice problems (extended abstract)," in *28th Annual ACM Symposium on Theory of Computing*, Philadephia, PA, USA: ACM Press, May 1996, pp. 99–108. DOI: 10.1145/237814.237838.

[20] D. Boneh and D. M. Freeman, "Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures," in *PKC 2011: 14th International Conference on Theory and Practice of Public Key Cryptography*, D. Catalano, N. Fazio, R. Gennaro, and A. Nicolosi, Eds., ser. Lecture Notes in Computer Science, vol. 6571, Taormina, Italy: Springer Berlin Heidelberg, Germany, Mar. 2011, pp. 1–16. DOI: 10.1007/978-3-642-19379-8_1.

[21] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM Journal on Computing*, vol. 18, no. 1, pp. 186–208, 1989.

[22] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer, "From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again," in *ITCS 2012: 3rd Innovations in Theoretical Computer Science*, S. Goldwasser, Ed., Cambridge, MA, USA: Association for Computing Machinery, Jan. 2012, pp. 326–349. DOI: 10.1145/2090236.2090263.

[23] H. Lipmaa, R. Parisella, and J. Siim, "On knowledge-soundness of plonk in ROM from falsifiable assumptions," in *Advances in Cryptology – CRYPTO 2025, Part VII*, Y. T. Kalai and S. F. Kamara, Eds., ser. Lecture Notes in Computer Science, vol. 16006, Santa Barbara, CA, USA: Springer, Cham, Switzerland, Aug. 2025, pp. 362–395. DOI: 10.1007/978-3-032-01907-3_12.

[24] M. J. Schervish, *Theory of Statistics* (Springer Series in Statistics). New York, NY: Springer, 1995, ISBN: 978-0-387-94546-0. DOI: 10.1007/978-1-4612-4250-5. [Online]. Available: https://doi.org/10.1007/978-1-4612-4250-5.

[25] J. T. Schwartz, "Fast probabilistic algorithms for verification of polynomial identities," *J. ACM*, vol. 27, no. 4, pp. 701–717, Oct. 1980, ISSN: 0004-5411. DOI: 10.1145/322217.322225. [Online]. Available: https://doi.org/10.1145/322217.322225.

[26] R. Zippel, "Probabilistic algorithms for sparse polynomials," in *Symbolic and Algebraic Computation*, E. W. Ng, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 1979, pp. 216–226, ISBN: 978-3-540-35128-3.

[27] V. Shoup, *Sequences of games: A tool for taming complexity in security proofs*, Cryptology ePrint Archive, Paper 2004/332, 2004. [Online]. Available: https://eprint.iacr.org/2004/332.

[28] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," *Journal of Cryptology*, vol. 17, no. 4, pp. 297–319, Sep. 2004. DOI: 10.1007/s00145-004-0314-9.

[29] S. Chatterjee, D. Hankerson, E. Knapp, and A. Menezes, "Comparing two pairing-based aggregate signature schemes," *Designs, Codes and Cryptography*, vol. 55, pp. 141–167, May 2010. DOI: 10.1007/s10623-009-9334-7.

[30] R. Gennaro and D. Wichs, "Fully homomorphic message authenticators," in *Advances in Cryptology - ASIACRYPT 2013*, K. Sako and P. Sarkar, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 301–320, ISBN: 978-3-642-42045-0.

[31] I. P. Razenshteyn, Z. Song, and D. P. Woodruff, "Weighted low rank approximations with provable guarantees," in *48th Annual ACM Symposium on Theory of Computing*, D. Wichs and Y. Mansour, Eds., Cambridge, MA, USA: ACM Press, Jun. 2016, pp. 250–263. DOI: 10.1145/2897518.2897639.

[32] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *The Annals of Statistics*, vol. 32, no. 2, Apr. 2004, ISSN: 0090-5364. DOI: 10.1214/009053604000000067. [Online]. Available: http://dx.doi.org/10.1214/009053604000000067.

[33] D. Fiore, A. Mitrokotsa, L. Nizzardo, and E. Pagnin, *Multi-key homomorphic authenticators*, Cryptology ePrint Archive, Paper 2016/804, 2016. [Online]. Available: https://eprint.iacr.org/2016/804.

[34] J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit, "Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting," in *Advances in Cryptology – EUROCRYPT 2016, Part II*, M. Fischlin and J.-S. Coron, Eds., ser. Lecture Notes in Computer Science, vol. 9666, Vienna, Austria: Springer Berlin Heidelberg, Germany, May 2016, pp. 327–357. DOI: 10.1007/978-3-662-49896-5_12.

[35] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," in *2018 IEEE Symposium on Security and Privacy*, San Francisco, CA, USA: IEEE Computer Society Press, May 2018, pp. 315–334. DOI: 10.1109/SP.2018.00020.

[36] A. Chiesa, L. Chua, and M. Weidner, "On cycles of pairing-friendly elliptic curves," *SIAM Journal on Applied Algebra and Geometry*, vol. 3, no. 2, pp. 175–192, 2019. DOI: 10.1137/18M1173708. eprint: https://doi.org/10.1137/18M1173708. [Online]. Available: https://doi.org/10.1137/18M1173708.

[37] L. Schabhüser, D. Butin, and J. Buchmann, "Context hiding multi-key linearly homomorphic authenticators," in *Topics in Cryptology – CT-RSA 2019*, M. Matsui, Ed., ser. Lecture Notes in Computer Science, vol. 11405, San Francisco, CA, USA: Springer, Cham, Switzerland, Mar. 2019, pp. 493–513. DOI: 10.1007/978-3-030-12612-4_25.

A

Proof of Theorem 6.1

We present the full security proof of the compressed construction $\mathsf{mkqhs\text{-}\widetilde{br}}$ from Chapter 6, given in Theorem 6.1. The proof follows the structure of Theorem 5.1 closely, with the main differences being described in Subsection 6.1.3.

Proof of Theorem 6.1. We distinguish between the three types of forgeries from Definition 3.3. We also define a reduction $\mathcal{B}$ with identical simulation to the one of Subsection 4.2.2 and whose correctness follows from Claim 4.3.

The only addition to the simulation is a random oracle $\mathcal{H}_\rho : \{0,1\}^* \to (\mathbb{Z}_q^*)^{R\times 2}$, which answers queries with uniformly random values while maintaining consistency across repeated queries. Since H_ρ is not used inside Sign, $\mathcal{H}_\rho$ is not subject to the delayed-hash restriction of the $\mathsf{HomUF\text{-}CMA\text{-}DHQ}$ experiment, and Claim 4.3 still applies.

We start by noting that with the same reasoning as in Theorem 4.1, all Type-1 forgeries reduce to Type-3 forgeries, so we disregard them in our analysis.

A.1 co-CDH* Extraction From Type-2 Forgeries

Let $\mathcal{A}$ output a successful Type-2 forgery $(\mathcal{P}^*, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}^*}, \tilde{m}^*, \tilde{\sigma}^*)$ after its interaction with $\mathcal{B}$, where $\mathcal{P}^* = (f^*, \ell_1^*, \ldots, \ell_n^*)$ and $f^* = \left(\{a_i^*, \mathbf{u}_i^*, \mathbf{v}_i^*\}_{i=1}^n\right)$.

By definition of a Type-2 forgery,

$$\mathsf{Verify}(\mathcal{P}^*, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}^*}, \tilde{m}^*, \tilde{\sigma}^*) = 1 \quad \text{and} \quad \tilde{m}^* \neq f^*(m_1, \ldots, m_n),$$

where m_i are the message queried during the sign query phase for labels ℓ_i^*. By the same reasoning as in Theorem 4.1, for every label ℓ_i^* appearing in $\mathcal{P}^*$, $\mathcal{H}(\ell_i^*)$ was programmed during the signing phase and not the hash phase.

$\mathcal{B}$ first parses $\tilde{\sigma}^* = \left(\tilde{\gamma}^*, \{\tilde{\mu}_{\mathsf{id}}^*\}_{\mathsf{id}\in\mathcal{P}^*}, \boldsymbol{\mu}^{(u)*}, \boldsymbol{\mu}^{(v)*}\right)$, $\tilde{\gamma}^* = \left(\gamma^{(a)*}, \boldsymbol{\gamma}^{(u)*}, \boldsymbol{\gamma}^{(v)*}\right)$, with $\tilde{\mu}_{\mathsf{id}}^* = \left(\mu_{\mathsf{id}}^{(a)*}, \tilde{\mu}_{\mathsf{id}}^{(u,v)*}\right)$ for each $\mathsf{id} \in \mathcal{P}^*$. $\mathcal{B}$ then obtains

$$(\boldsymbol{\rho}, \boldsymbol{\rho}') = \mathcal{H}_\rho\left(\mathcal{P}^*, \tilde{\gamma}^*, \{\mu_{\mathsf{id}}^{(a)*}\}_{\mathsf{id}\in\mathcal{P}^*}, \boldsymbol{\mu}^{(u)*}, \boldsymbol{\mu}^{(v)*}\right),$$

where $\boldsymbol{\rho} = (\rho_1, \ldots, \rho_R)^\top$ and $\boldsymbol{\rho}' = (\rho'_1, \ldots, \rho'_R)^\top$.

Successful verification for ver_1 implies

$$\tilde{m}^* = \mu^{(a)*} + \sum_{r=1}^{R} \mu_r^{(u)*} \mu_r^{(v)*}, \tag{A.1}$$

where we write $\mu^{(a)*} = \sum_{\mathsf{id} \in \mathcal{P}^*} \mu_{\mathsf{id}}^{(a)*}$.

We define the honestly computed counterparts of the forgery's aggregated messages as

$$\mu^{(a)\circ} := \sum_{i=1}^{n} a_i^* m_i, \quad \mu_r^{(u)\circ} := \sum_{i=1}^{n} u_{i,r}^* m_i, \quad \mu_r^{(v)\circ} := \sum_{i=1}^{n} v_{i,r}^* m_i.$$

Since the forgery is Type-2,

$$\tilde{m}^* \neq \mu^{(a)\circ} + \sum_{r=1}^{R} \mu_r^{(u)\circ} \mu_r^{(v)\circ}. \tag{A.2}$$

Therefore, at least one of the following must hold:

Case 1: linear forgery. There exists some identity $\mathsf{id} \in \mathcal{P}^*$ such that $\mu_{\mathsf{id}}^{(a)*} \neq \sum_{i \in \mathcal{I}_{\mathsf{id}}} a_i^* m_i$. $\mathcal{B}$ extracts the co-CDH* solution from $\gamma^{(a)*}$ exactly as in Theorem 4.1, giving

$$\mathsf{Adv}_{\mathcal{B}}^{\text{co-CDH}^*}(\lambda) \;\geq\; \left(1 - \frac{1}{q-1}\right) \cdot \Pr[\mathcal{A} \text{ outputs a valid Type-2 forgery}].$$

Case 2: quadratic forgery. All linear aggregates are correct, *i.e.* $\mu_{\mathsf{id}}^{(a)*} = \sum_{i \in \mathcal{I}_{\mathsf{id}}} a_i^* m_i$ for all $\mathsf{id} \in \mathcal{P}^*$. Hence $\mu^{(a)*} = \mu^{(a)\circ}$, and from (A.1) and (A.2),

$$\sum_{r=1}^{R} \mu_r^{(u)*} \mu_r^{(v)*} \neq \sum_{r=1}^{R} \mu_r^{(u)\circ} \mu_r^{(v)\circ}.$$

The rest of this section handles extraction in the quadratic forgery case.

Identifying inconsistent identities. For each identity $\mathsf{id} \in \mathcal{P}^*$, we define the *compressed mismatch term*

$$\delta_{\mathsf{id}} := \tilde{\mu}_{\mathsf{id}}^{(u,v)*} - \sum_{r=1}^{R} \left(\rho_r \sum_{i \in \mathcal{I}_{\mathsf{id}}} u_{i,r}^* m_i + \rho_r' \sum_{i \in \mathcal{I}_{\mathsf{id}}} v_{i,r}^* m_i \right). \tag{A.3}$$

Each δ_{id} is computable by $\mathcal{B}$ from values recorded during the signing phase and $\tilde{\sigma}^*$.

We also define the set of *consistent* identities as $\mathcal{C} = \{\mathsf{id} \in \mathcal{P}^* \mid \delta_{\mathsf{id}} = 0\}$. By Claim 6.2, $\mathcal{P}^* \setminus \mathcal{C}$ is non-empty except with probability at most $\frac{1}{q-1}$.

Finding an expression for Γ^*_ρ. $\mathcal{B}$ computes

$$\Gamma^*_\rho = \prod_{r=1}^{R} (\gamma_r^{(u)*})^{\rho_r} (\gamma_r^{(v)*})^{\rho'_r}.$$

For each $\mathsf{id} \in \mathcal{P}^*$, define

$$\hat{A}_{\mathsf{id}} = \tilde{g}_1^{\tilde{\mu}_{\mathsf{id}}^{(u,v)*}} \cdot \prod_{i \in \mathcal{I}_{\mathsf{id}}} \mathcal{H}(\ell_i^*)^{\sum_{r=1}^{R} \left(\rho_r u_{i,r}^* + \rho'_r v_{i,r}^*\right)}. \tag{A.4}$$

From verification equation ver_3 and $\mathsf{pk}_{\mathsf{id}} = X_2^{y_{\mathsf{id}}}$, we get

$$e(\Gamma^*_\rho, g_2) = \prod_{\mathsf{id} \in \mathcal{P}^*} e(\hat{A}_{\mathsf{id}}, X_2^{y_{\mathsf{id}}}) = e\left(\prod_{\mathsf{id} \in \mathcal{P}^*} \hat{A}_{\mathsf{id}}^{x y_{\mathsf{id}}}, g_2 \right),$$

and by non-degeneracy of the pairing,

$$\Gamma^*_\rho = \prod_{\mathsf{id} \in \mathcal{P}^*} \hat{A}_{\mathsf{id}}^{x y_{\mathsf{id}}}. \tag{A.5}$$

Substituting the programmed hashes. Expanding (A.4) using $\tilde{g}_1 = g_1^s h$ and the programmed values $\mathcal{H}(\ell_i^*) = g_1^{t_i} h^{-m_i}$,

$$\begin{aligned}
\hat{A}_{\mathsf{id}} &= (g_1^s h)^{\tilde{\mu}_{\mathsf{id}}^{(u,v)*}} \cdot \prod_{i \in \mathcal{I}_{\mathsf{id}}} (g_1^{t_i} h^{-m_i})^{\sum_{r=1}^{R} (\rho_r u_{i,r}^* + \rho'_r v_{i,r}^*)} \\
&= g_1^{s\tilde{\mu}_{\mathsf{id}}^{(u,v)*} + \sum_{i \in \mathcal{I}_{\mathsf{id}}} t_i \sum_{r=1}^{R} (\rho_r u_{i,r}^* + \rho'_r v_{i,r}^*)} \cdot h^{\tilde{\mu}_{\mathsf{id}}^{(u,v)*} - \sum_{i \in \mathcal{I}_{\mathsf{id}}} m_i \sum_{r=1}^{R} (\rho_r u_{i,r}^* + \rho'_r v_{i,r}^*)}.
\end{aligned}$$

By definition of δ_{id} in (A.3), the exponent of h is exactly δ_{id}. We also define

$$\hat{a}_{\mathsf{id}} := s\tilde{\mu}_{\mathsf{id}}^{(u,v)*} + \sum_{i \in \mathcal{I}_{\mathsf{id}}} t_i \sum_{r=1}^{R} \left(\rho_r u_{i,r}^* + \rho'_r v_{i,r}^*\right),$$

which is computable by $\mathcal{B}$ from s and the stored values $t_i \in L_H$. Hence, $\hat{A}_{\mathsf{id}} = g_1^{\hat{a}_{\mathsf{id}}} h^{\delta_{\mathsf{id}}}$.

Extracting the co-CDH* solution. For identities $\mathsf{id} \in \mathcal{C}$, $\delta_{\mathsf{id}} = 0$ and h becomes $1_{\mathbb{G}_1}$. Splitting (A.5) into consistent and inconsistent identities therefore yields

$$\Gamma^*_\rho = \prod_{\mathsf{id} \in \mathcal{P}^*} X_1^{y_{\mathsf{id}} \hat{a}_{\mathsf{id}}} \cdot (h^x)^{\sum_{\mathsf{id} \in \mathcal{P}^*} y_{\mathsf{id}} \delta_{\mathsf{id}}}.$$

Hence, we define

$$\hat{B} := \prod_{\mathsf{id} \in \mathcal{P}^*} X_1^{y_{\mathsf{id}} \hat{a}_{\mathsf{id}}}, \qquad \hat{c} := \sum_{\mathsf{id} \in \mathcal{P}^*} y_{\mathsf{id}} \delta_{\mathsf{id}},$$

both computable by $\mathcal{B}$, giving $\Gamma^*_\rho = \hat{B} \cdot (h^x)^{\hat{c}}$. If $\hat{c} = 0$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ outputs

$$\omega = \left(\frac{\Gamma^*_\rho}{\hat{B}} \right)^{\hat{c}^{-1}} = h^x,$$

which solves the co-CDH* instance.

Bounding the abort probability. $\hat{c}$ may be viewed as a polynomial with degree at most 2, in the random variables $(\rho_r, \rho'_r)_{r\in[R]} \xleftarrow{\$} (\mathbb{Z}_q^*)^{2R}$ and $(y_{\mathsf{id}})_{\mathsf{id}\in\mathcal{P}^*} \xleftarrow{\$} (\mathbb{Z}_q^*)^{|\mathsf{id}\in\mathcal{P}^*|}$. By Claim 6.2, there exists some $\mathsf{id} \in \mathcal{P}^*$ such that $\mathsf{id} \notin \mathcal{C}$ with probability $1-1/(q-1)$, so $\delta_{\mathsf{id}} \neq 0$ for that identity, and the monomial $y_{\mathsf{id}}\delta_{\mathsf{id}}$ appears with a non-zero coefficient. Hence, $\hat{c}$ is not the zero polynomial, and by the Schwartz–Zippel Lemma,

$$\Pr[\hat{c} = 0 \mid \mathcal{P}^* \setminus \mathcal{C} \neq \varnothing] \leq \frac{2}{q-1}.$$

By the law of total probability and the trivially obtained bounds $\Pr[\mathcal{P}^* \setminus \mathcal{C} \neq \varnothing] \leq 1$ and $\Pr[\hat{c} = 0 \mid \mathcal{P}^* \setminus \mathcal{C} = \varnothing] \leq 1$, we get

$$\begin{aligned}\Pr[\hat{c} = 0] &= \Pr[\hat{c} = 0 \mid \mathcal{P}^* \setminus \mathcal{C} \neq \varnothing] \cdot \Pr[\mathcal{P}^* \setminus \mathcal{C} \neq \varnothing] \\ &\quad + \Pr[\hat{c} = 0 \mid \mathcal{P}^* \setminus \mathcal{C} = \varnothing] \cdot \Pr[\mathcal{P}^* \setminus \mathcal{C} = \varnothing] \\ &\leq \frac{2}{q-1} \cdot 1 + 1 \cdot \frac{1}{q-1} = \frac{3}{q-1}.\end{aligned}$$

Therefore

$$\mathsf{Adv}^{\text{co-CDH}^*}_{\mathcal{B}}(\lambda) \;\geq\; \left(1 - \frac{3}{q-1}\right) \cdot \Pr[\mathcal{A} \text{ outputs a valid Type-2 forgery}].$$

A.2 co-CDH* Extraction From Type-3 Forgeries

Let $\mathcal{A}$ output a successful Type-3 forgery $(\mathcal{P}^*, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}^*}, \tilde{m}^*, \tilde{\sigma}^*)$ where $\mathcal{P}^* = (f^*, \ell_1^*, \ldots, \ell_n^*)$ and $f^* = \left(\{a_i^*, \mathbf{u}_i^*, \mathbf{v}_i^*\}_{i=1}^n\right)$. We partition the labels as in Theorem 5.1, defining $\mathcal{U} = \{i \in [n] \mid (\cdot, \ell_i^*, \cdot, \cdot) \notin L_\sigma\}$, $\mathcal{I}_{\mathsf{id}} = \{i \in [n] \mid \ell_i^* = (\mathsf{id}, \cdot)\}$, and $\mathcal{U}_{\mathsf{id}} = \mathcal{I}_{\mathsf{id}} \cap \mathcal{U}$. Since the forgery is Type-3, $\mathcal{U} \neq \varnothing$.

By admissibility of f^*, exactly one of the following holds:

Case 1: linear forgery. For every $i \in \mathcal{U}$, $a_i^* \neq 0$. $\mathcal{B}$ extracts the co-CDH* solution from $\gamma^{(a)*}$ as in Theorem 4.1, giving

$$\mathsf{Adv}^{\text{co-CDH}^*}_{\mathcal{B}}(\lambda) \;\geq\; \left(1 - \frac{1}{q}\right) \cdot \Pr[\mathcal{A} \text{ outputs a valid Type-3 forgery in Case 1}].$$

Case 2: quadratic forgery. There exists $i \in \mathcal{U}$ with $a_i^* = 0$, and by admissibility $(u_{i,r}^*, v_{i,r}^*) \neq (0,0)$ for some $r \in [R]$. The rest of this section proceeds with extraction in this case.

$\mathcal{B}$ queries $\mathcal{H}_\rho$ to obtain $(\boldsymbol{\rho}, \boldsymbol{\rho}') = \mathcal{H}_\rho\left(\mathcal{P}^*, \tilde{\gamma}^*, \{\mu_{\mathsf{id}}^{(a)*}\}_{\mathsf{id}\in\mathcal{P}^*}, \boldsymbol{\mu}^{(u)*}, \boldsymbol{\mu}^{(v)*}\right)$. Next, we write

$$\theta_i^* := \sum_{r=1}^{R}\left(\rho_r u_{i,r}^* + \rho'_r v_{i,r}^*\right), \qquad \Theta_{\mathsf{id}}^* := \tilde{\mu}_{\mathsf{id}}^{(u,v)*},$$

where the only change from the mkqhs-br proof is that Θ_{id}^* is now the compressed scalar $\tilde{\mu}_{\mathsf{id}}^{(u,v)*}$ rather than $\langle\boldsymbol{\rho}, \boldsymbol{\mu}_{\mathsf{id}}^{(u)*}\rangle + \langle\boldsymbol{\rho}', \boldsymbol{\mu}_{\mathsf{id}}^{(v)*}\rangle$.

With these θ_i^* and Θ_{id}^* variables, the quadratic Type-3 forgery extraction of Theorem 5.1 goes through verbatim. The Schwartz–Zippel argument bounding $\Pr[\theta_i^* = 0]$ in that proof requires $(\boldsymbol{\rho}, \boldsymbol{\rho}')$ to be uniformly random and independent of the forgery coefficients $(u_{i,r}^*, v_{i,r}^*)_{r\in[R]}$, which holds in $\widetilde{\mathsf{mkqhs\text{-}br}}$ because $\mathcal{P}^*$ (and thus f^*) appears in the input to $\mathcal{H}_\rho$, forcing the adversary to commit to these coefficients before $(\boldsymbol{\rho}, \boldsymbol{\rho}')$ is determined. The extraction therefore yields

$$\mathsf{Adv}_{\mathcal{B}}^{\text{co-CDH}^*}(\lambda) \;\geq\; \left(1 - \frac{2}{q}\right) \cdot \Pr[\mathcal{A} \text{ outputs a valid Type-3 forgery in Case 2}].$$

A.3 Combining the Bounds

Applying the union bound over forgery types and cases, and substituting the abort probability bounds, gives

$$\mathsf{Adv}_{\mathcal{A},\,\widetilde{\mathsf{mkqhs\text{-}br}}}^{\mathsf{HomUF\text{-}CMA\text{-}DHQ}}(\lambda) \leq \left(\frac{q-1}{q-2} + \frac{q-1}{q-4} + \frac{q}{q-1} + \frac{q}{q-2}\right) \cdot \mathsf{Adv}_{\mathcal{B}}^{\text{co-CDH}^*}(\lambda).$$

Since the sum of the four fractions is at most 5 for all primes $q \geq 11$,

$$\mathsf{Adv}_{\mathcal{A},\,\widetilde{\mathsf{mkqhs\text{-}br}}}^{\mathsf{HomUF\text{-}CMA\text{-}DHQ}}(\lambda) \;\leq\; 5 \cdot \mathsf{Adv}_{\mathcal{B}}^{\text{co-CDH}^*}(\lambda).$$

This completes the proof of Theorem 6.1.

□

B

Proof of Proposition 6.3

We give the proof of Proposition 6.3 from Subsection 6.2.3.

Proof of Proposition 6.3. We begin by handling the case when d is even, for which we may write (6.7) on the form

$$\|\mathbf{x}-\mathbf{y}\|^2 = \sum_{r=1}^{d/2} (m_{2r-1} - m_{2r-1+d})^2 + (m_{2r} - m_{2r+d})^2. \tag{B.1}$$

For each summand in (B.1), we note that (6.6) yields the rearrangement

$$\begin{aligned}(m_{2r-1} - m_{2r-1+d})^2 + (m_{2r} - m_{2r+d})^2 = \; & (m_{2r-1} + m_{2r} - m_{2r-1+d} + m_{2r+d}) \\ & \cdot (m_{2r-1} - m_{2r} - m_{2r-1+d} - m_{2r+d}) \\ & + 2m_{2r}^2 + 2m_{2r+d}^2.\end{aligned} \tag{B.2}$$

Each summand in (B.1) is therefore representable using one rank one product term together with two square coefficients, yielding $R = d/2 = \lceil d/2 \rceil$.

For the case of odd d, applying the above construction to the first $d-1$ messages yields $\sum_{i=1}^{d-1}(m_i - m_{i+d})^2$ using $(d-1)/2$ rank one terms. The remaining message contributes $(m_d - m_{2d})^2$ and may be represented using one additional rank one term of the form $u_d = v_d = 1$ and $u_{2d} = v_{2d} = -1$ with all other coefficients zero. Therefore, $R = (d-1)/2 + 1 = \lceil d/2 \rceil$ whenever d is odd.

Lastly, with the expression in (B.2) and accounting for the odd case, we identify the coefficients $b_i, u_{i,r}$, and $v_{i,r}$ for each message index as in Table 6.2. □

C

Proof of Theorem 6.4

We present the security proof of the message-squares construction mkqhs-br-m^2 from Chapter 6, given in Theorem 6.4. The proof follows the structure of Theorem 5.1 closely, the main differences being described in Subsection 6.2.6. Since this proof structure already has been presented a couple of times in previous chapters, we compress its presentation here.

Proof of Theorem 6.4. We define a reduction $\mathcal{B}$ identical to the one in Subsection 4.2.2 for Theorem 4.1, but with modifications described below. After simulating the HomUF-CMA-DHQ security experiment against an adversary $\mathcal{A}$, $\mathcal{B}$ splits returned forgeries into linear and quadratic type, then reduces each to extracting a co-CDH* solution with abort probability bounded by the Schwartz–Zippel lemma.

Simulation. The simulation follows Subsection 4.2.2 with the following changes to account for the two hash oracles $\mathcal{H}_1$ and $\mathcal{H}_2$. In the sign query phase, $\mathcal{B}$ now samples $t, t' \xleftarrow{\$} \mathbb{Z}_q$ and programs both oracles simultaneously as

$$\mathcal{H}_1(\ell) = g_1^t h^{-m} \quad \text{and} \quad \mathcal{H}_2(\ell) = g_1^{t'} h^{-m^2}, \tag{C.1}$$

storing both entries in L_H as

$$L_H \leftarrow L_H \cup (\ell, \mathcal{H}_1(\ell), 1, t) \quad \text{and} \quad L_H \leftarrow L_H \cup (\ell, \mathcal{H}_2(\ell), 2, t').$$

This allows $\mathcal{B}$ to compute the new signature component γ', analogously to γ, as

$$\gamma' = \left(X_1^{m^2 s + t'}\right)^{y_{\mathsf{id}}} \tag{C.2}$$

without knowledge of x.

For fresh hash queries of unsigned labels, $\mathcal{B}$ instead programs the random oracles as

$$\mathcal{H}_1(\ell) = h^r \quad \text{and} \quad \mathcal{H}_2(\ell) = h^{r'} \tag{C.3}$$

for independent $r, r' \xleftarrow{\$} \mathbb{Z}_q$.

Finally, we claim that the view of $\mathcal{A}$ is identically distributed to the real HomUF-CMA-DHQ experiment. By the argument of Claim 4.3, $\mathcal{H}_1$ is unchanged from $\mathcal{H}$ and is indistinguishable from a real random oracle, while the correctness of $\mathcal{H}_2$ follows by the same argument applied independently with $t' \xleftarrow{\$} \mathbb{Z}_q$.

It remains to verify that γ' is correctly distributed. In the real scheme, we have $\gamma' = (H_2(\ell) \cdot \tilde{g}_1^{m^2})^{\mathsf{sk}_{\mathsf{id}}}$. Substituting the programmed values from (C.1) along with $\tilde{g}_1 = g_1^s h$ and $\mathsf{sk}_{\mathsf{id}} = xy_{\mathsf{id}}$ gives

$$\gamma' = \left(g_1^{t'} h^{-m^2} \cdot (g_1^s h)^{m^2}\right)^{xy_{\mathsf{id}}} = (g_1^{sm^2+t'})^{xy_{\mathsf{id}}} = \left(X_1^{sm^2+t'}\right)^{y_{\mathsf{id}}},$$

which is exactly (C.2).

Extraction setup. Let $\mathcal{A}$ output a forgery $(\mathcal{P}^*, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}^*}, \tilde{m}^*, \tilde{\sigma}^*)$, where $\mathcal{P}^* = (f^*, \ell_1^*, \ldots, \ell_n^*)$ and $f^* = (\{a_i^*, b_i^*, \mathbf{u}_i^*, \mathbf{v}_i^*\}_{i=1}^n)$. $\mathcal{B}$ parses

$$\tilde{\sigma}^* = \left(\left(\gamma^{(a,b)*}, \boldsymbol{\gamma}^{(u)*}, \boldsymbol{\gamma}^{(v)*}\right), \left\{\mu_{\mathsf{id}}^{(a,b)*}, \boldsymbol{\mu}_{\mathsf{id}}^{(u)*}, \boldsymbol{\mu}_{\mathsf{id}}^{(v)*}\right\}_{\mathsf{id}\in\mathcal{P}^*}\right). \tag{C.4}$$

In the proofs of both Type-2 and Type-3 linear forgeries, the per-identity intermediary variable $\hat{A}_{\mathsf{id}}$ corresponding to ver_2 in Figure 6.3 now becomes

$$\hat{A}_{\mathsf{id}} = \tilde{g}_1^{\mu_{\mathsf{id}}^{(a,b)*}} \cdot \prod_{i\in\mathcal{I}_{\mathsf{id}}} \mathcal{H}_1(\ell_i^*)^{a_i^*}\, \mathcal{H}_2(\ell_i^*)^{b_i^*}, \tag{C.5}$$

and the non-degeneracy of the pairing together with ver_2 yields

$$\gamma^{(a,b)*} = \prod_{\mathsf{id}\in\mathcal{P}^*} \hat{A}_{\mathsf{id}}^{xy_{\mathsf{id}}}. \tag{C.6}$$

Forgery case split. Type-1 forgeries reduce to Type-3 forgeries by the same argument as in Theorem 4.1. For Type-2 and Type-3 forgeries, we split as follows.

For a *Type-2 forgery*, $\tilde{m}^* \neq f^*(m_1, \ldots, m_n)$ despite all labels being signed. Letting $\mu^{(a,b)\circ}$ and $\{\mu_r^{(u)\circ}, \mu_r^{(v)\circ}\}_{r=1}^R$ denote the honestly computed counterparts of the forged aggregates in (C.4), and defining the per-identity linear mismatch

$$\delta_{\mathsf{id}} := \mu_{\mathsf{id}}^{(a,b)*} - \sum_{i\in\mathcal{I}_{\mathsf{id}}} \left(a_i^* m_i + b_i^* m_i^2\right) = \mu_{\mathsf{id}}^{(a,b)*} - \mu_{\mathsf{id}}^{(a,b)\circ}, \tag{C.7}$$

the forgery falls in exactly one of the following cases:

Type-2 linear forgery case: Some $\mathsf{id} \in \mathcal{P}^*$ has $\delta_{\mathsf{id}} \neq 0$. $\mathcal{B}$ extracts from $\gamma^{(a,b)*}$ as below.

Type-2 quadratic forgery case: All $\delta_{\mathsf{id}} = 0$, so $\mu^{(a,b)*} = \mu^{(a,b)\circ}$, and therefore $\sum_r \mu_r^{(u)*}\mu_r^{(v)*} \neq \sum_r \mu_r^{(u)\circ}\mu_r^{(v)\circ}$. Since ver_3 is unchanged from mkqhs-br, $\mathcal{B}$ extracts from Γ_ρ^* exactly as in Theorem 5.1, yielding

$$\mathsf{Adv}_{\mathcal{B}}^{\text{co-CDH}^*}(\lambda) \;\geq\; \left(1 - \tfrac{2}{q-1}\right) \cdot \Pr[\mathcal{A} \text{ outputs a valid Type-2 quadratic forgery}].$$

For a *Type-3 forgery*, at least one label in $\mathcal{P}^*$ was never signed. Concretely, letting $\mathcal{U} = \{\, i \in [n] \mid (\cdot, \ell_i^*, \cdot, \cdot) \notin L_\sigma \,\}$ be the set of unsigned label indices, we have $\mathcal{U} \neq \varnothing$. By the admissibility of f^* as described in Subsection 6.2.2, every $i \in \mathcal{U}$ satisfies at least one of $(a_i^*, b_i^*) \neq (0,0)$ or $(u_{i,r}^*, v_{i,r}^*) \neq (0,0)$ for some $r \in [R]$. Hence, the forgery is exactly one of the following:

Type-3 linear forgery case: For every $i \in \mathcal{U}$, $(a_i^*, b_i^*) \neq (0,0)$. $\mathcal{B}$ extracts from $\gamma^{(a,b)*}$ as below.

Type-3 quadratic forgery case: Some $i \in \mathcal{U}$ has $a_i^* = b_i^* = 0$, so $(u_{i,r}^*, v_{i,r}^*) \neq (0,0)$ for that i and some r by admissibility. Since ver_3 is unchanged from mkqhs-br, $\mathcal{B}$ extracts from Γ_ρ^* exactly as in Theorem 5.1, yielding

$$\mathsf{Adv}_{\mathcal{B}}^{\text{co-CDH}^*}(\lambda) \;\geq\; \left(1 - \tfrac{2}{q}\right) \cdot \Pr[\mathcal{A} \text{ outputs a valid Type-3 quadratic forgery}].$$

Type-2 linear extraction. Expanding $\hat{A}_{\mathsf{id}}$ from (C.5) using $\tilde{g}_1 = g_1^s h$ and the programmed hashes from (C.1) yields $\hat{A}_{\mathsf{id}} = g_1^{\hat{a}_{\mathsf{id}}} \cdot h^{\delta_{\mathsf{id}}}$, where

$$\hat{a}_{\mathsf{id}} = s\mu_{\mathsf{id}}^{(a,b)*} + \sum_{i \in \mathcal{I}_{\mathsf{id}}} (a_i^* t_i + b_i^* t_i')$$

is computable by $\mathcal{B}$, and δ_{id} is the mismatch term from (C.7). Substituting into (C.6) gives $\gamma^{(a,b)*} = \hat{B} \cdot (h^x)^{\hat{c}}$, where

$$\hat{B} := \prod_{\mathsf{id} \in \mathcal{P}^*} X_1^{y_{\mathsf{id}} \hat{a}_{\mathsf{id}}}, \qquad \hat{c} := \sum_{\mathsf{id} \in \mathcal{P}^*} y_{\mathsf{id}}\, \delta_{\mathsf{id}}, \tag{C.8}$$

both computable by $\mathcal{B}$. If $\hat{c} = 0$, $\mathcal{B}$ aborts. Else, $\mathcal{B}$ outputs $\omega = (\gamma^{(a,b)*}/\hat{B})^{\hat{c}^{-1}} = h^x$.

It remains to bound $\Pr[\hat{c} = 0]$. Since we are in the linear forgery case, some $\delta_{\mathsf{id}} \neq 0$ and $\hat{c}$ in (C.8) is a non-zero linear polynomial in $\{y_{\mathsf{id}}\} \xleftarrow{\$} (\mathbb{Z}_q^*)^{|\mathsf{id} \in \mathcal{P}|}$. Hence, by the Schwartz–Zippel lemma,

$$\mathsf{Adv}_{\mathcal{B}}^{\text{co-CDH}^*}(\lambda) \;\geq\; \left(1 - \tfrac{1}{q-1}\right) \cdot \Pr[\mathcal{A} \text{ outputs a valid Type-2 linear forgery}].$$

Type-3 linear extraction. For each $\mathsf{id} \in \mathcal{P}^*$ let $\mathcal{U}_{\mathsf{id}} = \mathcal{I}_{\mathsf{id}} \cap \mathcal{U}$ be the unsigned indices for that identity. Expanding $\hat{A}_{\mathsf{id}}$ from (C.5) with hashes (C.1) for signed indices $i \in \mathcal{I}_{\mathsf{id}} \setminus \mathcal{U}_{\mathsf{id}}$, and hashes (C.3) for unsigned indices $i \in \mathcal{U}_{\mathsf{id}}$, yields $\hat{A}_{\mathsf{id}} = g_1^{\hat{a}_{\mathsf{id}}} h^{\hat{b}_{\mathsf{id}}}$, where

$$\begin{aligned}
\hat{a}_{\mathsf{id}} &:= s\,\mu_{\mathsf{id}}^{(a,b)*} + \sum_{i \in \mathcal{I}_{\mathsf{id}} \setminus \mathcal{U}_{\mathsf{id}}} \left(a_i^* t_i + b_i^* t_i'\right), \\
\hat{b}_{\mathsf{id}} &:= \mu_{\mathsf{id}}^{(a,b)*} - \sum_{i \in \mathcal{I}_{\mathsf{id}} \setminus \mathcal{U}_{\mathsf{id}}} \left(a_i^* m_i + b_i^* m_i^2\right) + \sum_{i \in \mathcal{U}_{\mathsf{id}}} \left(a_i^* r_i + b_i^* r_i'\right),
\end{aligned}$$

both computable by $\mathcal{B}$. Substituting into (C.6) gives $\gamma^{(a,b)*} = \hat{B} \cdot (h^x)^{\hat{c}}$, where

$$\hat{B} := \prod_{\mathsf{id} \in \mathcal{P}^*} X_1^{y_{\mathsf{id}} \hat{a}_{\mathsf{id}}}, \qquad \hat{c} := \sum_{\mathsf{id} \in \mathcal{P}^*} y_{\mathsf{id}}\, \hat{b}_{\mathsf{id}}, \tag{C.9}$$

both computable by $\mathcal{B}$. If $\hat{c} = 0$, $\mathcal{B}$ aborts. Else, $\mathcal{B}$ outputs $\omega := (\gamma^{(a,b)*}/\hat{B})^{\hat{c}^{-1}} = h^x$.

To bound $\Pr[\hat{c} = 0]$, we expand $\hat{c}$ from (C.9) using $\hat{b}_{\mathsf{id}}$, yielding

$$\hat{c} = \sum_{\mathsf{id} \in \mathcal{P}^*} y_{\mathsf{id}} \left(\mu_{\mathsf{id}}^{(a,b)*} - \sum_{i \in \mathcal{I}_{\mathsf{id}} \setminus \mathcal{U}_{\mathsf{id}}} (a_i^* m_i + b_i^* m_i^2) + \sum_{i \in \mathcal{U}_{\mathsf{id}}} (a_i^* r_i + b_i^* r_i') \right).$$

We condition on $\{y_{\mathsf{id}}\}$ fixed to an arbitrary outcome and view $\hat{c}$ as a polynomial in $\{r_i, r_i'\}_{i \in \mathcal{U}} \xleftarrow{\$} \mathbb{Z}_q^{2|\mathcal{U}|}$. For any $i \in \mathcal{U}$ with $\ell_i^* = (\mathsf{id}, \cdot)$, the coefficients of r_i and r_i' in $\hat{c}$ are $y_{\mathsf{id}} a_i^*$ and $y_{\mathsf{id}} b_i^*$ respectively. In the Type-3 linear forgery case, $(a_i^*, b_i^*) \neq (0, 0)$ and $y_{\mathsf{id}} \in \mathbb{Z}_q^*$, so at least one coefficient is non-zero and $\hat{c}$ is not the zero polynomial. By the Schwartz–Zippel lemma with domain $\mathbb{Z}_q$ and the law of total probability as in Theorem 4.1, $\Pr[\hat{c} = 0] \leq 1/q$, hence

$$\mathsf{Adv}_{\mathcal{B}}^{\text{co-CDH}^*}(\lambda) \;\geq\; \left(1 - \tfrac{1}{q}\right) \cdot \Pr[\mathcal{A} \text{ outputs a valid Type-3 linear forgery}].$$

Combining the bounds. Type-1 forgeries reduce to Type-3 forgeries by the same argument as in Theorems 4.1. Applying the union bound over the four cases gives

$$\begin{aligned} \mathsf{Adv}_{\mathcal{A},\,\mathsf{mkqhs\text{-}br\text{-}m^2}}^{\mathsf{HomUF\text{-}CMA\text{-}DHQ}}(\lambda) \;&\leq\; \left(\frac{q-1}{q-2} + \frac{q-1}{q-3} + \frac{q}{q-1} + \frac{q}{q-2} \right) \cdot \mathsf{Adv}_{\mathcal{B}}^{\text{co-CDH}^*}(\lambda) \\ &\leq\; 5 \cdot \mathsf{Adv}_{\mathcal{B}}^{\text{co-CDH}^*}(\lambda), \end{aligned}$$

for all primes $q \geq 11$. This completes the proof of Theorem 6.4.

□

D

Further Compression of mkqhs-$\widetilde{\text{br}}$

$\mathsf{Eval}(\mathcal{P}, \{\sigma_i\}_{i=1}^n)$

parse $\mathcal{P} = (f, \{\ell_i\}_{i=1}^n)$
parse $f = (\{a_i, \mathbf{u}_i, \mathbf{v}_i\}_{i=1}^n)$
parse $\mathbf{u}_i = (u_{i,1}, \ldots, u_{i,R})^\top$
parse $\mathbf{v}_i = (v_{i,1}, \ldots, v_{i,R})^\top$
parse $\sigma_i = (\mathsf{id}_i, \gamma_i, \mu_i)$
$\gamma^{(a)} = \prod_{i=1}^n \gamma_i^{a_i}$
for $\mathsf{id} \in \mathcal{P}$
  $\mathcal{I}_{\mathsf{id}} = \{\, i \in [n] \mid \ell_i = (\mathsf{id}, \cdot) \,\}$
  $\mu_{\mathsf{id}}^{(a)} = \sum_{i \in \mathcal{I}_{\mathsf{id}}} a_i \mu_i$
$\boldsymbol{\mu}^{(u)} = \sum_{i=1}^n \mu_i \, \mathbf{u}_i, \;\; \boldsymbol{\mu}^{(v)} = \sum_{i=1}^n \mu_i \, \mathbf{v}_i$
$(\boldsymbol{\rho}, \boldsymbol{\rho}') \leftarrow H_\rho\Big(\mathcal{P}, \gamma^{(a)}, \{\mu_{\mathsf{id}}^{(a)}\}_{\mathsf{id} \in \mathcal{P}}, \boldsymbol{\mu}^{(u)}, \boldsymbol{\mu}^{(v)}\Big)$
for $\mathsf{id} \in \mathcal{P}$
  $\boldsymbol{\mu}_{\mathsf{id}}^{(u)} = \sum_{i \in \mathcal{I}_{\mathsf{id}}} \mu_i \, \mathbf{u}_i, \;\; \boldsymbol{\mu}_{\mathsf{id}}^{(v)} = \sum_{i \in \mathcal{I}_{\mathsf{id}}} \mu_i \, \mathbf{v}_i$
  $\tilde{\mu}_{\mathsf{id}}^{(u,v)} = \langle \boldsymbol{\rho}, \boldsymbol{\mu}_{\mathsf{id}}^{(u)} \rangle + \langle \boldsymbol{\rho}', \boldsymbol{\mu}_{\mathsf{id}}^{(v)} \rangle$
  $\tilde{\gamma}_{\mathsf{id}}^{(u,v)} = \prod_{i \in \mathcal{I}_{\mathsf{id}}} \gamma_i^{\langle \boldsymbol{\rho}, \mathbf{u}_i \rangle + \langle \boldsymbol{\rho}', \mathbf{v}_i \rangle}$
  $\tilde{\mu}_{\mathsf{id}} = (\mu_{\mathsf{id}}^{(a)}, \tilde{\mu}_{\mathsf{id}}^{(u,v)})$
return $\tilde{\sigma} = \Big(\gamma^{(a)}, \{\tilde{\gamma}_{\mathsf{id}}^{(u,v)}, \tilde{\mu}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}}, \boldsymbol{\mu}^{(u)}, \boldsymbol{\mu}^{(v)}\Big)$

$\mathsf{Verify}(\mathcal{P}, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}}, \tilde{m}, \tilde{\sigma})$

parse $\mathcal{P} = (f, \{\ell_i\}_{i=1}^n), \;\; f = (\{a_i, \mathbf{u}_i, \mathbf{v}_i\}_{i=1}^n)$
parse $\tilde{\sigma} = \Big(\gamma^{(a)}, \{\tilde{\gamma}_{\mathsf{id}}^{(u,v)}, \tilde{\mu}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}}, \boldsymbol{\mu}^{(u)}, \boldsymbol{\mu}^{(v)}\Big), \;\; \tilde{\mu}_{\mathsf{id}} = (\mu_{\mathsf{id}}^{(a)}, \tilde{\mu}_{\mathsf{id}}^{(u,v)})$
$(\boldsymbol{\rho}, \boldsymbol{\rho}') \leftarrow H_\rho\Big(\mathcal{P}, \gamma^{(a)}, \{\mu_{\mathsf{id}}^{(a)}\}_{\mathsf{id} \in \mathcal{P}}, \boldsymbol{\mu}^{(u)}, \boldsymbol{\mu}^{(v)}\Big)$
$\mathsf{ver}_1 \leftarrow \left[\tilde{m} == \sum_{\mathsf{id} \in \mathcal{P}} \mu_{\mathsf{id}}^{(a)} + \langle \boldsymbol{\mu}^{(u)}, \boldsymbol{\mu}^{(v)} \rangle\right]$
$\mathsf{ver}_2 \leftarrow \left[e(\gamma^{(a)}, g_2) == \prod_{\mathsf{id} \in \mathcal{P}} e\left(g_1^{\mu_{\mathsf{id}}^{(a)}} \cdot \prod_{i \in \mathcal{I}_{\mathsf{id}}} H(\ell_i)^{a_i}, \mathsf{pk}_{\mathsf{id}}\right)\right]$
$\mathsf{ver}_3 \leftarrow \bigwedge_{\mathsf{id} \in \mathcal{P}} \left[e(\gamma_{\mathsf{id}}^{(u,v)}, g_2) == e\left(\tilde{g}_1^{\tilde{\mu}_{\mathsf{id}}^{(u,v)}} \cdot \prod_{i \in \mathcal{I}_{\mathsf{id}}} H(\ell_i)^{\langle \boldsymbol{\rho}, \mathbf{u}_i \rangle + \langle \boldsymbol{\rho}', \mathbf{v}_i \rangle}, \mathsf{pk}_{\mathsf{id}}\right)\right]$
$\mathsf{ver}_4 \leftarrow \left[\sum_{\mathsf{id} \in \mathcal{P}} \tilde{\mu}_{\mathsf{id}}^{(u,v)} == \langle \boldsymbol{\rho}, \boldsymbol{\mu}^{(u)} \rangle + \langle \boldsymbol{\rho}', \boldsymbol{\mu}^{(v)} \rangle\right]$
return $[\mathsf{ver}_1 \wedge \mathsf{ver}_2 \wedge \mathsf{ver}_3 \wedge \mathsf{ver}_4]$

Figure D.1: The Eval and Verify algorithms for the per-identity compressed γ version of mkqhs-$\widetilde{\text{br}}$. Its Setup, KeyGen, and Sign algorithms are as mklhs_3 in Figure 5.1.

This appendix chapter supports the further work suggestion in Section 9.2, specifically the first step of the inner-product argument approach to full-rank succinctness. As explained there, the R-dependence from the $\mathbb{G}_1$ elements of mkqhs-$\widetilde{\text{br}}$ can be removed by replacing the per-rank pairs $\{\gamma_r^{(u)}, \gamma_r^{(v)}\}_{r=1}^{R}$ with a per-identity element $\tilde{\gamma}_{\mathsf{id}}^{(u,v)}$. We give the resulting Eval and Verify algorithms in Figure D.1 and prove that this compression preserves HomUF-CMA-DHQ security under co-CDH*.

Proposition D.1. *The per-identity compressed variant of* mkqhs-$\widetilde{\text{br}}$ *in Figure D.1 is* HomUF-CMA-DHQ *secure under the co-CDH* assumption in the random oracle model, with the same advantage bound as* mkqhs-$\widetilde{\text{br}}$.

Proof. We define a reduction $\mathcal{B}$ with simulation identical to that of Theorem 6.1, whose correctness follows from Claim 4.3. Type-1 forgeries reduce to Type-3 forgeries by the same argument as in Theorem 4.1 and are disregarded. For Type-2 and Type-3 forgeries, the linear forgery cases are handled exactly as in Theorem 6.1, extracting the co-CDH* solution from $\gamma^{(a)*}$. It remains to handle the quadratic forgery cases.

Claim 6.2 still holds. The only change from mkqhs-$\widetilde{\text{br}}$ affecting Claim 6.2 is that the per-rank elements $\boldsymbol{\gamma}^{(u)}, \boldsymbol{\gamma}^{(v)}$ are no longer included in the input to the random oracle $\mathcal{H}_\rho$. However, the proof of the claim depends on that $\boldsymbol{\mu}^{(u)*}$ and $\boldsymbol{\mu}^{(v)*}$ appear in the input, which remain in Figure D.1. Hence, the relation (6.4) and its Schwartz–Zippel bound are unchanged.

Type-2 quadratic extraction. Fix $\bar{\mathsf{id}} \in \mathcal{P}^*$ with $\delta_{\bar{\mathsf{id}}} \neq 0$, which occurs with probability at most $1/(q-1)$ by Claim 6.2. Since $\mathsf{pk}_{\bar{\mathsf{id}}} = X_2^{y_{\bar{\mathsf{id}}}} = g_2^{x y_{\bar{\mathsf{id}}}}$, bilinearity and non-degeneracy on ver_3, yield $\tilde{\gamma}_{\bar{\mathsf{id}}}^{(u,v)*} = X_1^{\hat{a}_{\bar{\mathsf{id}}} y_{\bar{\mathsf{id}}}} \cdot (h^x)^{y_{\bar{\mathsf{id}}}\delta_{\bar{\mathsf{id}}}}$ with

$$\hat{a}_{\bar{\mathsf{id}}} = s\tilde{\mu}_{\bar{\mathsf{id}}}^{(u,v)*} + \sum_{i \in \mathcal{I}_{\bar{\mathsf{id}}}} t_i \sum_{r=1}^{R} \big(\rho_r u_{i,r}^* + \rho_r' v_{i,r}^*\big). \tag{D.1}$$

Since $\delta_{\bar{\mathsf{id}}} \neq 0$, $\mathcal{B}$ may output

$$\omega := \left(\frac{\tilde{\gamma}_{\bar{\mathsf{id}}}^{(u,v)*}}{X_1^{\hat{a}_{\bar{\mathsf{id}}} y_{\bar{\mathsf{id}}}}}\right)^{(y_{\bar{\mathsf{id}}}\delta_{\bar{\mathsf{id}}})^{-1}} = h^x.$$

Type-3 quadratic extraction. Since $\mathcal{U} \neq \varnothing$, fix $\bar{\mathsf{id}} \in \mathcal{P}^*$ with $\mathcal{U}_{\bar{\mathsf{id}}} \neq \varnothing$. For $i \in \mathcal{I}_{\bar{\mathsf{id}}} \setminus \mathcal{U}_{\bar{\mathsf{id}}}$ hashes were programmed as $\mathcal{H}(\ell_i^*) = g_1^{t_i} h^{-m_i}$, and for $i \in \mathcal{U}_{\bar{\mathsf{id}}}$ as $\mathcal{H}(\ell_i^*) = h^{r_i}$. The ver_3 equation and $\mathsf{pk}_{\bar{\mathsf{id}}} = X_2^{y_{\bar{\mathsf{id}}}}$ then give $\tilde{\gamma}_{\bar{\mathsf{id}}}^{(u,v)*} = X_1^{\hat{a}_{\bar{\mathsf{id}}} y_{\bar{\mathsf{id}}}} \cdot (h^x)^{y_{\bar{\mathsf{id}}}\hat{b}_{\bar{\mathsf{id}}}}$ with $\hat{a}$ as (D.1) and

$$\hat{b}_{\bar{\mathsf{id}}} = \tilde{\mu}_{\bar{\mathsf{id}}}^{(u,v)*} - \sum_{i \in \mathcal{I}_{\bar{\mathsf{id}}} \setminus \mathcal{U}_{\bar{\mathsf{id}}}} m_i \sum_{r=1}^{R} \big(\rho_r u_{i,r}^* + \rho_r' v_{i,r}^*\big) + \sum_{i \in \mathcal{U}_{\bar{\mathsf{id}}}} r_i \sum_{r=1}^{R} \big(\rho_r u_{i,r}^* + \rho_r' v_{i,r}^*\big).$$

By the same Schwartz–Zippel and admissibility argument as in the proof of Theorem 5.1, $\hat{b}_{\bar{\mathsf{id}}} = 0$ with probability at most $2/q$, and whenever $\hat{b}_{\bar{\mathsf{id}}} \neq 0$, $\mathcal{B}$ may output

$$\omega := \left(\frac{\tilde{\gamma}_{\bar{\mathsf{id}}}^{(u,v)*}}{X_1^{\hat{a}_{\bar{\mathsf{id}}} y_{\bar{\mathsf{id}}}}}\right)^{(y_{\bar{\mathsf{id}}}\hat{b}_{\bar{\mathsf{id}}})^{-1}} = h^x. \qquad \square$$