CHQS: Publicly Verifiable Homomorphic Signatures Beyond the Linear Case

Lucas Schabhüser, Denis Butin, and Johannes Buchmann

Technische Universität Darmstadt, Darmstadt, Germany
{lschabhueser,dbutin,buchmann}@cdc.informatik.tu-darmstadt.de

Abstract. Sensitive data is often outsourced to cloud servers, with the server performing computation on the data. Computational correctness must be efficiently verifiable by a third party while the input data remains confidential. This paper introduces CHQS, a homomorphic signature scheme from bilinear groups fulfilling these requirements. CHQS is the first such scheme to be both context hiding and publicly verifiable for arithmetic circuits of degree two. It also achieves amortized efficiency: after a precomputation, verification can be faster than the evaluation of the circuit itself.

1 Introduction

Today, it is common practice to outsource time-consuming computations to the cloud. In such a situation, it is desirable to be able to verify the outsourced computation. The verification must be *efficient*, by which we mean that the verification procedure is significantly faster than verified computation itself. Otherwise, the verifier could as well carry out the computation by himself, negating the advantage of outsourcing.

In addition, there are scenarios in which the verification is required to be *context hiding*, which refers to the verification not revealing anything about the input to the computation. For instance, consider a cloud service which collects signed health data of individuals and computes statistics on them. These statistical evaluations are then given to a third party: an insurance company, which does not trust the cloud service to provide correct statistics. As a consequence, the third party must be able to verify the statistical outcome. However, for privacy reasons, the third party must not be able to learn the individual health data. So the challenge arises to design efficient and context hiding verification procedures for outsourced computing.

Using a homomorphic signature scheme, the verification procedure for outsourced computations can be implemented as follows. The data owner uploads signed data to the cloud. The cloud server generates a signature on the computed function output from these signatures. The verifier uses this signature to check for correctness of the computation.

There are efficient and context hiding homomorphic signature schemes for linear functions (e.g. [8, 23]). However, linear functions are insufficient for many applications. For instance, statistics often require computing variance and covariance, which use

quadratic functions. Still, beyond the linear case, no efficient and context hiding homomorphic signature schemes are known. For quadratic functions, efficient verification is possible as shown in [3]. However, this scheme is not context hiding. For a more detailed overview of related work, we refer to Sec. 6.

Contribution In this paper, we solve the problem of providing efficient and context hiding verification for multivariate quadratic functions. The core component of our solution and our main contribution is the new homomorphic signature scheme CHQS (Context Hiding Quadratic Signatures). CHQS allows to generate a signature on the function value of a multivariate quadratic polynomial from signatures on the input values without knowledge of the signing key. CHQS is perfectly context hiding, i.e. the signature of the output value does not leak any information about the input values. Furthermore, verification time is linear (in an amortized sense). A trade-off of our approach is a signature size that grows during homomorphic evaluation, so our scheme is not succinct. Still, freshly generated signatures are of constant size. Like most solutions in this area, the CHQS construction is based on bilinear groups. However, CHQS showcases for the first time how to use such groups to simultaneously achieve both public verification and multiplicative depth.

Outline We recall relevant definitions for homomorphic signature schemes in Sec. 2. In Sec. 3, we present CHQS, our homomorphic signature scheme for multivariate polynomials of degree 2. We address its properties, notably correctness and context hiding in Sec. 4 and give a security reduction in Sec. 5. Next, in Sec. 6, we compare our contribution to existing work. Finally, in Sec. 7, we summarize our results and give an outlook to future work and open problems.

2 Homomorphic Signatures

In this section, we formally define homomorphic signature schemes and their relevant properties. Intuitively, homomorphic signatures allow to generate new signatures from existing signatures without the knowledge of the secret signing key. It is necessary that the homomorphic property cannot be abused to create forgeries. In order to specify homomorphic signatures with strong security properties, the notions of labeled and multi-labeled programs (see e.g. [3]) are introduced. They enable security guarantees by restricting the signatures that may be homomorphically combined to new signatures.

A *labeled program* $\mathcal{P}$ consists of a tuple $(f, \tau_1, \ldots, \tau_n)$, where $f : \mathcal{M}^n \to \mathcal{M}$ is a function with n inputs and $\tau_i \in \mathsf{T}$ is a label for the i-th input of f from some set T. Given a set of labeled programs $\mathcal{P}_1, \ldots, \mathcal{P}_k$ and a function $g : \mathcal{M}^k \to \mathcal{M}$, they can be composed by evaluating g over the labeled programs, i.e. $\mathcal{P}^* = g(\mathcal{P}_1, \ldots, \mathcal{P}_k)$. The identity program with label τ is given by $\mathcal{I}_\tau = (f_{id}, \tau)$, where $f_{id} : \mathcal{M} \to \mathcal{M}$ is the

identity function. The program $\mathcal{P} = (f, \tau_1, \ldots, \tau_n)$ can be expressed as the composition of n identity programs $\mathcal{P} = f(\mathcal{I}_{\tau_1}, \ldots, \mathcal{I}_{\tau_n})$.

A *multi-labeled program* $\mathcal{P}_\Delta$ is a pair $(\mathcal{P}, \Delta)$ of the labeled program $\mathcal{P}$ and a dataset identifier Δ. Given a set of k multi-labeled programs with the same dataset identifier Δ, i.e. $(\mathcal{P}_1, \Delta), \ldots, (\mathcal{P}_k, \Delta)$, and a function $g : \mathcal{M}^k \to \mathcal{M}$, a composed multi-labeled program $\mathcal{P}^*_\Delta$ can be computed, consisting of the pair $(\mathcal{P}^*, \Delta)$, where $\mathcal{P}^* = g(\mathcal{P}_1, \ldots, \mathcal{P}_k)$. Analogously to the identity program for labeled programs, we refer to a multi-labeled identity program by $\mathcal{I}_{(\Delta,\tau)} = ((f_{id}, \tau), \Delta)$.

Definition 1 (Homomorphic Signature Scheme). *A homomorphic signature scheme is a tuple of the following probabilistic polynomial time (PPT) algorithms:*

$\mathsf{HKeyGen}(1^\lambda, n)$: *On input a security parameter λ and an integer n, the algorithm returns a key pair $(\mathsf{sk}, \mathsf{pk})$, where sk is the secret key kept private and pk is the public key which determines the message space $\mathcal{M}$, the signature space $\mathcal{Y}$, and the set $\mathcal{F}$ of admissible labeled programs $\mathcal{P} : \mathcal{M}^n \to \mathcal{M}$.*

$\mathsf{HSign}(\mathsf{sk}, \Delta, \tau, m)$: *On input a secret key sk, a dataset identifier Δ, an input identifier τ, and a message $m \in \mathcal{M}$, the algorithm returns a signature $\sigma \in \mathcal{Y}$ which is the signature for the message labeled by τ in the dataset identified by Δ.*

$\mathsf{HEval}(\mathsf{pk}, \mathcal{P}_\Delta, \boldsymbol{\sigma})$: *On input a public key pk, a multi-labeled program $\mathcal{P}_\Delta$, and a set of signatures $\boldsymbol{\sigma} \in \mathcal{Y}^k$, the algorithm returns a signature $\sigma' \in \mathcal{Y}$ for the multi-labeled program $\mathcal{P}$ over the (tuple of) signatures $\boldsymbol{\sigma}$ identified by Δ.*

$\mathsf{HVerify}(\mathsf{pk}, \mathcal{P}_\Delta, m, \sigma)$: *On input a public key pk, a multi-labeled program $\mathcal{P}_\Delta$, a message $m \in \mathcal{M}$, and a signature $\sigma \in \mathcal{Y}$, the algorithm either accepts the signature σ for the multi-labeled program $\mathcal{P}$ over the dataset identified by Δ, i.e. it returns $\mathtt{1}$, or rejects the signature, i.e. it returns $\mathtt{0}$.*

An obvious requirement of such a scheme is to be *correct*, i.e. fresh signatures created using the secret key should be authenticated, and homomorphically derived signatures should be verified under the correct function.

Definition 2 (Correctness). *A homomorphic signature scheme* ($\mathsf{HKeyGen}$, HSign, HEval, $\mathsf{HVerify}$) *is called* correct *if, for any security parameter λ, any integer n, and any key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{HKeyGen}(1^\lambda, n)$ the following two conditions are satisfied:*

1. *For any dataset identifier Δ, input identifier τ, and message $m \in \mathcal{M}$,*

$$\mathsf{HVerify}(\mathsf{pk}, \mathcal{I}_{(\Delta,\tau)}, m, \mathsf{HSign}(\mathsf{sk}, \Delta, \tau, m)) = 1.$$

2. *For any dataset identifier Δ, multi-labeled program $\mathcal{P}_\Delta = (f, \tau_1, \ldots, \tau_n, \Delta)$ containing a valid function f, and set of messages $\boldsymbol{m} \in \mathcal{M}^n$ with $\boldsymbol{m} = (m_1, \ldots, m_n)$,*

$$\mathsf{HVerify}(\mathsf{pk}, \mathcal{P}_\Delta, f(m_1, \ldots, m_n), \mathsf{HEval}(\mathsf{pk}, \mathcal{P}_\Delta, \boldsymbol{\sigma})) = 1$$

where $\boldsymbol{\sigma} = (\sigma_{\tau_1}, ..., \sigma_{\tau_n}) \in \mathcal{Y}^n$ with $\sigma_{\tau_i} \leftarrow \mathsf{HSign}(\mathsf{sk}, \Delta, \tau, m_{\tau_i})$ for all $i \in [n]$.

To formalise the security of a homomorphic signature scheme, we first provide a definition for *well defined programs*, which we need to define *forgeries* on these programs. Then, we introduce an experiment the attacker can run in order to make a successful forgery and present a definition for unforgeability based on this experiment. Due to their homomorphic properties, these schemes allow anyone to create signatures for messages not signed by the owner of the secret key. However, not only messages but also input- and dataset identifiers are used during signing. A verifier can thus always see whether a message has been signed by the owner of the secret key (by giving the identity function to $\mathsf{HVerify}$), or whether a signature has been homomorphically derived. Based on this, there exists a meaningful definition of unforgeability for homomorphic signatures. In order to present this, we first define *well defined programs*.

Definition 3 (Well Defined Program). *A labeled program $\mathcal{P} = (f, \tau_1, \ldots, \tau_n)$ is* well defined *with respect to a list $\mathcal{Q} \subset [n] \times \mathcal{M}$ if one of the two following cases holds: First, there are messages $m_1, \ldots, m_n$ such that $(\tau_i, m_i) \in \mathcal{Q}\ \forall i \in [n]$. Second, there is an $i \in \{1, \ldots, n\}$ such that $(\tau_i, \cdot) \notin \mathcal{Q}$ and $f(\{m_j\}_{(\tau_j, m_j) \in \mathcal{Q}} \cup \{m'_l\}_{(\tau_l, \cdot) \notin \mathcal{Q}})$ is constant over all possible choices of $m'_l \in \mathcal{M}$.*

Freeman pointed out [16] that it may generally not be possible to decide whether a multi-labeled program is well defined with regard to a list $\mathcal{Q}$. For this, we use this lemma:

Lemma 1 ([7]). *Let $\lambda, n, d \in \mathbb{N}$ and let F be the class of arithmetic circuits $f : \mathbb{F}^n \to \mathbb{F}$ over a finite field $\mathbb{F}$ of order p, such that the degree of f is at most d, for $\frac{d}{p} \leq \frac{1}{2}$. Then, there exists a PPT algorithm that for any given $f \in F$, decides if there exists $y \in \mathbb{F}$, such that $f(u) = y$ for all $u \in \mathbb{F}$ (i.e. if f is constant) and is correct with probability at least $1 - 2^{-\lambda}$.*

For the notion of unforgeability of a homomorphic signature scheme $\mathcal{H} = (\mathsf{HKeyGen}, \mathsf{HSign}, \mathsf{HEval}, \mathsf{HVerify})$ we use the following experiment between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ defined in [7]. During the experiment, the adversary $\mathcal{A}$ can adaptively query the challenger $\mathcal{C}$ for signatures on messages of his choice under identifiers of his choice.

Definition 4 ($\mathsf{HomUF} - \mathsf{CMA}_{\mathcal{A},\mathcal{H}}(\lambda)$ [7]).

Key Generation *$\mathcal{C}$ calls $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{HKeyGen}(1^\lambda, k)$ and gives pk to $\mathcal{A}$.*

Queries *$\mathcal{A}$ adaptively submits queries for (Δ, τ, m) where Δ is a dataset, τ is an input identifier, and m is a message. $\mathcal{C}$ proceeds as follows: if (Δ, τ, m) is the first query with dataset identifier Δ, it initializes an empty list $\mathcal{Q} = \emptyset$ for Δ. If $\mathcal{Q}$ does not contain a tuple $(\tau, \cdot)$, i.e. $\mathcal{A}$ never queried $(\Delta, \tau, \cdot)$, $\mathcal{C}$ calls $\sigma \leftarrow \mathsf{HSign}(\mathsf{sk}, \Delta, \tau, m)$, updates the list $\mathcal{Q} = \mathcal{Q} \cup (\tau, m)$, and gives σ to $\mathcal{A}$. If $(\tau, m) \in \mathcal{Q}$, then $\mathcal{C}$ returns*

the same signature σ as before. If $\mathcal{Q}$ already contains a tuple (τ, m') for $m \neq m'$, $\mathcal{C}$ returns $\perp$.

Forgery *$\mathcal{A}$ outputs a tuple $(\mathcal{P}_\Delta, m, \sigma)$. The experiment outputs $\mathtt{1}$ if $(\mathcal{P}_\Delta, m, \sigma)$ is a forgery in the following sense:*
A forgery is a tuple $(\mathcal{P}^*{}_{\Delta^*}, m^*, \sigma^*)$ such that $\mathsf{HVerify}(\mathsf{pk}, \mathcal{P}^*{}_{\Delta^*}, m^*, \sigma^*) = 1$ holds and exactly one of the following conditions is met:
Type 1: *The list $\mathcal{Q}$ was not initialized during the security experiment, i.e. no message was ever committed under the dataset identifier Δ.*
Type 2: *$\mathcal{P}^*{}_{\Delta^*}$ is well defined with respect to list $\mathcal{Q}$ and m^* is not the correct output of the computation, i.e. $m^* \neq f(\{m_j\}_{(\tau_j, m_j) \in \mathcal{Q}})$*
Type 3: *$\mathcal{P}^*{}_{\Delta^*}$ is not well defined with respect to $\mathcal{Q}$ (see Definition 3).*

Definition 5 (Unforgeability). *A homomorphic signature scheme $\mathcal{H}$ is* unforgeable *if for any PPT adversary $\mathcal{A}$, $Pr[\mathsf{HomUF-CMA}_{\mathcal{A},\mathcal{H}}(\lambda) = 1] = \mathsf{negl}(\lambda)$,*
where $\mathsf{negl}(\lambda)$ denotes any function negligible in the security parameter λ.

Additionally, we require the following statement to deal with Type-3 forgeries:

Lemma 2 ([7, Proposition 2]). *Let $\lambda \in \mathbb{N}$, and let $\mathcal{F}$ be the class of arithmetic circuits $f : \mathbb{F}_p^n \to \mathbb{F}$ such that the degree of f is at most d for $\frac{d}{p} < \frac{1}{2}$. Let $\mathcal{H} = (\mathsf{HKeyGen}, \mathsf{HSign}, \mathsf{HEval}, \mathsf{HVerify})$ be a homomorphic signature scheme with message space $\mathbb{F}_p$. Let $\mathcal{E}_b$ be the event that the adversary returns a Type-b forgery (for $b = 1, 2, 3$) in experiment $\mathsf{HomUF-CMA}_{\mathcal{A},\mathcal{H}}(\lambda)$. If for any adversary $\mathcal{A}$ we have $Pr[\mathsf{HomUF-CMA}_{\mathcal{A},\mathcal{H}}(\lambda) = 1 \ \wedge \ \mathcal{E}_2] \leq \epsilon$, then for any adversary $\mathcal{A}'$ producing a Type-3 forgery it holds that $Pr[\mathsf{HomUF-CMA}_{\mathcal{A}',\mathcal{H}}(\lambda) = 1 \ \wedge \ \mathcal{E}_3] \leq \epsilon + 2^{-\lambda}$.*

In order to use homomorphic signatures to improve bandwidth and computational effort further properties are desired, namely *succinctness* and *efficient verification*.

Definition 6 (Succinctness). *A homomorphic signature scheme* ($\mathsf{HKeyGen}$, HSign, HEval, $\mathsf{HVerify}$) *is called* succinct *if, for a fixed security parameter λ, the size of the signatures depends at most logarithmically on the dataset size n.*

Definition 7 (Efficient Verification [8]). *A homomorphic signature scheme for multi-labeled programs allows for* efficient verification *if there exist two additional algorithms* ($\mathsf{HVerPrep}, \mathsf{HEffVer}$) *such that:*

$\mathsf{HVerPrep}(\mathsf{pk}, \mathcal{P})$: *Given a public key pk and a labeled program $\mathcal{P} = (f, \tau_1, \ldots, \tau_n)$, generate a concise public key $\mathsf{pk}_\mathcal{P}$. This does not depend on a dataset identifier Δ.*
$\mathsf{HEffVer}(\mathsf{pk}_\mathcal{P}, m, \sigma, \Delta)$**:** *Given a concise public key $\mathsf{pk}_\mathcal{P}$, a message m, a signature σ and a dataset Δ, output $\mathtt{1}$ or $\mathtt{0}$.*

The above algorithms are required to satisfy the following two properties:

Correctness: *Let* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{HKeyGen}(1^\lambda, n)$ *be honestly generated keys and* $(\mathcal{P}, m, \sigma)$ *be a tuple such that, for* $\mathcal{P}_\Delta = (\mathcal{P}, \Delta)$, $\mathsf{HVerify}(\mathsf{pk}, \mathcal{P}_\Delta, m, \sigma) = 1$.
Then, for every $\mathsf{pk}_\mathcal{P} \xleftarrow{\$} \mathsf{HVerPrep}(\mathsf{pk}, \mathcal{P})$, $\mathsf{HEffVer}(\mathsf{pk}_\mathcal{P}, m, \sigma, \Delta) = 1$ *holds except with negligible probability.*

Amortized Efficiency: *Let* $\mathcal{P}$ *be a program, let* $m_1, \ldots, m_n$ *be valid input values and let* $t(n)$ *be the time required to compute* $\mathcal{P}(m_1, \ldots, m_n)$ *with output* m. *Then, for* $\mathsf{pk}_\mathcal{P} \xleftarrow{\$} \mathsf{HVerPrep}(\mathsf{pk}, \mathcal{P})$, *the time required to compute* $\mathsf{HEffVer}(\mathsf{pk}_\mathcal{P}, m, \sigma, \Delta)$ *is* $t' = o(t(n))$.

Here, *efficiency* is used in an amortized sense. There is a function-dependent pre-processing, so that the cost of verification amortizes over multiple datasets.

Finally, to derive additional privacy with regard to the verifier from using homomorphic signatures, we require a signature to the outcome of a computation not to leak information about the input values. Our definition is inspired by Gorbunov et al.'s definition [19]. However, in our case, the simulator is explicitly given the circuit for which the signature is supposed to verify. With respect to this difference, our definition is more general. We stress that the circuit is not hidden in either of the two context hiding notions.

Definition 8 (Context Hiding). *A homomorphic signature scheme for multi-labeled programs is called* context hiding *if there exist additional PPT procedures* $\tilde{\sigma} \leftarrow \mathsf{HHide}(\mathsf{pk}, m, \sigma)$ *and* $\mathsf{HHideVer}(\mathsf{pk}, \mathcal{P}_\Delta, m, \tilde{\sigma})$ *such that:*

Correctness: *For any* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{HKeyGen}(1^\lambda, n)$ *and any tuple* $(\mathcal{P}_\Delta, m, \sigma)$ *such that* $\mathsf{HVerify}(\mathsf{pk}, \mathcal{P}_\Delta, m, \sigma) = 1$ *and* $\tilde{\sigma} \leftarrow \mathsf{HHide}(\mathsf{pk}, m, \sigma)$, $\mathsf{HHideVer}(\mathsf{pk}, \mathcal{P}_\Delta, m, \tilde{\sigma}) = 1$.

Unforgeability: *The homomorphic signature scheme is unforgeable (see Definition 5) when replacing the algorithm* HVerify *with* HHideVer *in the security experiment.*

Context hiding security: *There is a simulator* Sim *such that, for any fixed (worst-case) choice of* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{HKeyGen}(1^\lambda, n)$, *any multi-labeled program* $\mathcal{P}_\Delta = (f, \tau_1, \ldots, \tau_n, \Delta)$, *messages* $m_1, \ldots, m_n$, *and distinguisher* $\mathcal{D}$ *there exists a function* $\epsilon(\lambda)$ *such that:*

$$|Pr[\mathcal{D}(I, \mathsf{HHide}(\mathsf{pk}, m, \sigma)) = 1] - Pr[\mathcal{D}(I, \mathsf{Sim}(\mathsf{sk}, \mathcal{P}_\Delta, m)) = 1]| = \epsilon(\lambda)$$

where $I = (\mathsf{sk}, \mathsf{pk}, \mathcal{P}_\Delta, \{m_i\}_{i=1}^n, m, \sigma)$ *for* $\sigma_i \leftarrow \mathsf{HSign}(\mathsf{sk}, \Delta, \tau_i, m_i)$, $m \leftarrow f(m_1, \ldots, m_n)$, $\sigma \leftarrow \mathsf{HEval}(\mathsf{pk}, \mathcal{P}_\Delta, \sigma_1, \ldots, \sigma_n)$,*and the probabilities are taken over the randomness of* HSign, HHide *and* Sim. *If* $\epsilon(\lambda) = \mathsf{negl}(\lambda)$, *we call the homomorphic signature scheme* statistically context hiding, *if* $\epsilon(\lambda) = 0$, *we call it* perfectly context hiding.

3 Construction of CHQS

In this section, we present our novel homomorphic signature scheme CHQS. We first recall the hardness assumptions on which its security is based. Our construction is

then described in detail. We also give an intuition of how CHQS works by providing an example in a simple case.

We now recall computational hardness assumptions on which CHQS is based.

Definition 9. *Let $\mathcal{G}$ be a generator of cyclic groups of order p and let $\mathbb{G} \xleftarrow{\$} \mathcal{G}(1^\lambda)$. We say the Discrete Logarithm assumption (DL) holds in $\mathbb{G}$ if there exists no PPT adversary $\mathcal{A}$ that, given (g, g^a) for a random generator $g \in \mathbb{G}$ and random $a \in \mathbb{Z}_p$, can output a with more than negligible probability, i.e. if $Pr[a \leftarrow \mathcal{A}(g, g^a) | g \xleftarrow{\$} \mathbb{G},\ a \xleftarrow{\$} \mathbb{Z}_p] = \mathsf{negl}(\lambda)$.*

Definition 10 (Asymmetric bilinear groups). *An* asymmetric bilinear group *is a tuple $\mathsf{bgp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$, such that:*

- *$\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_T$ are cyclic groups of prime order p,*
- *$g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ are generators for their respective groups,*
- *the DL assumption holds in $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_T$,*
- *$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is bilinear, i.e. $e({g_1}^a, {g_2}^b) = e(g_1, g_2)^{ab}$ holds for all $a, b \in \mathbb{Z}$,*
- *e is non-degenerate, i.e. $e(g_1, g_2) \neq 1_{\mathbb{G}_T}$, and*
- *e is efficiently computable.*

We write $g_t = e(g_1, g_2)$.

Definition 11 ([8]). *Let $\mathcal{G}$ be a generator of asymmetric bilinear groups and let $\mathsf{bgp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \xleftarrow{\$} \mathcal{G}(1^\lambda)$. We say the Flexible Diffie–Hellman Inversion (FDHI) assumption holds in bgp if for every PPT adversary $\mathcal{A}$,*

$$Pr[W \in \mathbb{G}_1 \backslash \{1_{\mathbb{G}_1}\} \wedge W' = W^{\frac{1}{z}} : (W, W') \leftarrow \mathcal{A}(g_1, g_2, g_2^z, g_2^v, g_1^{\frac{z}{v}}, g_1^r, g_1^{\frac{r}{v}}) |$$
$$z, r, v \xleftarrow{\$} \mathbb{Z}_p] = \mathsf{negl}(\lambda).$$

We now present the algorithms making up CHQS. It is homomorphic with respect to arithmetic circuits $f : \mathbb{Z}_p^n \to \mathbb{Z}_p$ of degree 2, where $p \geq 5$ (see Lemma 1). CHQS is *graded*, i.e. there exist level-1 and level-2 signatures. Level-1 signatures are created by signing messages, whereas level-2 signatures occur during homomorphic evaluation over multiplication gates. Graded structures like this occur naturally in homomorphic schemes like the ones by Catalano and others [6, 10]. We use dedicated elements (which we will denote by T_τ) in our level-1 signatures to handle multiplication gates. Those elements no longer occur in the level-2 signatures.

$\mathsf{HKeyGen}(1^\lambda, n)$**:** On input a security parameter λ and an integer n, the algorithm runs $\mathcal{G}(1^\lambda)$ to obtain a bilinear group $\mathsf{bgp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T,\ g_1, g_2, e)$. It chooses $x, y \leftarrow \mathbb{Z}_p$ uniformly at random. It sets $h_t = g_t^x$. It then samples t_{τ_i}, k_{τ_i} uniformly at random for all $i \in [n]$ and sets $F_{\tau_i} = g_2^{t_{\tau_i}}$, as well as $f_{\tau_i} = g_t^{y t_{\tau_i}}$, $f_{\tau_i, \tau_j} = g_t^{t_{\tau_i} k_{\tau_j}}$, for all $i, j \in [n]$.

Additionally it makes use of a regular signature scheme $\mathsf{Sig}' = (\mathsf{KeyGen}', \mathsf{Sign}', \mathsf{Verify}*)$ and a a pseudorandom function $\mathsf{PRF} : \mathcal{K} \times \{0,1\}^* \to \mathbb{Z}_p$. For these it generates keys $(\mathsf{sk}', \mathsf{pk}') \leftarrow \mathsf{KeyGen}'(1^\lambda)$ and $K \xleftarrow{\$} \mathcal{K}$. It returns the key pair $(\mathsf{sk}, \mathsf{pk})$ with $\mathsf{sk} = (\mathsf{sk}', K, x, y, \{t_{\tau_i}\}_{i=1}^n)$ and $\mathsf{pk} = (\mathsf{pk}', \mathsf{bgp}, h_t, \{F_{\tau_i}, f_{\tau_i}\}_{i=1}^n, \{f_{\tau_i,\tau_j}\}_{i,j=1}^n)$.

$\mathsf{HSign}(\mathsf{sk}, \Delta, \tau, m)$**:** On input a secret key sk, a dataset identifier Δ, an input identifier $\tau \in \mathcal{T}$, and a message $m \in \mathbb{Z}_p$, the algorithm generates the parameters for the dataset identified by Δ, by running $z \leftarrow \mathsf{PRF}_K(\Delta)$ and computing $Z = g_2^{\frac{1}{z}}$. Z is bound to the dataset identifier Δ by using the regular signature scheme, i.e. it sets $\sigma_\Delta \leftarrow \mathsf{Sign}'(\mathsf{sk}', Z|\Delta)$.

It chooses $r, s \in \mathbb{Z}_p$ uniformly at random. Then it computes $\Lambda \leftarrow g_1^{z(xm+(y+s)t_\tau+r)}$, $R \leftarrow g_1^r$, $S_\tau \leftarrow g_1^s$, as well as $T_\tau \leftarrow g_1^{ym-k_\tau}$. It sets $\mathcal{T} = \{(\tau, S_\tau, T_\tau)\}$ and then returns the signature $\sigma = (m, \sigma_\Delta, Z, \Lambda, R, \mathcal{T})$. Following the convention of Backes et al. [3], our signature contains the message m.

$\mathsf{HEval}(\mathsf{pk}, \mathcal{P}_\Delta, \boldsymbol{\sigma})$**:** Inputs are a public key pk, a multi-labeled program $\mathcal{P}_\Delta$ containing an arithmetic circuit f of degree at most 2, and signatures $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_n)$, where $\sigma_i = (m_i, \sigma_{\Delta,i}, Z_i, \Lambda_i, R_i, \mathcal{T}_i)$. The algorithm checks if the signatures share the same public values, i.e. if $\sigma_{\Delta,1} = \sigma_{\Delta,i}$ and $Z_1 = Z_i$ for all $i = 2, \ldots, n$, and the signature for each set of public values is correct and matches the dataset identifier Δ, i.e. $\mathsf{Verify}'(\mathsf{pk}', Z_i|\Delta, \sigma_{\Delta,i}) = 1$ for any $i \in [n]$. If this is not the case, the algorithm rejects the signature. Otherwise, it proceeds as follows. We describe this algorithm in terms of six different procedures $(\mathsf{Add}_1, \mathsf{Mult}, \mathsf{Add}_2, \mathsf{cMult}_1, \mathsf{cMult}_2, \mathsf{Shift})$ allowing to evaluate the circuit gate by gate.

Add_1**:** On input two level-1 signatures $\sigma_i = (m_i, \sigma_\Delta, Z, \Lambda_i, R_i, \mathcal{T}_i)$ for $i = 1, 2$ it computes as follows: $m = m_1 + m_2$, $\Lambda = \Lambda_1 \cdot \Lambda_2$, $R = R_1 \cdot R_2$, and $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$. It outputs a level-1 signature $\sigma = (m, \sigma_\Delta, Z, \Lambda, R, \mathcal{T})$.

Mult**:** On input two level-1 signatures $\sigma_i = (m_i, \sigma_\Delta, Z, \Lambda_i, R_i, \mathcal{T}_i)$ for $i = 1, 2$ and the public key pk, it computes as follows: $m = m_1 m_2$, $\Lambda = \Lambda_1^{m_2}$, $R = R_1^{m_2}$, $S_\tau = S_{\tau_1}^{m_2} \cdot T_{\tau_2}$, for all $\tau \in \mathcal{T}_1$, and $\mathcal{L} = \{(\tau, S_\tau)\}$ for all $\tau \in \mathcal{T}_1$. It outputs a level-2 signature $\sigma = (m, \sigma_\Delta, Z, \Lambda, R, \mathcal{L})$.

Add_2**:** On input two level-2 signatures $\sigma_i = (m_i, \sigma_\Delta, Z, \Lambda_i, R_i, \mathcal{L}_i)$ for $i = 1, 2$, it computes as follows: $m = m_1 + m_2$, $\Lambda = \Lambda_1 \cdot \Lambda_2$, $R = R_1 \cdot R_2$, $S_\tau = S_{\tau,1} \cdot S_{\tau,2}$ for all $(\tau, \cdot) \in \mathcal{L}_1 \cap \mathcal{L}_2$, $S_\tau = S_{\tau,i}$ for all τ such that $(\tau, \cdot) \in \mathcal{L}_1 \Delta \mathcal{L}_2$, and $\mathcal{L} = \{(\tau, S_\tau)\}$ for all $(\tau, \cdot) \in \mathcal{L}_1 \cup \mathcal{L}_2$. It outputs a level-2 signature $\sigma = (m, \sigma_\Delta, Z, \Lambda, R, \mathcal{L})$.

cMult_1**:** On input a level-1 signature $\sigma' = (m', \sigma_\Delta, Z, \Lambda', R', \mathcal{T}')$ and a constant $c \in \mathbb{Z}_p$, it computes as follows: $m = cm'$, $\Lambda = \Lambda'^c$, $R = R'^c$, $S_\tau = S_\tau'^c$, $T_\tau = T_\tau'^c$ for all $\tau \in \mathcal{T}'$, and $\mathcal{T} = \{(\tau, S_\tau, T_\tau)\}_{\tau \in \mathcal{T}}$. It outputs a level-1 signature $\sigma = (m, \sigma_\Delta, Z, \Lambda, R, \mathcal{T})$.

cMult_2**:** On input a level-2 signature $\sigma = (m', \sigma_\Delta, Z, \Lambda', R', \mathcal{L}')$ and a constant $c \in \mathbb{Z}_p$, it computes as follows: $m = cm'$, $\Lambda = \Lambda'^c$, $R = R'^c$, $S_\tau = S_\tau'^c$ for all

$(\tau, S'_\tau) \in \mathcal{L}'$, and $\mathcal{L} = \{(\tau, S_\tau)\}$ for all $(\tau, S'_\tau) \in \mathcal{L}'$. It outputs a level-2 signature $\sigma = (m, \sigma_\Delta, Z, \Lambda, R, \mathcal{L})$.

Shift**:** On input a level-1 signature $\sigma' = (m', \sigma_\Delta, Z, \Lambda', R', \mathcal{T}')$, it computes as follows: $m = m'$, $\Lambda = \Lambda'$, $R = R'$, and $\mathcal{L} = \{(\tau, S_\tau)\}_{\tau \in \mathcal{T}'}$. It outputs a level-2 signature $\sigma = (m, \sigma_\Delta, Z, \Lambda, R, \mathcal{L})$. Shift simply describes how to derive a level-2 signature from a level-1 signature.

$\mathsf{HVerify}(\mathsf{pk}, \mathcal{P}_\Delta, M, \sigma)$**:** On input a public key pk, a message M, a (level-1 or -2) signature σ, a multi-labeled program $\mathcal{P}_\Delta$ containing an arithmetic circuit f of degree at most 2, the algorithm parses (without loss of generality) $\sigma = (m, \sigma_\Delta, Z, \Lambda, R, \mathcal{L})$.
It then checks whether the following three equations hold:

1. $M = m$
2. $\mathsf{Verify}'(\mathsf{pk}', Z|\Delta, \sigma_\Delta) = 1$
3. $$e(\Lambda, Z) = e(R, g_2) \cdot h_t^m \cdot \prod_{i=1}^{n} f_i^{c_i} \cdot \prod_{(\tau,\cdot,\cdot)\in\mathcal{L}} e(S_\tau, F_\tau)$$
   for level-1 signatures and
   $$e(\Lambda, Z) = e(R, g_2) \cdot h_t^m \cdot \prod_{i,j=1}^{n} f_{i,j}^{c_{i,j}} \cdot \prod_{j=1}^{n} f_j^{c_j} \cdot \prod_{(\tau,\cdot)\in\mathcal{L}} e(S_\tau, F_\tau)$$
   for level-2 signatures, respectively, where $c_{i,j}$ and c_j are the coefficients in $\mathcal{P}_\Delta$.

If all 3 equations hold respectively, it returns `1`. Otherwise, it returns `0`.

We now give a brief example of how the HSign and HEval algorithms work.

Example 1. Let us consider the three messages $m_1 = 5, m_2 = 11, m_3 = 23$ and the function $f(m_1, m_2, m_3) = m_1 m_2 + m_1 m_3$. We have

$\mathsf{HSign}(\mathsf{sk}, \Delta, \tau, 5)$**:** Compute $\Lambda_1 = g_1^{z(5x+yt_1+s_1t_1+r_1)}, R_1 = g_1^{r_1}, S_1 = g_1^{s_1}, T_1 = g_1^{5y-k_1}$.
$\mathsf{HSign}(\mathsf{sk}, \Delta, \tau, 11)$**:** $\Lambda_2 = g_1^{z(11x+yt_2+s_2t_2+r_2)}, R_2 = g_1^{r_2}, S_2 = g_1^{s_2}, T_2 = g_1^{11y-k_2}$.
$\mathsf{HSign}(\mathsf{sk}, \Delta, \tau, 23)$**:** $\Lambda_3 = g_1^{z(23x+yt_3+s_3t_3+r_3)}, R_3 = g_1^{r_3}, S_3 = g_1^{s_3}, T_3 = g_1^{23y-k_3}$.

We consider the multi-labeled program $\mathcal{P}_\Delta = (f, \tau_1, \tau_2, \tau_3, \Delta)$. We perform $\mathsf{HEval}(\mathsf{pk}, \mathcal{P}_\Delta, (\sigma_1, \sigma_2, \sigma_3))$ by twice performing Mult and then performing Add_2.

$\mathsf{Mult}(\sigma_1, \sigma_2)$**:** Set $m = 5 \cdot 11 = 55$, $\Lambda = \Lambda_1^{11} = g_1^{z(55x+11yt_1+11s_1t_1^{+}11r_1)}$, $R = R_1^{11} = g_1^{11r_1}$, $S_1 = g_1^{11s_1} \cdot g_1^{11y-k_2} = g_1^{11s_1+11y-k_2}$, and $\mathcal{L} = \{(\tau_1, S_1)\}$.
$\mathsf{Mult}(\sigma_1, \sigma_3)$**:** Set $m' = 5 \cdot 23 = 115$, $\Lambda' = \Lambda_1^{23} = g_1^{z(115x+23yt_1+23s_1t_1+23r_1)}$, $R' = R_1^{23} = g_1^{23r_1}$, $S'_1 = g_1^{23s_1} \cdot g_1^{23y-k_3} = g_1^{23s_1+23y-k_3}$, and $\mathcal{L}' = \{(\tau_1, S'_1)\}$.
$\mathsf{Add}_2(\sigma, \sigma')$ **:** Set

- $m^* = 55 + 115 = 170$
- $\Lambda^* = \Lambda \cdot \Lambda' = g_1^{z(55x+11r_1+115x+23(y+s_1)t_1+11(y+s_1)t_1+23r_1)}$
  $= g_1^{z(170x+34(y+s_1)t_1+34r_1)}$
- $R^* = R \cdot R' = g_1^{11r_1+23r_1} = g_1^{34r_1}$,
- $S_1^* = S_1 \cdot S_1' = g_1^{11s_1+11y-k_2+23s_1+23y-k_3} = g_1^{34s_1+34y-k_2-k_3}$
- $\mathcal{L}^* = \{(\tau_1, S_1^*)\}$.

We run $\mathsf{HVerify}(\mathsf{pk}, \mathcal{P}_\Delta, 170, \sigma^*)$. The first two checks obviously pass. The third one is

$$e\left(g_1^{z(170x+34yt_1+34s_1t_1+34r_1)}, g_2^{\frac{1}{z}}\right) = g_t^{170x+34yt_1+34s_1t_1+34r_1} =$$

$$g_t^{34r_1} \cdot h_t^{170} \cdot g_t^{t_1k_2+t_1k_3} \cdot g_t^{-(t_1k_2+t_1k_3)+34s_1t_1+34yt_1} = e\left(R^*, g_2\right) h_t^{170} \cdot f_{1,2} \cdot f_{1,3} \cdot e\left(S_1^*, F_1\right).$$

4 Correctness, Efficiency, and Context Hiding Property of CHQS

We now prove the essential properties of CHQS, in particular correctness, amortized efficiency, and context hiding.

Theorem 1. *CHQS is correct in the sense of Definition 2.*

Proof. We first show the correctness for freshly generated signatures. We then show the correctness of the six procedures $(\mathsf{Add}_1, \mathsf{Mult}, \mathsf{Add}_2, \mathsf{cMult}_1, \mathsf{cMult}_2, \mathsf{Shift})$.

Sign: Let $\sigma = (m, \sigma_\Delta, Z, \Lambda, R, \mathcal{T}) \leftarrow \mathsf{HSign}(\mathsf{sk}, \Delta, \tau, m)$. By construction, $\mathsf{Verify}'(\mathsf{pk}', Z|\Delta, \sigma_\Delta) = 1$. Also, $e\left(\Lambda, Z\right) = e\left(g_1^{z(xm+(y+s)t_\tau+r)}, g_2^{\frac{1}{z}}\right) = e\left(g_1^{xm+(y+s)t_\tau+r}, g_2\right) = g_t^{xm+(y+s)t_\tau+r} = h_t^m \cdot f_\tau \cdot e\left(R, g_2\right) \cdot e\left(S, F_\tau\right)$ and consequently σ is a correct signature.

Add_1: We have two valid signatures σ_1 and σ_2. Thus $Z_1 = Z_2$ and $\mathsf{Verify}'(\mathsf{pk}', Z_1|\Delta, \sigma_\Delta) = 1$. Furthermore, $e\left(\Lambda_i, Z_i\right) = e\left(R_i, g_2\right) \cdot h_t^{m_i} \cdot f_i \cdot e\left(S_i, F_i\right)$, by construction. After performing Add_1, $e\left(\Lambda, Z\right) = e\left(\Lambda_1 \cdot \Lambda_2, Z_1\right) = e\left(\Lambda_1, Z_1\right) \cdot e\left(\Lambda_2, Z_2\right)$

$$\begin{aligned}
&= e\left(R_1, g_2\right) \cdot h_t^{m_1} \cdot f_1 \cdot e\left(S_1, F_1\right) \cdot e\left(R_2, g_2\right) \cdot h_t^{m_2} \cdot f_2 \cdot e\left(S_2, F_2\right) \\
&= e\left(R_1 \cdot R_2, g_2\right) \cdot h_t^{m_1+m_2} \cdot f_1 \cdot f_2 \cdot e\left(S_1, F_1\right) \cdot e\left(S_2, F_2\right) \\
&= e\left(R, g_2\right) \cdot h_t^{m} \cdot f_1 \cdot f_2 \cdot e\left(S_1, F_1\right) \cdot e\left(S_2, F_2\right)
\end{aligned}$$

hence σ is a correct signature. We also have $T = T_1 \cdot T_2 = g_1^{y(m_1+m_2)-(k_1+k_2)}$.

Mult: We have two valid signatures σ_1 and σ_2. Thus $Z_1 = Z_2$ and $\mathsf{Verify}'(\mathsf{pk}', Z_1|\Delta, \sigma_\Delta) = 1$. Furthermore, $e\left(\Lambda_i, Z_i\right) = e\left(R_i, g_2\right) \cdot h_t^{m_i} \cdot f_i \cdot e\left(S_i, F_i\right)$, by construction.

After performing Mult, $e(\Lambda, Z) = e(\Lambda_1^{m_2}, Z) = e(R_1^{m_2}, g_2) \cdot h_t^{m_1 m_2} \cdot f_1^{m_2} \cdot e(S_1, F_1)$

$$
\begin{aligned}
&= e(R_1^{m_2}, g_2) \cdot h_t^{m_1 m_2} \cdot f_1^{m_2} \cdot e(S_1^{m_2}, F_1) \cdot g_t^{-ym_2t_1 - k_2t_1} \cdot g_t^{ym_2t_1 + k_2t_1} \\
&= e(R_1^{m_2}, g_2) \cdot h_t^{m_1 m_2} \cdot f_1^{m_2} \cdot e\left(g_1^{s_1 m_2}, g_2^{t_1}\right) \cdot g_t^{-ym_2t_1} \cdot g_t^{k_2t_1} \cdot e\left(g_1^{ym_2t_1 - k_2t_1}, g_2\right) \\
&= e(R_1^{m_2}, g_2) \cdot h_t^{m_1 m_2} \cdot f_1^{m_2} \cdot f_1^{-m_2} \cdot f_{1,2} \cdot e\left(g_1^{ym_2 - k_2 + s_1 m_2}, g_2^{t_1}\right) \\
&= e(R, g_2) \cdot h_t^{m_1 m_2} \cdot f_{1,2} \cdot e(S_1^{m_2} \cdot T_1, F_1) \text{ so } \sigma \text{ is a correct signature.}
\end{aligned}
$$

$\mathsf{Add_2}$**:** We have two valid signatures σ_1 and σ_2. Thus $Z_1 = Z_2$ and $\mathsf{Verify}'(\mathsf{pk}', Z_1 | \Delta, \sigma_\Delta) = 1$. Furthermore, $e(\Lambda_i, Z_i) = e(R_i, g_2) \cdot h_t^{m_i} \cdot f_i \cdot \prod_{(\tau,\cdot)\in\mathcal{L}_1} e(S_{\tau,i}, F_\tau)$, by construction. After performing $\mathsf{Add_2}$, $e(\Lambda, Z) = e(\Lambda_1 \cdot \Lambda_2, Z_1)$

$$
\begin{aligned}
&= (R_1, g_2) \cdot h_t^{m_1} \cdot f_1 \prod_{(\tau,\cdot)\in\mathcal{L}_1} e(S_{\tau,1}, F_\tau) \cdot e(R_2, g_2) \cdot h_t^{m_2} \cdot f_2 \prod_{(\tau,\cdot)\in\mathcal{L}_2} e(S_{\tau,2}, F_\tau) \\
&= e(R_1 \cdot R_2, g_2) \cdot h_t^{m_1+m_2} \cdot f_1 \cdot f_2 \prod_{(\tau,\cdot)\in\mathcal{L}} e(S_\tau, F_\tau) \\
&= e(R, g_2) \cdot h_t^{m} \cdot f_1 \cdot f_2 \prod_{(\tau,\cdot)\in\mathcal{L}} e(S_\tau, F_\tau) \text{ thus } \sigma \text{ is a correct signature.}
\end{aligned}
$$

The correctness of $\mathsf{cMult_1}$ and $\mathsf{cMult_2}$ follows immediately from the correctness of $\mathsf{Add_1}$ and $\mathsf{Add_2}$ respectively. The correctness of Shift is trivial.

Theorem 2. *CHQS provides verification in time $\mathcal{O}(n)$ in an amortized sense.*

Proof. We describe the two algorithms $(\mathsf{HVerPrep}, \mathsf{HEffVer})$.

$\mathsf{HVerPrep}(\mathsf{pk}, \mathcal{P})$: This algorithm parses $\mathcal{P} = (f, \tau_1, \ldots \tau_n)$ with $f(m_1, \ldots m_n) = \sum_{i=1}^n c_i m_i + \sum_{i,j=1}^n c_{i,j} m_i m_j$ and takes the $f_i, f_{i,j}$ for $i, j \in [n]$ contained in the public key. It computes $F_\mathcal{P} \leftarrow \prod_{i,j=1}^n f_{i,j}^{c_{i,j}} \cdot \prod_{i=1}^n f_i^{c_i}$ and outputs $\mathsf{pk}_\mathcal{P} = (\mathsf{pk}', \mathsf{bgp}, h_t, \{F_i\}_{i=1}^n, F_\mathcal{P})$ where $\mathsf{pk}', \mathsf{bgp}, h_t, \{F_i\}_{i=1}^n$ are taken from pk.

$\mathsf{HEffVer}(\mathsf{pk}_\mathcal{P}, m, \sigma, \Delta)$**:** This algorithm is analogous to $\mathsf{HVerify}$, except that the value $\prod_{i,j=1}^n f_{i,j}^{c_{i,j}} \cdot \prod_{i=1}^n f_i^{c_i}$ has been precomputed as $F_\mathcal{P}$.

Obviously this satisfies correctness. During $\mathsf{HEffVer}$ the verifier now computes

$$e(\Lambda, Z) = e(R, g_2) \cdot h_t^m \cdot F_\mathcal{P} \cdot \prod_{i=1}^n e(S_i, F_i)$$

The running time of $\mathsf{HEffVer}$ is thus $\mathcal{O}(n)$.

Thus, CHQS achieves amortized efficiency in the sense of Definition 7 for every arithmetic circuit f of multiplicative depth 2, that has *superlinear* runtime complexity. We continue Example 1 to showcase these two algorithms.

Example 2. We again consider the labeled program $\mathcal{P} = (f, \tau_1, \tau_2, \tau_3)$, as well as the multi-labeled program $\mathcal{P}_\Delta = (f, \tau_1, \tau_2, \tau_3, \Delta)$, with the function $f(m_1, m_2, m_3) = m_1 m_2 + m_1 m_3$, and the messages $m_1 = 5, m_2 = 11, m_3 = 23$. We have

$\mathsf{HVerPrep}(\mathsf{pk}, \mathcal{P})$: We compute $F_\mathcal{P} = f_{1,2} \cdot f_{1,3} = g_t^{t_{\tau_1} k_2 + t_{\tau_1} k_3}$.

$\mathsf{HEffVer}(\mathsf{pk}_\mathcal{P}, m, \sigma, \Delta)$**:** As in Example 1, $m^* = 170$, $\Lambda^* = g_1^{z(170x+34yt_1+34s_1t_1+34r_1)}$, $R^* = g_1^{34r_1}$, $S_1^* = g_1^{34y-k_2-k_3}$ and $\sigma^* = (m^*, \sigma_\Delta, Z, \Lambda^*, R^*, \{(\tau_1, S_1^*)\})$.

$\mathsf{HVerify}(\mathsf{pk}, \mathcal{P}_\Delta, 170, \sigma^*)$**:** Again, the first two checks obviously pass. The third one is like in Example 1 $e\left(g_1^{z(170x+34yt_1+34s_1t_1+34r_1)}, g_2^{\frac{1}{z}}\right) = e\left(R^*, g_2\right) h_t^{170} \cdot f_{1,2} \cdot f_{1,3} \cdot e\left(S_1^*, F_1\right) = e\left(R^*, g_2\right) h_t^{170} \cdot F_\Delta \cdot e\left(S_1^*, F_1\right)$.

Bandwidth: CHQS is *not succinct*. However, the output of HSign is of constant size and thus independent of n. Hence no extensive bandwidth is needed during the upload of the data.

Theorem 3. *CHQS is perfectly context hiding according to Definition 8 if* Sig' *is a deterministic signature scheme.*

Proof. We show that our scheme is perfectly context hiding in the sense of Definition 8, by comparing the distributions of homomorphically derived signatures to that of simulated signatures. In our case, HHide is just the identity function, i.e. $\sigma \leftarrow \mathsf{HHide}(\mathsf{pk}, m, \sigma)$ for all pk, m, σ and $\mathsf{HHideVer} = \mathsf{HVerify}$. We show how to construct a simulator Sim that outputs signatures perfectly indistinguishable from the ones obtained by running HEval. Parse the simulator's input as $\mathsf{sk} = (\mathsf{sk}', K)$, $\mathcal{P}_\Delta = (f, \tau_1, \ldots, \tau_n, \Delta)$. For each τ appearing in $\mathcal{P}_\Delta$, it chooses $s_\tau \in \mathbb{Z}_p$ uniformly at random as well as $r \in \mathbb{Z}_p$ uniformly at random. With this information, the simulator computes $m' = m$, $Z' = g_2^z$ where $z \leftarrow \mathsf{PRF}_K(\Delta)$, $\sigma'_\Delta \xleftarrow{\$} \mathsf{Sign}'(\mathsf{sk}', Z|\Delta)$, $\Lambda' = g_1^{z(xm' + y(\sum_{i,j=1}^n c_{ij} t_i k_j + \sum_{i=1}^n c_i t_i) + \sum_{i=1}^n s_{\tau_i} t_i + r)}$, $R' = g_1^r$, $S'_\tau = g_1^{s_\tau}$ for all τ appearing in $\mathcal{P}_\Delta$, and $\mathcal{T}' = \{(\tau, S_\tau)\}_{\tau \in \mathcal{P}_\Delta}$. The simulator outputs the signature $\sigma' = (m', \sigma'_\Delta, Z', \Lambda', R', \mathcal{T}')$.

We now show that this simulator allows for perfectly context hiding security. We fix an arbitrary key pair $(\mathsf{sk}, \mathsf{pk})$, a multi-labeled program $(f, \tau_1, \ldots, \tau_n, \Delta)$, and messages $m_1, \ldots, m_n \in \mathbb{Z}_p$. Let $\sigma \leftarrow \mathsf{HEval}(\mathsf{pk}, \mathcal{P}_\Delta, \boldsymbol{\sigma})$ and parse it as $\sigma = (\sigma_\Delta, Z, \Lambda)$. We inspect each component of the signature. $Z = \mathsf{PRF}_K(\Delta)$ by definition and thus also $Z = Z'$. In particular, $z = z'$ where $Z = g_2^z$ and $Z' = g_2^{z'}$. We have $\sigma_\Delta = \mathsf{Sign}'(\mathsf{sk}', Z|\Delta)$ by definition, and since $Z = Z'$, also $\sigma_\Delta = \sigma'_\Delta$ since Sign' is deterministic. We consider Λ as an exponentiation of g_1^z. Since $\Lambda = \prod_{i,j=1}^n \Lambda_i^{c_{ij} m_j}$ by construction, for the exponent

we have:

$$
\begin{aligned}
& xm + \sum_{i,j=1}^{n} c_{ij} m_j (y t_i + s_i t_i + r_i) \\
&= xm' + y(\sum_{i,j=1}^{n} c_{ij} t_i k_j + \sum_{i=1}^{n} c_i t_i) - y(\sum_{i,j=1}^{n} c_{ij} t_i k_j + \sum_{i=1}^{n} c_i t_i) \\
&\quad + \sum_{i,j=1}^{n} c_{ij} m_j (y t_i + s_i t_i + r_i) \\
&= xm' + y(\sum_{i,j=1}^{n} c_{ij} t_i k_j + \sum_{i=1}^{n} c_i t_i) + \sum_{i=1}^{n} (\sum_{j=1}^{n} -y c_{ij} k_j - y c_i + c_{ij} m_j y + c_{ij} s_i m_j) t_i \\
&\quad + \sum_{i,j=1}^{n} c_{ij} m_j r_i \\
&= xm' + y(\sum_{i,j=1}^{n} c_{ij} t_i k_j + \sum_{i=1}^{n} c_i t_i) + \sum_{i=1}^{n} \tilde{s}_i t_i + \tilde{r}
\end{aligned}
$$

Thus the exponent corresponds to a different choice of $r, s_i \in \mathbb{Z}_p$. Analogously, $S_\tau = g_1^{\tilde{s}_{\tau_i}}$ and $R = g_1^{\tilde{r}}$, where $\tilde{r}, \tilde{s}_{\tau_i}$ are distributed uniformly at random as linear combinations of uniformly random field elements.

All elements are either identical, or have the exact same distribution. Thus even a computationally unbounded distinguisher has no advantage distinguishing the two cases.

5 Unforgeability of CHQS

This section deals with the security reduction of CHQS to the FDHI assumption (see Definition 11). We first present the hybrid games used in the proof, and then argue their indistinguishability for a PPT adversary $\mathcal{A}$ in the form of several lemmata.

Theorem 4. *If* Sig' *is an unforgeable signature scheme,* PRF *is a pseudorandom function, and the FDHI assumption (see Definition 11) holds in* bgp*, then CHQS is an unforgeable homomorphic signature scheme in the sense of Definition 5.*

Proof. To prove Theorem 4, we define a series of games with the adversary $\mathcal{A}$ and we show that the adversary $\mathcal{A}$ wins, i.e. the game outputs $\mathtt{1}$ only with negligible probability. Following the notation of [8], we write $G_i(\mathcal{A})$ to denote that a run of game i with adversary $\mathcal{A}$ returns $\mathtt{1}$. We use flag values bad_i, initially set to false. If at the end of the game any of these flags is set to true, the game simply outputs $\mathtt{0}$. Let $\mathsf{Bad_i}$ denote the event that $\mathsf{bad_i}$ is set to true during game i. As shown in [10, Proposition 2], any

adversary who outputs a Type 3 forgery (see Definition 4) can be converted into one that outputs a Type 2 forgery. Hence we only have to deal with Type 1 and Type 2 forgeries.

Game 1 is the security experiment $\mathsf{HomUF-CMA}_{\mathcal{A},\mathsf{HSign}}$ between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, where $\mathcal{A}$ only outputs Type 1 or Type 2 forgeries.

Game 2 is defined as Game 1, except for the following change. Whenever $\mathcal{A}$ returns a forgery $({\mathcal{P}^*}_{\Delta^*}, \sigma*)$ with $\sigma* = (m^*, \mathcal{T}^*, \sigma^*_{\Delta^*}, Z^*, \Lambda^*, R^*, S^*)$ or $\sigma* = (m^*, \sigma^*_{\Delta^*}, Z^*, \Lambda^*, R^*, \mathcal{L}^*)$ and Z^* has not been generated by the challenger during the queries, then Game 2 sets $\mathsf{bad}_2 = \mathsf{true}$. After this change, the game never outputs 1 if $\mathcal{A}$ returns a Type 1 forgery.

Game 3 is defined as Game 2, except that the pseudorandom function F is replaced by a random function $\mathcal{R} : \{0,1\}^* \to \mathbb{Z}_p$.

Game 4 is defined as Game 3, except for the following change. At the beginning $\mathcal{C}$ chooses $\mu \in [Q]$ uniformly at random, where $Q = \mathsf{poly}(\lambda)$ is the number of queries made by $\mathcal{A}$ during the game. Let $\Delta_1, \ldots, \Delta_Q$ be all the datasets queried by $\mathcal{A}$. Then if in the forgery $\Delta^* \neq \Delta_\mu$ set $\mathsf{bad}_4 = \mathsf{true}$.

Game 5 is defined as Game 4, except for the following change. At the very beginning $\mathcal{C}$ chooses $z_\mu \in \mathbb{Z}_p$ at random and computes $Z_\mu = g_2^{z_\mu}$. It will use Z_μ whenever queried for dataset Δ_μ. It chooses $a_i, b_i \in \mathbb{Z}_p$ uniformly at random for $i \in [n]$ and sets $f_{\tau_i} = g_t^{y(a_i + z_\mu b_i)}$, $F_{\tau_i} = g_2^{a_i + z_\mu b_i}$ as well as $f_{\tau_i,\tau_j} = g_t^{k_j y(a_i + z_\mu b_i)}$.

Game 6 is defined as Game 5, except for the following change. The challenger runs an additional check. If $\mathsf{HVerify}(\mathsf{pk}, \mathcal{P}^*_\Delta, m^*, \sigma^*) = 1$, the challenger computes $\hat{\sigma} \leftarrow \mathsf{HEval}(\mathsf{pk}, \mathcal{P}^*_\Delta, \boldsymbol{\sigma})$ over the signatures σ_i generated in dataset Δ^*. We have $\hat{\sigma} = (\hat{m}, \hat{\mathcal{T}}, \sigma_\Delta, Z, \hat{\Lambda}, \hat{R}, \hat{S})$ in case of a level-1 signature and $\hat{\sigma} = \sigma = (\hat{m}, \sigma_\Delta, Z, \hat{\Lambda}, \hat{R}, \hat{\mathcal{L}})$ in case of a level-2 signature. If $\Lambda^* \cdot \prod_{i=1}^n \hat{S}_i^{b_i} = \hat{\Lambda} \cdot \prod_{i=1}^n S_i^{*b_i}$ then $\mathcal{C}$ sets $\mathsf{bad}_6 = \mathsf{true}$.
Any noticeable difference between Games 1 and 2 can be reduced to producing a forgery for the signature scheme. If Bad_2 occurs, then $\mathcal{A}$ produced a valid signature $\sigma^*_{\Delta^*}$ for $(\Delta^*|Z^*)$ despite never having queried a signature on any $(\Delta^*|\cdot)$. This is obviously a forgery on the signature scheme.
Under the assumption that F is pseudorandom, Games 2 and 3 are computationally indistinguishable.
We have, by definition, $Pr[G_3(\mathcal{A})] = Q \cdot Pr[G_4(\mathcal{A})]$. It is obvious that $Pr[G_4(\mathcal{A})] = Pr[G_5(\mathcal{A})]$, since the public keys are perfectly indistinguishable. It is easy to see that $|Pr[G_5(\mathcal{A})] - \Pr[G_6(\mathcal{A})]| \leq Pr[\mathsf{Bad}_6]$. This occurs only with negligible probability if the FDHI assumption holds. For a proof of this statement, see Lemma 3 in Appendix A.
After these modifications, Game 6 can only output 1 if $\mathcal{A}$ produces a forgery $({\mathcal{P}^*}_{\Delta^*}, m^*, \sigma^*)$ such that $\mathsf{HVerify}(\mathsf{pk}, \mathcal{P}^*_\Delta, m^*, \sigma^*) = 1$ and $m^* \neq \hat{m}$, $\Lambda^* \neq \hat{\Lambda}$. This only occurs with negligible probability if the FDHI assumption holds. For a corresponding proof, we refer to Lemma 4 in Appendix A.

6 Related Work

Linearly homomorphic signature schemes were introduced by Desmedt [14] and later refined by Johnson et al [20]. Freeman proposed stronger security definitions [16]. A first instantiation, based on the 2-out-of-3 Computational Diffie–Hellmann assumption, was proposed by Boneh et al [5]. It was followed by multiple schemes [1, 2, 4, 8, 9, 18, 23], based on various hardness assumptions. None of these schemes support quadratic functions, unlike CHQS.

Some constructions for homomorphic authenticators go beyond the linear case. Backes et al. presented a homomorphic MAC for arithmetic circuits of degree 2 constructed from bilinear maps [3]. However, this approach is not context hiding and only offers private verifiability, while we offer verifiability for arbitrary third parties and perfect context hiding. Catalano et al. showed how to construct homomorphic signatures for arithmetic circuits of fixed depth from graded encoding schemes, a special type of multilinear maps [10]. Existing graded encoding schemes [12, 17] have, however, suffered strong cryptanalytic attacks in recent years [11, 21, 22]. In contrast, CHQS can be instantiated with elliptic curve-based bilinear groups, which have been reliable building block in cryptography for years.

Some lattice-based homomorphic signatures schemes [15,19] support boolean circuits of fixed degree. However, these schemes suffer the performance drawback of signing every single input bit, while our solution can sign entire finite field elements. Additionally [15] is also not shown to be context hiding.

More generally verifiable computing can be used to achieve verifiability of delegated computations. Many different schemes have been proposed. For a detailed comparison we refer to [13]. A general feature of homomorphic-signature-based schemes is that they allow for incremental updates of data sets, i.e. additional data can be added after the first delegation of data. Other verifiable computing schemes require all data to be used during the computation to be known before outsourcing.

7 Conclusion

Our new homomorphic signature scheme CHQS can be instantiated from ordinary bilinear groups, but still allows public verifiability for polynomials of degree greater than 1. Previous proposals either were limited to private verifiability, or relied on advanced primitives like graded encoding schemes. Such alternatives have recently been threatened by substantial cryptanalytic progress. Bilinear groups, however, are well understood and have been a reliable cryptographic building block for years.

We have demonstrated a novel approach using pairings to obtain both public verifiability and the ability to homomorphically evaluate a multiplication at the same time. CHQS achieves several desirable properties, including context hiding and amortized

efficiency. Furthermore, we reduced its security to the FDHI assumption in the standard model. This enables homomorphic signature schemes as a means of achieving verifiability for delegated computations over authenticated data, for example in the case of second-order statistics over health data in the cloud.

Future Work While CHQS is both context hiding and achieves efficient verification, the construction of a scheme also achieving succinctness and constant time verification is an open problem. Another question remains: can primitives supporting degrees higher than two still be constructed from bilinear maps?

Acknowledgments

References

1. Attrapadung, N., Libert, B., Peters, T.: Computing on Authenticated Data: New Privacy Definitions and Constructions. In: ASIACRYPT 2012. LNCS, vol. 7658, pp. 367–385. Springer (2012)
2. Attrapadung, N., Libert, B., Peters, T.: Efficient Completely Context-Hiding Quotable and Linearly Homomorphic Signatures. In: PKC 2013. LNCS, vol. 7778, pp. 386–404. Springer (2013)
3. Backes, M., Fiore, D., Reischuk, R.M.: Verifiable delegation of computation on outsourced data. In: ACM CCS 2013. pp. 863–874. ACM (2013)
4. Boneh, D., Freeman, D.M.: Linearly Homomorphic Signatures over Binary Fields and New Tools for Lattice-Based Signatures. In: PKC 2011. LNCS, vol. 6571, pp. 1–16. Springer (2011)
5. Boneh, D., Freeman, D.M., Katz, J., Waters, B.: Signing a Linear Subspace: Signature Schemes for Network Coding. In: PKC 2009. LNCS, vol. 5443, pp. 68–87. Springer (2009)
6. Catalano, D., Fiore, D.: Using Linearly-Homomorphic Encryption to Evaluate Degree-2 Functions on Encrypted Data. In: ACM CCS 2015. pp. 1518–1529. ACM (2015)
7. Catalano, D., Fiore, D., Gennaro, R., Nizzardo, L.: Generalizing Homomorphic MACs for Arithmetic Circuits. In: PKC 2014. LNCS, vol. 8383, pp. 538–555. Springer (2014)
8. Catalano, D., Fiore, D., Nizzardo, L.: Programmable Hash Functions Go Private: Constructions and Applications to (Homomorphic) Signatures with Shorter Public Keys. In: CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 254–274. Springer (2015)
9. Catalano, D., Fiore, D., Warinschi, B.: Efficient Network Coding Signatures in the Standard Model. In: PKC 2012. LNCS, vol. 7293, pp. 680–696. Springer (2012)
10. Catalano, D., Fiore, D., Warinschi, B.: Homomorphic Signatures with Efficient Verification for Polynomial Functions. In: CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 371–389. Springer (2014)
11. Coron, J., Lee, M.S., Lepoint, T., Tibouchi, M.: Cryptanalysis of GGH15 Multilinear Maps. In: CRYPTO 2016, Part II. LNCS, vol. 9815, pp. 607–628. Springer (2016)
12. Coron, J., Lepoint, T., Tibouchi, M.: Practical Multilinear Maps over the Integers. In: CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 476–493. Springer (2013)
13. Demirel, D., Schabhüser, L., Buchmann, J.A.: Privately and Publicly Verifiable Computing Techniques — A Survey. Springer Briefs in Computer Science, Springer (2017)
14. Desmedt, Y.: Computer security by redefining what a computer is. In: NSPW. pp. 160–166. ACM (1993)
15. Fiore, D., Mitrokotsa, A., Nizzardo, L., Pagnin, E.: Multi-key Homomorphic Authenticators. In: ASIACRYPT 2016, Part II. LNCS, vol. 10032, pp. 499–530 (2016)
16. Freeman, D.M.: Improved Security for Linearly Homomorphic Signatures: A Generic Framework. In: PKC 2012. LNCS, vol. 7293, pp. 697–714. Springer (2012)

17. Garg, S., Gentry, C., Halevi, S.: Candidate Multilinear Maps from Ideal Lattices. In: EUROCRYPT 2013. LNCS, vol. 7881, pp. 1–17. Springer (2013)
18. Gennaro, R., Katz, J., Krawczyk, H., Rabin, T.: Secure Network Coding over the Integers. In: PKC 2010. LNCS, vol. 6056, pp. 142–160. Springer (2010)
19. Gorbunov, S., Vaikuntanathan, V., Wichs, D.: Leveled Fully Homomorphic Signatures from Standard Lattices. In: STOC 2015. pp. 469–477. ACM (2015)
20. Johnson, R., Molnar, D., Song, D.X., Wagner, D.A.: Homomorphic Signature Schemes. In: CT-RSA 2002. LNCS, vol. 2271, pp. 244–262. Springer (2002)
21. Lee, H.T., Seo, J.H.: Security Analysis of Multilinear Maps over the Integers. In: CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 224–240. Springer (2014)
22. Miles, E., Sahai, A., Zhandry, M.: Annihilation Attacks for Multilinear Maps: Cryptanalysis of Indistinguishability Obfuscation over GGH13. In: CRYPTO 2016, Part II. LNCS, vol. 9815, pp. 629–658. Springer (2016)
23. Schabhüser, L., Buchmann, J.A., Struck, P.: A Linearly Homomorphic Signature Scheme from Weaker Assumptions. In: IMACC 2017. LNCS, vol. 10655, pp. 261–279. Springer (2017)

A Lemmata for Theorem 4

Lemma 3. *If there exists a PPT adversary $\mathcal{A}$ for whom Bad_6 occurs with non-negligible probability during Game 6 as described in Theorem 4, there exists a PPT simulator $\mathcal{S}$ who can solve the FDHI problem (see Definition 11) with non-negligible probability.*

Proof. Assume we have a PPT adversary $\mathcal{A}$ that can produce the result Bad_6 during Game 6. We show how a simulator $\mathcal{S}$ can use this to break the FDHI assumption.

Given $(g_1, g_2, g_2^z, g_2^v, g_1^{\frac{z}{v}}, g_1^r, g_1^{\frac{r}{v}})$ simulator $\mathcal{S}$ simulates Game 6.

Setup Simulator $\mathcal{S}$ chooses an index $\mu \in [Q]$ uniformly at random. During the key generation it chooses $a_i, b_i, k_i, y \in \mathbb{Z}_p$ uniformly at random for all $i \in [n]$. It sets $f_{\tau_i} = g_t^{ya_i} \cdot e\,(g_1, g_2^z)^{yb_i}$, $F_{\tau_i} = g_2^{a_i} \cdot (g_2^z)^{b_i}$, for all $i \in [n]$, as well as $f_{\tau_i,\tau_j} = g_t^{k_j y a_i} \cdot e\,(g_1, g_2^z)^{k_j y b_i}$, for all $i, j \in [n]$. It sets $h_t = e(g_1, g_2^z)$.
Additionally, it generates a key pair $(\mathsf{sk}', \mathsf{pk}') \leftarrow \mathsf{KeyGen}'(1^\lambda)$ of a regular signature scheme. It gives $\mathcal{A}$ the public key $\mathsf{pk} = (\mathsf{pk}', \mathsf{bgp}, h_t, \{F_{\tau_i}, f_{\tau_i}\}_{i=1}^n, \{f_{\tau_i,\tau_j}\}_{i,j=1}^n)$.

Queries Let l be a counter for the number of datasets queried by $\mathcal{A}$ (initially, it sets $l = 1$). For every new queried dataset Δ, simulator $\mathcal{S}$ creates a list $\mathcal{Q}_\Delta$ of tuples (τ, m), which collects all the label/message pairs queried by the adversary on Δ.
Moreover, whenever the l-th new dataset Δ_l is queried, $\mathcal{S}$ does the following: If $l = \mu$, it samples a random $\zeta_\mu \in \mathbb{Z}_p$, sets $Z_\mu = (g_2^z)^{\frac{1}{\zeta_\mu}}$ and stores ζ_μ.
If $l \neq \mu$, it samples a random $\zeta_l \in \mathbb{Z}_p$ and sets $Z_l = (g_2^v)^{\frac{1}{\zeta_l}}$ and stores ζ_l. Since all Z_l are randomly distributed in $\mathbb{G}_2$, they have the same distribution as in Game 6. Given a query (Δ, τ, m) with $\Delta = \Delta_m$, simulator $\mathcal{S}$ first computes $\sigma_{\Delta_l} \leftarrow \mathsf{Sign}'(\mathsf{sk}', Z_l|\Delta_l)$.
If $l \neq \mu$, it samples $s_\tau, \rho_\tau \in \mathbb{Z}_p$ uniformly at random and computes

$\Lambda_\tau = \left((g_1^{\frac{z}{v}})^{(y+s_\tau)b_\tau} \cdot (g_1^{\frac{r}{v}})^{\rho_\tau} \cdot (g_1^{\frac{z}{v}})^m\right)^{\zeta_l}$, $R_\tau = g_1^{-(y+s_\tau)a_\tau} \cdot (g_1^r)^{\rho_\tau}$, $S_\tau = g_1^{s_\tau}$, $T_\tau = g_1^{my-k_\tau}$, $\mathcal{T} = \{(\tau, S_\tau, T_\tau)\}$ and gives $\sigma = (m, \sigma_{\Delta_l}, Z_l, \Lambda_\tau, R_\tau, \mathcal{T})$ to $\mathcal{A}$. We have

$$
\begin{aligned}
&e(\Lambda_\tau, Z_l) = e\left((g_1^{\frac{z}{v}})^{(y+s_\tau)b_\tau} \cdot (g_1^{\frac{r}{v}})^{\rho_\tau} \cdot (g_1^{\frac{z}{v}})^m, g_2^{\frac{v}{\zeta_l}}\right)^{\zeta_l} \\
&= e\left((g_1^{\frac{z}{v}})^{(y+s_\tau)b_\tau} \cdot (g_1^{\frac{r}{v}})^{\rho_\tau} \cdot (g_1^{\frac{z}{v}})^m, g_2^{v}\right) = g_t^{z(y+s_\tau)b_\tau + r\rho_\tau + zm} \\
&= g_t^{z(y+s_\tau)b_\tau + r\rho_\tau + zm + a_\tau(y+s_\tau) - a_\tau(y+s_\tau)} = g_t^{zm} \cdot g_t^{y(a_\tau + zb_\tau)} \cdot g_t^{-ya_\tau + r\rho_\tau} \cdot g_t^{s_\tau(a_\tau + b_\tau z)} \\
&= h_t^m \cdot f_\tau \cdot e\left(R_\tau, g_2\right) \cdot e\left(S_\tau, F_\tau\right)
\end{aligned}
$$

hence this output is indistinguishable from the challenger's output during Game 6.
If $l = \mu$, simulator $\mathcal{S}$ computes $\Lambda_\tau = \left(g_1^{(y+s_\tau)b_\tau + m}\right)^{\zeta_\mu}$, $R_\tau = g_1^{-(y+s_\tau)a_\tau}$, $S_\tau = g_1^{s_\tau}$, $T_\tau = g_1^{my-k_\tau}$, $\mathcal{T} = \{(\tau, S_\tau, T_\tau)\}$ and gives $\sigma = (m, \sigma_{\Delta_\mu}, Z_\mu, \Lambda_\tau, R_\tau, \mathcal{T})$ to $\mathcal{A}$.
We have

$$
\begin{aligned}
&e(\Lambda_\tau, Z_\mu) = e\left(g_1^{(y+s_\tau)b_\tau + m}, g_2^{\frac{z}{\zeta_\mu}}\right)^{\zeta_\mu} = e\left(g_1^{(y+s_\tau)b_\tau + m}, g_2^{z}\right) = g_t^{z(y+s_\tau)b_\tau + zm} \\
&= g_t^{z(y+s_\tau)b_\tau + zm + (y+s_\tau)a_\tau - (y+s_\tau)a_\tau} = g_t^{zm} \cdot g_t^{y(a_\tau + zb_\tau)} \cdot g_t^{-ya_\tau} \cdot g_t^{s_\tau(a_\tau + b_\tau z)} \\
&= h_t^m \cdot f_\tau \cdot e\left(R_\tau, g_2\right) \cdot e\left(S_\tau, F_\tau\right)
\end{aligned}
$$

hence this output is indistinguishable from the challenger's output during Game 6.

Forgery Let $(\mathcal{P}^*_{\Delta^*}, \sigma^*)$ be a forgery with $\sigma^* = (m^*, \sigma^*_{\Delta^*}, Z^*, \Lambda^*, R^*, \mathcal{L}^*)$ and $\mathcal{L}^* = \{(\tau, S^*_\tau)\}_{\tau \in \mathcal{I}}$, where $\mathcal{I}$ is a subset of the label space, be the forgery returned by $\mathcal{A}$. The case of σ^* as a level-1 signature is just a simplification of the level-2 case and is omitted.
$\mathcal{S}$ computes $\hat{\sigma} \leftarrow \mathsf{HEval}(\mathsf{pk}, \mathcal{P}^*_{\Delta}, \boldsymbol{\sigma})$ over the signatures σ_i generated in dataset Δ^*.
$\mathcal{S}$ parses $\hat{\sigma} = (\hat{m}, \sigma^*_{\Delta^*}, Z^*, \hat{\Lambda}, \hat{R}, \hat{\mathcal{L}})$ with $\hat{\mathcal{L}} = \{(\tau, \hat{S}_\tau)\}_{\tau \in \hat{\mathcal{I}}}$ where $\hat{\mathcal{I}}$ is a subset of the label space. Without loss of generality, we assume $\mathcal{I} = \hat{\mathcal{I}} = \{\tau_i\}_{i \in [n]}$, i.e. $\mathcal{I}$ is the whole label space. We can always append $\mathcal{L}$ with $(\tau_i, 1_{\mathbb{G}_1})$ and write $S_{\tau_i} = S_i$.
If Game 6 outputs 1, we have $Z^* = Z_\mu$, $\Lambda^* \cdot \prod_{i=1}^n \hat{S}_i^{b_i} = \hat{\Lambda} \cdot \prod_{i=1}^n S_i^{*b_i}$, and the following hold:

$$
\begin{aligned}
e\left(\Lambda^*, Z_\mu\right) &= e\left(R^*, g_2\right) \cdot h_t^{m*} \cdot \prod_{i,j=1}^n f_{i,j}^{c_{i,j}} \cdot \prod_{j=1}^n f_j^{c_j} \cdot \prod_{i=1}^n e\left(S^*_{\tau_i}, F_{\tau_i}\right) \\
e\left(\hat{\Lambda}, Z_\mu\right) &= e\left(\hat{R}, g_2\right) \cdot h_t^{\hat{m}} \cdot \prod_{i,j=1}^n f_{i,j}^{c_{i,j}} \cdot \prod_{j=1}^n f_j^{c_j} \cdot \prod_{i=1}^n e\left(\hat{S}_{\tau_i}, F_{\tau_i}\right)
\end{aligned}
$$

Dividing those equations yields

$$\left(g_1^{(\hat{m}-m^*)z}\right)^{\frac{\zeta_\mu}{z}} = \left(\frac{R^*}{\hat{R}} \cdot \prod_{i=1}^{n} \frac{S_i^{*\,a_i}}{\hat{S}_i}\right)^{\frac{\zeta_\mu}{z}}$$

Thus, $\mathcal{S}$ can compute $W = g_1^{\hat{m}-m^*}$, $W' = \frac{R^*}{\hat{R}} \cdot \prod_{i=1}^{n} \frac{S_i^{*\,a_i}}{\hat{S}_i}$, and return W, W' as a solution to the FDHI problem. Since we have $m^* \neq \hat{m}$, we have $(W, W') \neq (1, 1)$. Our simulation has the same distribution as a real execution of Game 6.

Lemma 4. *If there exists a PPT adversary $\mathcal{A}$ who wins Game 6 with non-negligible probability, then there exists a PPT simulator $\mathcal{S}$ who can solve the FDHI problem (see Definition 11) with non-negligible probability.*

Proof. Assume a PPT adversary $\mathcal{A}$ wins Game 6. We show how a simulator $\mathcal{S}$ can use this to break the FDHI assumption. Given $(g_1, g_2, g_2^z, g_2^v, g_1^{\frac{z}{v}}, g_1^r, g_1^{\frac{r}{v}})$, $\mathcal{S}$ simulates Game 6.

Setup Simulator $\mathcal{S}$ chooses an index $\mu \in [Q]$ uniformly at random. During the key generation, it chooses $a_i, b_i, k_i, y \in \mathbb{Z}_p$ uniformly at random for all $i \in [n]$. It sets $f_{\tau_i} = g_t^{ya_i} \cdot e(g_1, g_2^z)^{yb_i}$, $F_{\tau_i} = g_2^{a_i} \cdot (g_2^z)^{b_i}$, for all $i \in [n]$, as well as $f_{\tau_i,\tau_j} = g_t^{k_j y a_i} \cdot e(g_1, g_2^z)^{k_j y b_i}$, for all $i, j \in [n]$. It then chooses $x \in \mathbb{Z}_p$ uniformly at random. It sets $h_t = e(g_1, g_2)^x$. Additionally it generates a key pair $(\mathsf{sk}', \mathsf{pk}') \leftarrow \mathsf{KeyGen}'(1^\lambda)$ of a regular signature scheme. It gives the public key $\mathsf{pk} = (\mathsf{pk}', \mathsf{bgp}, h_t, \{F_{\tau_i}, f_{\tau_i}\}_{i=1}^n, \{f_{\tau_i,\tau_j}\}_{i,j=1}^n)$ to $\mathcal{A}$.

Queries Let l be a counter for the number of datasets queried by $\mathcal{A}$ (initially, it sets $l = 1$). For every new queried dataset Δ, simulator $\mathcal{S}$ creates a list $\mathcal{Q}_\Delta$ of tuples (τ, m), which collects all the label/message pairs queried by the adversary on Δ. Moreover, whenever the l-th new dataset Δ_l is queried, $\mathcal{S}$ does the following: If $l = \mu$, it samples a random $\zeta_\mu \in \mathbb{Z}_p$, sets $Z_\mu = (g_2^z)^{\frac{1}{\zeta_\mu}}$ and stores ζ_μ. If $l \neq \mu$, it samples a random $\zeta_l \in \mathbb{Z}_p$ and sets $Z_l = (g_2^v)^{\frac{1}{\zeta_l}}$ and stores ζ_l. Since all Z_l are randomly distributed in $\mathbb{G}_2$, they have the same distribution as in Game 6. Given a query (Δ, τ, m) with $\Delta = \Delta_m$, simulator $\mathcal{S}$ first computes $\sigma_{\Delta_l} \leftarrow \mathsf{Sign}'(\mathsf{sk}', Z_l | \Delta_l)$. If $l \neq \mu$, it samples $\rho_\tau, s_\tau \in \mathbb{Z}_p$ uniformly at random and computes $\Lambda_\tau = \left((g_1^{\frac{z}{v}})^{(y+s_\tau)b_\tau} \cdot (g_1^{\frac{r}{v}})^{\rho_\tau}\right)^{\zeta_l}$, $R_\tau = g_1^{-mx} \cdot g_1^{-(y+s_\tau)a_\tau} \cdot (g_1^r)^{\rho_\tau}$, $S_\tau = g_1^{s_\tau}$, $T_\tau = g_1^{mxy-k_\tau}$, $\mathcal{T} = \{(\tau, S_\tau, T_\tau)\}$ and gives $\sigma = (m, \sigma_{\Delta_l}, Z_l, \Lambda_\tau, R_\tau, \mathcal{T})$ to $\mathcal{A}$. We have

$$\begin{aligned} e(\Lambda_\tau, Z_l) &= e\left((g_1^{\frac{z}{v}})^{(y+s_\tau)b_\tau} \cdot (g_1^{\frac{r}{v}})^{\rho_\tau}, g_2^{\frac{v}{\zeta_l}}\right)^{\zeta_l} = e\left((g_1^{\frac{z}{v}})^{(y+s_\tau)b_\tau} \cdot (g_1^{\frac{r}{v}})^{\rho_\tau}, g_2^v\right) \\ &= g_t^{z(y+s_\tau)b_\tau + r\rho_\tau} = g_t^{z(y+s_\tau)b_\tau + r\rho_\tau + ym - ym + a_\tau(y+s_\tau) - a_\tau(y+s_\tau)} \\ &= g_t^{ym} \cdot g_t^{y(a_\tau + zb_\tau)} \cdot g_t^{-ym - ya_\tau + r\rho_\tau} \cdot g_t^{s_\tau(a_\tau + zb_\tau)} = h_t^m \cdot f_\tau \cdot e(R_\tau, g_2) \cdot e(S_\tau, F_\tau) \end{aligned}$$

thus this output is indistinguishable from the challenger's output during Game 6.
If $l = \mu$, simulator $\mathcal{S}$ computes
$\Lambda_\tau = \left(g_1^{(y+s_\tau)b_\tau}\right)^{\zeta_\mu}$, $R_\tau = g_1^{-mx-(y+s_\tau)a_\tau}$, $S_\tau = g_1^{s_\tau}$, $T_\tau = g_1^{mxy-k_\tau}$, $\mathcal{T} = \{(\tau, S_\tau, T_\tau)\}$ and gives $\sigma = (m, \sigma_{\Delta_l}, Z_l, \Lambda_\tau, R_\tau, \mathcal{T})$ to $\mathcal{A}$. We have

$$\begin{aligned} e(\Lambda_\tau, Z_\mu) &= e\left(g_1^{(y+s_\tau)b_\tau}, g_2^{\frac{z}{\zeta_\mu}}\right)^{\zeta_\mu} = e\left(g_1^{(y+s_\tau)b_\tau}, g_2^{z}\right) = g_t^{z(y+s_\tau)b_\tau} \\ &= g_t^{z(y+s_\tau)b_\tau + xm - xm + (y+s_\tau)a_\tau - (y+s_\tau)a_\tau} = g_t^{mx} \cdot g_t^{y(a_\tau + zb_\tau)} \cdot g_t^{-mx-ya_\tau} \cdot g_t^{s_\tau(a_\tau+zb_\tau)} \\ &= h_t^m \cdot f_\tau \cdot e\left(R_\tau, g_2\right) \cdot e\left(S_\tau, F_\tau\right) \end{aligned}$$

hence this output is indistinguishable from the challenge's output during Game 6.

Forgery Let $(\mathcal{P}^*{}_{\Delta^*}, \sigma^*)$ be a forgery with $\sigma^* = (m^*, \sigma^*_{\Delta^*}, Z^*, \Lambda^*, R^*, \mathcal{L}^*)$ and $\mathcal{L}^* = \{(\tau, S^*_\tau)\}_{\tau \in \mathcal{I}}$ where $\mathcal{I}$ is a subset of the label space, be the forgery returned by $\mathcal{A}$. The case of σ^* as a level-1 signature is just a simplification of the level-2 case and is omitted.
$\mathcal{S}$ computes $\hat{\sigma} \leftarrow \mathsf{HEval}(\mathsf{pk}, \mathcal{P}^*_{\Delta}, \boldsymbol{\sigma})$ over the signatures σ_i generated in dataset Δ^*.
$\mathcal{S}$ parses $\hat{\sigma} = (\hat{m}, \sigma^*_{\Delta^*}, Z^*, \hat{\Lambda}, \hat{R}, \hat{\mathcal{L}})$ with $\hat{\mathcal{L}} = \{(\tau, \hat{S}_\tau)\}_{\tau \in \hat{\mathcal{I}}}$ where $\hat{\mathcal{I}}$ is a subset of the label space. Without loss of generality, we assume $\mathcal{I} = \hat{\mathcal{I}} = \{\tau_i\}_{i \in [n]}$, i.e. $\mathcal{I}$ is the whole label space. We can always append $\mathcal{L}$ with $(\tau_i, 1_{\mathbb{G}_1})$, and write $S_{\tau_i} = S_i$.
If Game 6 outputs 1, we have $Z^* = Z_\mu$, $\Lambda^* = \hat{\Lambda}$, as well as

$$e\left(\Lambda^*, Z_\mu\right) = e\left(R^*, g_2\right) \cdot h_t^m \cdot \prod_{i,j=1}^{n} f_{i,j}^{c_{i,j}} \cdot \prod_{j=1}^{n} f_j^{c_j} \cdot \prod_{i=1}^{n} e\left(S^*_{\tau_i}, F_{\tau_i}\right) \quad \text{and}$$

$$e\left(\hat{\Lambda}, Z_\mu\right) = e\left(\hat{R}, g_2\right) \cdot h_t^m \cdot \prod_{i,j=1}^{n} f_{i,j}^{c_{i,j}} \cdot \prod_{j=1}^{n} f_j^{c_j} \cdot \prod_{i=1}^{n} e\left(\hat{S}_{\tau_i}, F_{\tau_i}\right).$$

Dividing those equations yields

$$\frac{\Lambda^*}{\hat{\Lambda}} = \left(\frac{R^*}{\hat{R}} \cdot g_1^{x(m^*-\hat{m})} \cdot \prod_{i=1}^{n} \frac{S_i^{*\,a_i + b_i z}}{\hat{S}_i}\right)^{\frac{\zeta_\mu}{z}} = \left(\frac{R^*}{\hat{R}} \cdot g_1^{x(m^*-\hat{m})} \cdot \prod_{i=1}^{n} \frac{S_i^{*\,a_i}}{\hat{S}_i}\right)^{\frac{\zeta_\mu}{z}} \cdot \prod_{i=1}^{n} \frac{S_i^{*\,b_i \zeta_\mu}}{\hat{S}_i}$$

Thus $\mathcal{S}$ can compute $W = \left(\frac{R^*}{\hat{R}} \cdot g_1^{x(m^*-\hat{m})} \cdot \prod_{i=1}^{n} \frac{S_i^{*\,a_i}}{\hat{S}_i}\right)^{\zeta_\mu}$, $W' = \frac{\Lambda^*}{\hat{\Lambda}} \cdot \prod_{i=1}^{n} \frac{S_i^{*\,-b_i\zeta_\mu}}{\hat{S}_i}$, and return W, W' as a solution to the FDHI problem. Since we have $\mathsf{bad}_6 = \mathsf{false}$, we have $W' \neq 1$. Our simulation has the same distribution as a real execution of Game 6.