Fully-Succinct Multi-Key Homomorphic Signatures from Standard Assumptions

Gaspard Anthoine[1,2], David Balbás[1,2], and Dario Fiore[1]

[1] IMDEA Software Institute, Madrid, Spain
{gaspard.anthoine, david.balbas, dario.fiore}@imdea.org
[2] Universidad Politécnica de Madrid, Madrid, Spain

Abstract. Multi-Key Homomorphic Signatures (MKHS) allow one to evaluate a function on data signed by distinct users while producing a succinct and publicly-verifiable certificate of the correctness of the result. All the constructions of MKHS in the state of the art achieve a weak level of succinctness where signatures are succinct in the total number of inputs but *grow linearly with the number of users* involved in the computation. The only exception is a SNARK-based construction which relies on a strong notion of knowledge soundness in the presence of signing oracles that not only requires non-falsifiable assumptions but also encounters some impossibility results.

In this work, we present the first construction of MKHS that are fully succinct (also with respect to the number of users) while achieving adaptive security under standard falsifiable assumptions. Our result is achieved through a novel combination of batch arguments for NP (BARGs) and functional commitments (FCs), and yields diverse MKHS instantiations for circuits of unbounded depth based on either pairing or lattice assumptions. Additionally, our schemes support efficient verification with pre-processing, and they can easily be extended to achieve multi-hop evaluation and context-hiding.

1 Introduction

The rise of decentralized and remote computing has sparked an interest in cryptographic solutions for secure outsourcing of data to untrusted parties. In parallel to the effort of ensuring data privacy, which is the object of study of, e.g., works on fully homomorphic encryption [Gen09], another important goal is to provide *authenticity* for the data used during computation – a problem that can be described as follows.

Consider a scenario where a user, Alice, authenticates a large data set $m_1, \ldots, m_n$, producing signatures $\sigma_1, \ldots, \sigma_n$, and stores both data and signatures on an untrusted platform. Subsequently, a third entity, called the *evaluator*, performs a computation on Alice's data, denoted as $y = f(m_1, \ldots, m_n)$, and sends y to another user, Bob. *How can the evaluator convince Bob that y is the correct result obtained by running f on data signed by Alice?* A naive solution is for the evaluator to send to Bob the n inputs along with the corresponding signatures, and for Bob to verify their validity and to recompute f. A natural question is whether the evaluator can convince Bob with *short communication*, i.e., by sending $o(n)$ bits of information.

Homomorphic Signatures. *Homomorphic signatures* (HS) [JMSW02] stand out as a solution for the above problem of *authenticity-preserving computation.* They allow the evaluator to compute on signed data, deriving not only the output y but also a signature $\sigma_{f,y}$. Anyone can publicly verify the tuple $(f, y, \sigma_{f,y})$ and get convinced of the correctness of y as the result of computing f on Alice's data, without having to download the large data input. Crucially, in HS the signature $\sigma_{f,y}$ is

A short version of this paper appears in the proceedings of CRYPTO 2024.

succinct, namely its size should remain independent, or at most grow sublinearly in the number of messages n used in the computation.[3]

Multi-Key Homomorphic Signatures. In many real-world scenarios, however, computations are performed on data that belongs to (and is authenticated by) multiple entities. Typical examples include aggregating data collected by several hospitals for clinical studies, smart monitoring of signals produced by IoT devices (e.g., medical/environmental/traffic sensors, wearable devices, etc.), or transactions made by different users in a blockchain. In this context, the standard notion of homomorphic signatures falls short, since it requires that all messages are signed under the same key. To address this issue, Fiore, Mitrokotsa, Nizzardo, and Pagnin introduced *multi-key homomorphic signatures* (MKHS) [FMNP16]. In a MHKS, the *evaluator* computes a function f over n messages $m_1, \ldots, m_n$, where each m_i is authenticated by someone in a set of t parties that we denote by $\mathsf{id}_1, \ldots, \mathsf{id}_t$. In this case, the resulting signature $\sigma_{f,y}$ must vouch for the correctness of y as output of f on inputs that were signed under public signature keys $\mathsf{vk}_1, \ldots, \mathsf{vk}_t$, where each vk_i corresponds to id_is.

The construction of succinct MKHS conveys a greater challenge than for their single-user counterparts. All the MKHS constructions in the standard model [FMNP16, FP18, SBB19, SFVA21] achieve only a weak notion of succinctness in which, for a function f with n inputs signed by t distinct users, the size of the evaluated signature $\sigma_{f,y}$ grows as $\mathsf{poly}(\lambda, t, \log n)$, i.e., at least linearly in the number t of users involved in the computation. Even if this level of succinctness may be acceptable in applications where a few users provide each a large amount of data, it is clearly undesirable in scenarios that involve computing on data from many parties, such as the case of IoT sensors or users in a blockchain.

The only MKHS construction that overcomes this succinctness limitation uses SNARKs [LTWC18], which in this context are a double-edged sword. On the good side, SNARKs lead to fully succinct signatures whose size grows only polynomially in the security parameter, i.e., $|\sigma_{f,y}| = \mathsf{poly}(\lambda)$. As a drawback, the security of SNARK-based MKHS relies on non-falsifiable assumptions, since this is known to be the case for SNARKs [GW11]. Even more problematically, the security of MKHS from SNARKs [LTWC18] needs the stronger notion of *knowledge-soundness in the presence of signing oracles*, for which there are some impossibility results [FN16].

The state of the art in MKHS therefore raises the following open question:

Is it possible to build a fully-succinct multi-key homomorphic signature scheme under standard falsifiable assumptions?

1.1 Our Contribution

In this paper, we answer the above question in the affirmative, proposing *the first multi-key homomorphic signatures that achieve full succinctness while being secure in the standard model under falsifiable assumptions*. Our construction relies on a novel combination of standard digital signatures, succinct functional commitments (FC) [LRY16], and batch arguments for NP (BARG) [KPY19, CJJ21]. Our MKHS allows the evaluation of the same functions supported by the FC scheme, and inherits succinctness from the succinctness of the FC and of the BARG. We present a simplified version of our main theorem below.

[3] HS may incorporate additional useful properties, such as amortized efficiency (enabling verification in time independent of the complexity of f, after preprocessing) and context-hiding (preventing the verifier to learn information on the inputs beyond the computation's output); see Section 3 for more details.

Theorem 3 (simplified). *Let* FC *be a functional commitment scheme for a class of functions* $\mathcal{F}$*,* BARG *a somewhere-extractable batch argument for NP,* SEC *a somewhere extractable commitment, and* Σ *a digital signature scheme. Then, there exists an adaptively-secure multi-key homomorphic signature* MKHS *for* $\mathcal{F}$*. Moreover, if the* BARG *generates proofs of size* s_{BARG} *and the* FC *generates proofs of size* s_{FC}*, then the signatures produced by* MKHS *have size* $s_{\mathsf{MKHS}} \approx s_{\mathsf{FC}} + s_{\mathsf{BARG}}$*.*

Both BARGs and FCs have been in the spotlight in recent years and currently offer several instantiations from different (falsifiable) assumptions, which in turn yield MKHS for all functions from a variety of assumptions. For instance, we can instantiate our MKHS from building blocks based on correlation-intractable hash functions and probabilistic checkable proofs, such as the BARGs from [CJJ21, CJJ22, CGJ+23] and an FC for circuits based on the SNARG for P from [KLVW23], to obtain constructions from standard assumptions such as LWE or subexponential DDH. Alternatively, we can use the algebraic BARG of [WW22] based on the k-Lin assumption, and the algebraic FC from [BCFL23] based on the (falsifiable) HiKer assumption, obtaining a pairing-based construction for unbounded-depth circuits. We summarize these instantiations below.

Corollary 2 (simplified). *Assuming the hardness of either (1) subexponential DDH, or (2) learning with errors, there exists a multi-key homomorphic signature* MKHS *for boolean circuits of unbounded depth* d *with public parameter size* $\mathsf{poly}(\lambda, \log n)$ *and signature size* $\mathsf{poly}(\lambda, \log n) \cdot d$*.*

Corollary 3 (simplified). *Assuming the hardness of HiKer and* k*-Lin for* $k \geq 2$*, there exists a multi-key homomorphic signature* MKHS *for arithmetic circuits of unbounded depth* d *and bounded width* w *from algebraic building blocks, with public parameter size* $\mathcal{O}(w^5)$ *and signature size* $\mathcal{O}(\lambda \cdot d^2) + \mathsf{poly}(\lambda)$*.*

Additional Properties. Compared to the weakly succinct scheme of [FMNP16], our MKHS schemes achieve a variety of useful properties. First, we do not need to bound a priori the number of values to be signed, but only the class of functions (to the extent required by the FC); this feature is useful in applications where one computes on portions of very large data (e.g., sliding-window statistics on unbounded data streams). Second, our schemes are secure against adversaries that can adaptively corrupt users, whereas [FMNP16] can only handle non-adaptive corruptions. Third, our MKHS have efficient verification time, after preprocessing the function; this is similar to [FMNP16] though we support a more flexible preprocessing model (see Section 3.2). Furthermore, all our instantiations allow multi-hop sequential composition of different functions (Section 5.1) and can be compiled to provide context-hiding via a generic NIZK-based technique (Theorem 2).

Other Contributions. In order to broaden the instantiations of our constructions and to enable the evaluation of unbounded-depth circuits, we give two additional results on output-succinctness and unbounded FCs, which are generic and may be of independent interest.

- In **Theorem 1**, we show that any succinct FC for n-to-1 functions can be transformed into a n-to-m FC that is fully succinct in the output, i.e., the proof size does not grow with m.
- In **Theorem 4**, we show that any suitably expressive FC can be boosted into a single-input *chainable* FC [BCFL23], which is sufficient to construct FC schemes for unbounded-depth circuits.

Beyond our result for MKHS, the techniques underlying our construction present, to the best of our knowledge, a novel approach for building an advanced cryptographic primitive that was only known to be (with full succinctness) realizable from SNARKs. We expect that our techniques can be applied in other settings, leading to further constructions of advanced primitives from standard assumptions.

1.2 Technical Overview

Background: Labeled Programs. In a multi-key homomorphic signature scheme, the evaluator must declare the evaluated function as a *labeled program* [GW13]. A labeled program is specified by a tuple $(f, \ell_1, \ldots, \ell_n)$ where $f : \mathcal{M}^n \to \mathcal{M}^m$ is a function represented by an arithmetic or boolean circuit, and the ℓ_i are labels of the inputs. Without loss of generality, we assume that $\ell_i := (\mathsf{id}_i, \tau_i)$, where id_i is an identity and τ_i an arbitrary string.

Upon evaluating the homomorphic signature, the n messages $m_1, \ldots, m_n \in \mathcal{M}$ that are collected by the evaluator are each uniquely associated to labels $\ell_1, \ldots, \ell_n$, and therefore to identities $\mathsf{id}_1, \ldots, \mathsf{id}_n$ (not necessarily all distinct). Additionally, each of these identities is associated to a public key vk_i, that can be used to verify the authenticity of a message-label pair (m_i, ℓ_i). Program labelling is required to properly define both correctness and security of MKHS, since e.g. otherwise the order in which the m_i's are input to f is unspecified.

Warm-Up: Aggregating Signatures with BARGs . The initial inspiration for our MKHS construction lies in the mechanism to construct aggregate signatures from Waters and Wu [WW22]. In an *aggregate signature* scheme [BGLS03], an aggregator (or also evaluator) can take multiple message-signature pairs $(m_1, \sigma_1), \ldots, (m_n, \sigma_n)$ from different users, and compress all the signatures into a succinct σ_{Agg}. Their construction is based on *batch arguments for NP* (BARGs), which are a standard-model proof system for batches of NP statements. In other words, given a boolean circuit $\mathcal{C}$, a BARG allows one to prove that n statements $\mathtt{x}_1, \ldots, \mathtt{x}_n$, have NP witnesses $\mathtt{w}_1, \ldots, \mathtt{w}_n$ such that $\mathcal{C}(\mathtt{x}_i, \mathtt{w}_i)$ accepts for every $i \in [n]$. Moreover, the size of the proof π grows sublinearly with n.

To construct aggregate signatures, [WW22] start from any digital signature scheme $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$. Then, to sign messages, aggregate signatures, and verify the aggregation proof, their algorithm broadly proceeds as follows:

- Aggregate: To aggregate n message-signature pairs, $(m_1, \sigma_1), \ldots, (m_n, \sigma_n)$, let $\mathtt{x}_i = (m_i, \mathsf{vk}_i)$ be the statements and $\mathtt{w}_i = \sigma_i$ be the witnesses for the circuit $\mathcal{C}(\mathtt{x}_i, \mathtt{w}_i)$ that checks:

$$\Sigma.\mathsf{Ver}(\mathsf{vk}_i, m_i, \sigma_i) = 1$$

  Then, the aggregate signature σ_{Agg} is a BARG proof on $(\mathcal{C}, \{\mathtt{x}_i\}, \{\mathtt{w}_i\})$.
- Verify: To verify σ_{Agg}, one runs the BARG verification algorithm on $(\mathcal{C}, \{\mathtt{x}_i\})$.

A mechanism to aggregate signatures can be seen as the "first step" of the evaluation of a fully-fledged MKHS. Indeed, similarly to aggregate signatures, in MKHS one also needs to prove that all signatures and messages are valid in a succinct manner. However, while in aggregate signatures the verifier knows all the messages $(m_1, \ldots, m_n)$, in MKHS one only knows the result $\boldsymbol{y} = f(m_1, \ldots, m_n)$. Therefore the evaluation step must additionally prove in a succinct manner the correctness of f's computation. This is what makes the realization of fully succinct MKHS challenging. A natural attempt to deal with this problem is to extend the BARG circuit by placing m_i in the witness, and by additionally proving that $\boldsymbol{y} = f(m_1, \ldots, m_n)$. An example of such a language could be the following:

$$\Sigma.\mathsf{Ver}(\mathsf{vk}_i, m_i | \ell_i, \sigma_i) = 1 \ \wedge \ \boldsymbol{y} = f(m_1, \ldots, m_n),$$

where $\mathtt{x}_i = (\ell_i, \mathsf{vk}_i, f, \boldsymbol{y})$ and $\mathtt{w}_i = (m_i, \sigma_i)$. Unfortunately, the computation of f can be "global", i.e., involve messages from all the witnesses, and thus the language is not compatible with that supported by BARGs.

Proving $\boldsymbol{f(m_1, \ldots, m_n)}$: Functional Commitments. To address the problem of f being global, our second idea is to resort to *functional commitments* (FCs). A functional commitment scheme [LRY16] allows an entity to first commit to some input $\boldsymbol{x}$ in c, and then open c to $f(\boldsymbol{x})$ for some function $f \in \mathcal{F}$, where $\mathcal{F}$ is the class of functions supported by the scheme. Importantly for our goal, FCs ensure commitments and openings to be succinct and can be realized from falsifiable assumptions.

By using FCs, the evaluator could create a commitment c to the inputs of the computation $(m_1, \ldots, m_n)$ and then use the opening feature to prove the evaluation $\boldsymbol{y} = f(m_1, \ldots, m_n)$. This way we can take this "global" task outside of the BARG. Unfortunately, doing two separate proofs, one for the BARG and one for the FC, does not suffice. The issue is that there is no connection between the m_i committed inside the FC and the messages whose signatures are verified in the BARG circuit $\mathcal{C}$, i.e., in $\Sigma.\mathsf{Ver}(\mathsf{vk}_i, m_i|\ell_i, \sigma_i)$. To integrate FCs and BARGs into a working solution, we need to be able to link the commitment c to the messages m_i that are in the witnesses $\mathtt{w}_i$ of $\mathcal{C}$. One natural example of such a connection may consist of showing that, at every local step i, c opens to message m_i at position i (i.e., à la vector commitment), a local check that could be easily integrated in the circuit $\mathcal{C}(\mathtt{x}_i, \mathtt{w}_i)$ and proven with a BARG. This approach, while giving correctness, is unsuccessful for the security proof. At a very high level, the MKHS adversary produces a forgery which contains a commitment c^* and a functional-opening π^* to $\boldsymbol{y}^* \neq f(m_1, \ldots, m_n)$. To break the security of the FC we would need to come up with another functional-opening to a different value, say the honest output $f(m_1, \ldots, m_n)$. The reduction could compute this by itself if we had the guarantee that c^* is a commitment to $(m_1, \ldots, m_n)$ but this is not ensured; we can only use the BARG to extract, for a single index i at a time, a position-opening to a validly signed m_i at position i in c^*. This is however not enough to break the evaluation binding of the FC.

Our Solution: Proving FC updates in the BARG. To get around the above problem, our approach consists of *iteratively computing* c *inside the BARG circuit.* We start by defining a sequence of partial commitments $\mathsf{c}_0, \ldots, \mathsf{c}_n$, where the i-th commitment commits to the first i messages. Namely, let $\mathsf{c}_i \leftarrow \mathsf{FC.Com}(\mathsf{ck}, (m_1, \ldots, m_i, 0, \ldots, 0))$. Then, at step i of the BARG proof, $\mathcal{C}(\mathtt{x}_i, \mathtt{w}_i)$ verifies a proof π_i that c_i and c_{i-1} only differ on m_i at position i. In other words, that if we update c_{i-1} with m_i at position i, then we obtain c_i.

For this idea to work, we require two properties from our FC. One, *determinism*, such that we can compare commitments without having to open them. Two, *local updatability*, such that there exists an efficient update verification algorithm $\mathsf{FC.VerUpd}$ that runs in constant (or at least sublinear) time in n. Moreover, update verification should only require a succinct section ck_i of the commitment key. We describe a simplified version of the resulting BARG circuit in Figure 1.

Given the description of $\mathcal{C}$, our construction of MKHS can be summarized as follows:

- Sign: To sign a message m_i with label ℓ_i under key sk_i, compute and output $\sigma_i \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}_i, m_i|\ell_i)$.
- Evaluate: To evaluate $(f, \ell_1, \ldots, \ell_n)$ on n message-signature pairs $(m_1, \sigma_1), \ldots, (m_n, \sigma_n)$, compute:
  - An FC commitment $\mathsf{c} \leftarrow \mathsf{FC.Com}(\mathsf{ck}, (m_1, \ldots, m_n))$.
  - A BARG proof π_σ for $\mathcal{C}(\mathtt{x}_1, \mathtt{w}_1) \wedge \cdots \wedge \mathcal{C}(\mathtt{x}_n, \mathtt{w}_n)$.
  - An FC opening proof π_f that c opens to $\boldsymbol{y} = f(m_1, \ldots, m_n)$ on f.

Description of $\mathcal{C}(\mathtt{x}, \mathtt{w})$ (simplified):
Statement: $\mathtt{x} = (\mathsf{vk}_i, \mathsf{ck}_i, \ell_i, i)$
Witness: $\mathtt{w} = (m_i, \sigma_i, \pi_i, \mathsf{c}_{i-1}, \mathsf{c}_i)$
Circuit:

- If $i = 1$, check that $\mathsf{c}_{i-1} = \mathsf{FC.Com}(\mathsf{ck}, \mathbf{0})$.
- If $i = n$, check that $\mathsf{c}_i = \mathsf{c}$.
- Check that:

$$\Sigma.\mathsf{Ver}(\mathsf{vk}_i, m_i | \ell_i, \sigma_i) = 1$$
$$\wedge\ \mathsf{FC.VerUpd}(\mathsf{ck}_i, i, \mathsf{c}_{i-1}, 0, \mathsf{c}_i, m_i, \pi_i) = 1$$

Fig. 1: Simplified description of the BARG circuit $\mathcal{C}$ in our MKHS construction. The commitment c is hardwired into the circuit.

Then, the output signature is $\sigma_{f,y} = (\mathsf{c}, \pi_\sigma, \pi_f)$.

- Verify: To verify $\sigma_{f,y}$, simply check the BARG and FC proofs w.r.t. c.

We note that our actual construction in Section 4 is slightly more complex, as it additionally involves a somewhere extractable commitment scheme (SEC) which we require to connect the consecutive steps $i-1$ and i of the BARG and for the security proof to go through.

Security and Proof Strategy. The security notion for MKHS considers adversaries that can make signing queries for messages and labels of their choice and it captures that it should be hard for the adversary to (1) claim valid messages and signatures that were never received from the signing oracle, and (2) forge the output of the computation of the labeled program $(f, \ell_1, \ldots, \ell_n)$. The notion is adaptive as the adversary may arbitrarily expose parties' secret keys, yet compromised keys cannot be involved in a forgery.

Our security proof proceeds by partitioning the winning condition in multiple events, according to the type of forgery that is produced by the adversary, and then handles each event separately. The most interesting component of the proof, and arguably the hardest technical challenge of this work, is to deal with the event when the adversary produces a forgery for $\boldsymbol{y} \neq f(m_1, \ldots, m_n)$, where the (deterministic) commitment to the messages c^* output by $\mathcal{A}$ is dishonest, $\mathsf{c}^* \neq \mathsf{FC.Com}(\mathsf{ck}, (m_1, \ldots, m_n))$.

To bound the probability of this event, the general proof strategy is to show that all partial commitments c_i for $i \in [n]$ must have been computed honestly. We define multiple hybrids for each index i, which implement a 'sliding window' strategy where we roughly: (1) extract from both the BARG and the SEC at step i, (2) compare the extracted c_i to their honest counterparts, and (3) extract the message m_i and signature σ_i (a potential forgery) from the adversary's output, such that we can certify the validity of the i-th update. Then, we "reboot" the BARG and SEC extraction and start again at step $i+1$. From the above proof strategy, steps (1) and (2) follow the blueprint of a line of work on succinct delegation schemes (also known as SNARGs for P) [KPY19, GZ21, KVZ21, CJJ22, KLVW23], whereas step (3) requires to go a few steps beyond. Notably, in contrast to delegation schemes where the proven computation is deterministic, in our MKHS scheme the statement includes a non-deterministic part, messages and signatures, that are not available to the verifier.

1.3 Related Work

Homomorphic Signatures. The concept of homomorphic signatures was introduced by Desmedt [Des93] and Johnson et al. [JMSW02] and properly formalized by Boneh and Freeman [BF11]. Starting from seminal works on *linearly-homomorphic* signatures, e.g., [BFKW09, GKKR10, AL11, CFW12, Fre12, LPJY13, CFGV13, CFN15], the expressivity of HS has significantly improved, capturing bounded-degree polynomials [BF11, CFW14, CFT22], and circuits of logarithmic depth [KNYY19, CFT22], bounded polynomial depth [GVW15], and unbounded depth [BCFL23, GU24]. Among these works, the closest to ours in terms of techniques is that of Catalano, Fiore and Tucker [CFT22] who first proposed to use functional commitments to build HS. In their solution, each signer signs a commitment to the vector with m_i in position i and 0 elsewhere; the evaluator builds a commitment to the inputs using the additive homomorphic property and uses a (single-key) linearly-homomorphic signature to prove that the commitment is correctly aggregated. Unfortunately generalizing this approach to the multi-key setting fails, as there exists no fully-succinct MKHS scheme, not even for linear functions. Our solution uses a similar idea of employing an FC of the inputs; however we develop a different set of techniques to validate the correct authentication, in the multi-key setting, of the committed inputs.

Multi-Key HS. Fiore et al. [FMNP16] introduced the definition of MKHS and proposed a construction that supports circuits of bounded depth and is weakly succinct (see earlier for a detailed comparison). Lai et al. [LTWC18] proposed the first fully succinct MKHS by using SNARKs. Compared to ours, their construction achieves the stronger notion of *unforgeability under insider corruption*, which tolerates adversaries that can even corrupt users involved in the input of a computation. Unfortunately [LTWC18] also shows that MKHS secure in this model imply SNARGs and thus need non-falsifiable assumptions. The state of the art in MKHS includes works that have investigated how to construct MKHS schemes starting from single-key HS [FP18, SFVA21], as well as constructions that aim for concrete efficiency for linear functions [AP19, SBB19]. However, with the only exception of the SNARK-based solution of [LTWC18], all these works feature signatures whose size grows linearly in the number of users.

Aggregate Signatures. The concept of aggregate signatures was introduced by Boneh et. al. [BGLS03]. Their initial construction was pairing-based and relied on random oracles. Since then, constructions have also been proposed from multilinear maps [RS09] and indistinguishability obfuscation [HKW15]. In recent years, progress on building BARGs for NP sparked multiple constructions of aggregate signatures from standard assumptions. Examples include [CJJ21, DGKV22, WW22, Goy24] for n-out-of-n policies, and [NWW23, BCJP24] for monotone policies.

2 Preliminaries

Notation. We denote by $\mathbb{N}$ the set of natural numbers > 0. We denote the security parameter by $\lambda \in \mathbb{N}$. We call a function ϵ *negligible*, denoted $\epsilon(\lambda) = \mathsf{negl}(\lambda)$, if $\epsilon(\lambda) = O(\lambda^{-c})$ for every constant $c > 0$, and call a function $p(\lambda)$ *polynomial*, denoted poly, if $p(\lambda) = O(\lambda^c)$ for some constant $c > 0$. We say that an algorithm is *probabilistic polynomial time* (PPT) if it consumes randomness and its running time is bounded by some $p(\lambda) = \mathsf{poly}(\lambda)$. For a finite set S, $x \leftarrow\!\!\$\, S$ denotes sampling x uniformly at random in S. For an algorithm A, we write $y \leftarrow A(x)$ for the output of A on input x. For a positive $n \in \mathbb{N}$, $[n]$ is the set $\{1, \dots, n\}$. We denote vectors $\boldsymbol{x}$ and matrices $\mathbf{M}$ using bold fonts. We define a message space $\mathcal{M}$ which is common to all the cryptographic primitives used in this work. The operator $|$ refers to the concatenation.

2.1 Digital Signatures

Definition 1 (Digital signature). *A digital signature scheme* $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$ *is defined as the following tuple of efficient algorithms.*

$\mathsf{KeyGen}(1^\lambda) \to (\mathsf{sk}, \mathsf{vk})$**:** *On input the security parameter, creates a public-private key pair* $(\mathsf{sk}, \mathsf{vk})$
$\mathsf{Sign}(\mathsf{sk}, m) \to \sigma$**:** *On input a message* $m \in \mathcal{M}$ *and the secret key* sk*, generates a signature* σ*.*
$\mathsf{Ver}(\mathsf{vk}, \sigma, m) \to b$**:** *Given a signature* σ*, a message* $m \in \mathcal{M}$ *and a public key* pk*, outputs* $b \in \{0, 1\}$*, indicating acceptance or rejection.*

We say that the signature scheme is correct if for any admissible $m \in \mathcal{M}$ *and all choices of randomness, if* $(\mathsf{sk}, \mathsf{vk}) \leftarrow\$ \mathsf{KeyGen}(1^\lambda)$ *and* $\sigma \leftarrow\$ \mathsf{Sign}(\mathsf{sk}, m)$*, then* $\mathsf{Ver}(\mathsf{vk}, \sigma, m) = 1$*.*

Definition 2 (EUF-CMA security for signatures). Let Σ be a signature scheme. Existential unforgeability, or EUF-CMA security, for Σ is defined via the game $\mathsf{EUF\text{-}CMA}_{\mathcal{A},\Sigma}(\lambda)$ depicted in Figure 2. We define the advantage of adversary $\mathcal{A}$ in the game

$$\mathsf{Adv}^{\mathrm{eufcma}}_{\mathcal{A},\Sigma}(\lambda) := \Pr[\mathsf{EUF\text{-}CMA}_{\mathcal{A},\Sigma}(\lambda) = 1].$$

We say that Σ is EUF-CMA if for all PPT adversaries $\mathcal{A}$ we have $\mathsf{Adv}^{\mathrm{eufcma}}_{\mathcal{A},\Sigma}(\lambda) = \mathsf{negl}(\lambda)$.

$\mathsf{EUF\text{-}CMA}_{\mathcal{A},\Sigma}(\lambda)$:	**Oracle** $\mathcal{O}^{\mathsf{Sign}}(m)$
$(\mathsf{sk}, \mathsf{vk}) \leftarrow\$ \Sigma.\mathsf{KeyGen}(1^\lambda)$	$\sigma \leftarrow\$ \Sigma.\mathsf{Sign}(\mathsf{sk}, m)$
$\mathsf{L_{Sig}} \leftarrow \emptyset$	$\mathsf{L_{Sig}} \leftarrow \mathsf{L_{Sig}} \cup \{m\}$
$(m, \sigma) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Sign}}}(\mathsf{pk})$	**return** σ
Output 1 iff $\Sigma.\mathsf{Ver}(\mathsf{pk}, \sigma, m) = 1 \wedge m \notin \mathsf{L_{Sig}}$	

Fig. 2: EUF-CMA security game for a signature scheme Σ.

2.2 Somewhere Extractable Commitments

We recall the notion of somewhere extractable commitment scheme from [CJJ22, WW22], which is closely related to the notion of somewhere statistically binding hash functions introduced in [HW15, OPWW15]. In a nutshell, a somewhere extractable commitment is a vector commitment [CF13] with a dual-mode, programmable commitment key dk. In extractable mode, a trapdoor td allows one to extract at the programmed location i^*. Moreover, a commitment key in extractable mode is indistiguishable from a key in normal mode.

Without loss of generality, we adopt the convention that SEC admits local verification, where the commitment key dk naturally splits into sub-keys $\{\mathsf{dk}_i\}_{i \in [n]}$, not necessarily disjoint. Then, the verification algorithm at location i requires only to read dk_i. If this property does not apply to a given SEC scheme, one can simply let $\mathsf{dk}_i := \mathsf{dk}\ \forall i \in [n]$.

Definition 3 (Somewhere Extractable Commitment, adapted from [CJJ22, WW22]). *A somewhere extractable commitment scheme* SEC *with local verification is a tuple of algorithms* $\mathsf{SEC} = (\mathsf{Setup}, \mathsf{Com}, \mathsf{Open}, \mathsf{Ver})$ *defined as follows.*

$\mathsf{Setup}(1^\lambda, 1^n, B) \to \mathsf{dk}$: *On input the security parameter* λ*, the input length* n*, and the block size* B*, outputs a commitment key* dk*.*

$\mathsf{Com}(\mathsf{dk}, \boldsymbol{x}) \to (\mathsf{c}, \mathsf{aux})$: *On input the commitment key* dk*, and a vector* $\boldsymbol{x} \in \mathcal{M}^{nB}$*, outputs a commitment* c *and auxiliary input* aux*.*

$\mathsf{Open}(\mathsf{dk}, \mathsf{aux}, i) \to \pi_i$: *On input the commitment key* dk*, the auxiliary input* aux*, and an index* i*, outputs a local opening* π_i*.*

$\mathsf{Ver}(\mathsf{dk}_i, \mathsf{c}, i, x_i, \pi_i) \to b$: *On input the (local) verification key* dk_i*, the commitment* c*, an index* $i \in [n]$*, an input* $x_i \in \mathcal{M}^B$*, and a proof* π_i*, outputs a bit* $b \in \{0, 1\}$*.*

In addition, SEC *must include the following trapdoor-extraction algorithms:*

$\mathsf{TdSetup}(1^\lambda, 1^n, B, i^*) \to (\mathsf{dk}, \mathsf{td})$ *works as the setup algorithm, and additionally outputs a trapdoor* td *associated to index* i^**.*

$\mathsf{Ext}(\mathsf{td}, \mathsf{c}, i^*) \to x_{i^*}$ *On input a trapdoor* td*, a commitment* c *and an index* i^**, extracts an input* x_{i^*}*.*

Moreover, the algorithms must satisfy the following properties:

Correctness. *For any* $\lambda \in \mathbb{N}$*, any integers* n, B*, index* $i \in [n]$*, and admissible inputs* $\boldsymbol{x} \in \mathcal{M}^{nB}$*,*

$$\Pr\left[\mathsf{Ver}(\mathsf{dk}_i, \mathsf{c}, i, x_i, \pi_i) = 1 \;\middle|\; \begin{array}{l} \mathsf{dk} \leftarrow \mathsf{Setup}(1^\lambda, 1^n, B, i) \\ (\mathsf{c}, \mathsf{aux}) \leftarrow \mathsf{Com}(\mathsf{dk}, \boldsymbol{x}) \\ \pi_i \leftarrow \mathsf{Open}(\mathsf{dk}, \mathsf{aux}, i) \end{array}\right] = 1$$

Succinct local verification. *For any admissible set of parameters, there exists a function* $s_{\mathsf{SEC}}(\lambda, n, B) = \mathsf{poly}(\lambda, B) \cdot o(n)$ *such that the following properties hold:*

- *Succinct local verification keys:* $|\mathsf{dk}_i| \leq s_{\mathsf{SEC}}(\lambda, n, B)$*.*
- *Succinct commitments:* $|\mathsf{c}| \leq s_{\mathsf{SEC}}(\lambda, n, B)$*.*
- *Succinct local openings:* $|\pi_i| \leq s_{\mathsf{SEC}}(\lambda, n, B)$*.*
- *Fast local verification:* $\mathsf{Ver}(\mathsf{dk}_i, \mathsf{c}, i, x_i, \pi_i)$ *runs in time* $\leq s_{\mathsf{SEC}}(\lambda, n, B)$*.*

Setup indistinguishability. *For any PPT adversary* $\mathcal{A}$*, and any integers* n, B*,*

$$\Pr\left[\mathcal{A}(\mathsf{dk}) = 1 \;\middle|\; \begin{array}{l} i^* \leftarrow \mathcal{A}(1^\lambda, 1^n, B) \\ (\mathsf{dk}, \mathsf{td}) \leftarrow \mathsf{TdSetup}(1^\lambda, 1^n, B, i^*) \end{array}\right] - \Pr\left[\mathcal{A}(\mathsf{dk}) = 1 \;\middle|\; \begin{array}{l} i^* \leftarrow \mathcal{A}(1^\lambda, 1^n, B) \\ \mathsf{dk} \leftarrow \mathsf{Setup}(1^\lambda, 1^n, B) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Somewhere extractability. *For any PPT adversary* $\mathcal{A}$*, and any integers* n, B*,*

$$\Pr\left[\begin{array}{l} \mathsf{Ver}(\mathsf{dk}_{i^*}, \mathsf{c}, i^*, x_{i^*}, \pi_{i^*}) = 1 \\ \wedge\ \mathsf{Ext}(\mathsf{td}, \mathsf{c}, i^*) \neq x_{i^*} \end{array} \;\middle|\; \begin{array}{l} i^* \leftarrow \mathcal{A}(1^\lambda, 1^n, B) \\ (\mathsf{dk}, \mathsf{td}) \leftarrow \mathsf{TdSetup}(1^\lambda, 1^n, B, i^*) \\ (\mathsf{c}, x_{i^*}, \pi_{i^*}) \leftarrow \mathcal{A}(\mathsf{dk}) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

2.3 Batch Arguments for NP

A Batch Argument for NP (BARG) is a proof system for a particular subclass of NP, which is the conjunction of k NP statements corresponding to the same NP language. More precisely, given an NP language $\mathcal{L}$, the prover attests that k statements $\mathtt{x}_1, \ldots, \mathtt{x}_k$ belong to $\mathcal{L}$. In terms of efficiency, BARGs produce proofs π whose size is sublinear in the number of instances k. As opposed to SNARKs, the proof is linear in the size of the circuit $\mathcal{C}(\mathtt{x}_i, \mathtt{w}_i)$ that decides the language given an input $\mathtt{x}_i$ and an NP witness $\mathtt{w}_i$.

The usual security notion for BARGs is *somewhere soundness*. This means that we can program the BARG crs at some index i^* such that it is hard for any PPT adversary to produce a valid BARG proof when $\mathtt{x}_{i^*} \notin \mathcal{L}$. Additionally, a crs programmed at i^* should be indistiguishable from a crs programmed at any other index, or at no index.

We adapt the definition of BARGs for NP from [CJJ22, KLVW23]. We directly introduce BARGs with somewhere extractability, also known as seBARGs [KLVW23], since these will be the ones required by the constructions in this paper. In a nutshell, a somewhere extractable BARG strengthens the usual BARG somewhere soundness by allowing us to extract, with the help of a trapdoor, the witness $\mathtt{w}_{i^*}$ corresponding to the index i^* that was set at the time of programming the crs. We describe BARGs for boolean circuits $\mathcal{C} : \{0,1\}^n \to \{0,1\}$.

Definition 4 (BARG for NP [CJJ22, KLVW23]). *A somewhere extractable batch argument (BARG) for NP is a tuple of algorithms* $\mathsf{BARG} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Ver})$*:*

$\mathsf{Setup}(1^\lambda, k, 1^{|\mathcal{C}|}) \to \mathsf{crs}:$ *on input the security parameter λ, a number of instances k and a circuit size $|\mathcal{C}|$, outputs a common reference string* crs.

$\mathsf{Prove}(\mathsf{crs}, \mathcal{C}, \{(\mathtt{x}_i, \mathtt{w}_i)\}_{i\in[k]}) \to \pi :$ *on input the common reference string crs, a boolean circuit $\mathcal{C} : \{0,1\}^n \to \{0,1\}$, and a batch of input-witness pairs $\{(\mathtt{x}_i, \mathtt{w}_i)\}_{i\in[k]}$, outputs a proof π.*

$\mathsf{Ver}(\mathsf{crs}, \mathcal{C}, \{\mathtt{x}_i\}_{i\in[k]}, \pi) \to b :$ *on input the common reference string crs, a circuit $\mathcal{C}$, a batch of statements $\{\mathtt{x}_i\}_{i\in[k]}$, and a proof π, accepts ($b = 1$) or rejects ($b = 0$).*

In addition, BARG *must include the following trapdoor-extraction algorithms:*

$\mathsf{TdSetup}(1^\lambda, k, 1^{|\mathcal{C}|}, i^*) \to (\mathsf{crs}, \mathsf{td})$ *works as* Setup*, and additionally outputs a trapdoor* td *associated to the index i^* that is given as input.*

$\mathsf{Ext}(\mathsf{td}, \mathcal{C}, \{\mathtt{x}_i\}_{i\in[k]}, \pi) \to \mathtt{w}^*$ *On input a trapdoor td, a circuit $\mathcal{C}$, a batch of statements $\{\mathtt{x}_i\}_{i\in[k]}$ and a proof π, outputs a witness $\mathtt{w}_{i^*}$ corresponding to the position specified by the trapdoor.*

Moreover, the algorithms must satisfy the following properties:

Completeness. *For any $\lambda, k \in \mathbb{N}$, any boolean circuit $\mathcal{C} : \{0,1\}^n \to \{0,1\}$, all statement-witness pairs $(\mathtt{x}_i, \mathtt{w}_i)_{i\in[k]}$ such that $\mathcal{C}(\mathtt{x}_i, \mathtt{w}_i) = 1$ for all $i \in [k]$,*

$$\Pr\left[\mathsf{Ver}(\mathsf{crs}, \mathcal{C}, \{\mathtt{x}_i\}_{i\in[k]}, \pi) = 1 \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, k, 1^{|\mathcal{C}|}) \\ \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathcal{C}, \{(\mathtt{x}_i, \mathtt{w}_i)\}_{i\in[k]}) \end{array}\right] = 1$$

Succinctness. *For any admissible set of parameters as before, there exists a function $s_{\mathsf{BARG}}(\lambda, k, |\mathcal{C}|) = \mathsf{poly}(\lambda, \log k, |\mathcal{C}|)$ such that $|\pi| \leq s_{\mathsf{BARG}}(\lambda, k, |\mathcal{C}|)$. Besides, if $|\mathsf{crs}| \leq s_{\mathsf{BARG}}(\lambda, k, |\mathcal{C}|)$ then we say that* BARG *is crs-succinct.*

Setup indistinguishability. *For any PPT adversary $\mathcal{A}$, any $k, |\mathcal{C}| = \mathsf{poly}(\lambda)$, and index $i^* \in [k]$,*

$$\Pr\left[\mathcal{A}(\mathsf{crs}) = 1 \,\middle|\, (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{TdSetup}(1^\lambda, k, 1^{|\mathcal{C}|}, i^*)\right] - \Pr\left[\mathcal{A}(\mathsf{crs}) = 1 \,\middle|\, \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, k, 1^{|\mathcal{C}|})\right] \leq \mathsf{negl}(\lambda).$$

Somewhere argument of knowledge. *For any PPT adversary $\mathcal{A}$, any $k, |\mathcal{C}| = \mathsf{poly}(\lambda)$, and index $i^* \in [k]$,*

$$\Pr\left[\begin{array}{l}\mathsf{Ver}(\mathsf{crs}, \mathcal{C}, \{\mathtt{x}_i\}_{i\in[k]}, \pi) = 1 \\ \wedge\ \mathcal{C}(\mathtt{x}_{i^*}, \mathtt{w}) \neq 1\end{array} \,\middle|\, \begin{array}{l}(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{TdSetup}(1^\lambda, k, 1^{|\mathcal{C}|}, i^*) \\ (\mathcal{C}, \{\mathtt{x}_i\}_{i\in[k]}, \pi) \leftarrow \mathcal{A}(\mathsf{crs}) \\ \mathtt{w}^* \leftarrow \mathsf{Ext}(\mathsf{td}, \{\mathtt{x}_i\}_{i\in[k]}, \mathcal{C}, \pi)\end{array}\right] \leq \mathsf{negl}(\lambda)$$

Definition 5 (Efficient Verification). *A BARG for NP has efficient amortized verification (also called split verification in [WW22]) if there exists a pair of algorithms:*

$\mathsf{PreVer}(\mathsf{crs}, \{\mathtt{x}_i\}_{i\in[k]}) \to \mathsf{vk}$: *On input the common reference string* crs *and k instances $\{\mathtt{x}_i\}_{i\in[k]}$, it creates a succinct verification key* vk *such that $|\mathsf{vk}| \leq \mathsf{poly}(\lambda, \log k, |x_i|)$*

$\mathsf{EffVer}(\mathsf{vk}, \mathcal{C}, \pi) \to b$: *On input the succinct verification key* vk*, the circuit $\mathcal{C}$ and a proof π, accepts ($b = 1$) or rejects ($b = 0$).*

Furthermore, $\mathsf{EffVer}(\mathsf{vk}, \mathcal{C}, \pi)$ *runs in time bounded by* $\mathsf{poly}(\lambda, |\mathsf{vk}|, |\mathcal{C}|, |\pi|) = \mathsf{poly}(\lambda, \log k, |\mathcal{C}|)$*.*

2.4 Functional Commitments

A Functional Commitment (FC) [LRY16] is a powerful primitive that allows an entity to first commit to some input $\boldsymbol{x} \in \mathcal{M}^n$ and then open the commitment to $f(\boldsymbol{x})$ for some admissible function $f \in \mathcal{F}$. Both the commitment c and the opening proof π are succinct. In the description below, we follow the syntax from [BCFL23].

Definition 6 (Functional Commitments). *Let $\mathcal{M}$ be some domain, $n = \mathsf{poly}(\lambda)$ and let $\mathcal{F} \subseteq \{f : \mathcal{M}^n \to \mathcal{M}^m\}$ be a family of functions representable as arithmetic or boolean circuits over $\mathcal{M}$ for any integer $m = \mathsf{poly}(\lambda)$. A functional commitment scheme for $\mathcal{F}$ is a tuple of algorithms* $\mathsf{FC} = (\mathsf{Setup}, \mathsf{Com}, \mathsf{Open}, \mathsf{Ver})$ *that works as follows:*

$\mathsf{Setup}(1^\lambda, 1^n) \to \mathsf{ck}$ *on input the security parameter λ and the vector length n, outputs a commitment key* ck.

$\mathsf{Com}(\mathsf{ck}, \boldsymbol{x}; r) \to (\mathsf{c}, \mathsf{aux})$ *on input the commitment key* ck*, a vector $\boldsymbol{x} \in \mathcal{M}^n$ and (possibly) randomness r, outputs a commitment* c *and related auxiliary information* aux.

$\mathsf{Open}(\mathsf{ck}, \mathsf{aux}, f) \to \pi$ *on input the commitment key* ck*, auxiliary information* aux*, and a function $f \in \mathcal{F}$, returns an opening proof π.*

$\mathsf{Ver}(\mathsf{ck}, \mathsf{c}, \boldsymbol{y}, f, \pi) \to b \in \{0, 1\}$ *on input the commitment key* ck*, a commitment* c*, an output $\boldsymbol{y} \in \mathcal{M}^m$, an opening proof π, and a function $f \in \mathcal{F}$, accepts ($b = 1$) or rejects ($b = 0$).*

Moreover, the algorithms must satisfy:

Correctness. FC *is correct if for any $n \in \mathbb{N}$, all* $\mathsf{ck} \leftarrow\$\ \mathsf{Setup}(1^\lambda, 1^n)$*, any $f : \mathcal{M}^n \to \mathcal{M}^m$ in the class $\mathcal{F}$, and any $\boldsymbol{x} \in \mathcal{M}^n$, if* $(\mathsf{c}, \mathsf{aux}) \leftarrow \mathsf{Com}(\mathsf{ck}, \boldsymbol{x})$*, then*

$$\Pr[\mathsf{Ver}(\mathsf{ck}, \mathsf{c}, f, f(\boldsymbol{x}), \mathsf{Open}(\mathsf{ck}, \mathsf{aux}, f)) = 1] = 1.$$

Succinctness. *For any set of admissible parameters, there exists a function* $s_{\mathsf{FC}}(\lambda, n, m, |f|) = \mathsf{poly}(\lambda, \log n, \log m, o(|f|))$ *such that* $|\pi| \leq s_{\mathsf{FC}}(\lambda, n, m, |f|)$ *and* $|\mathsf{c}|, \leq s_{\mathsf{FC}}(\lambda, n, 1, 1)$*.*

Evaluation Binding. *For any PPT adversary* $\mathcal{A}$*, the following probability is* $\mathsf{negl}(\lambda)$*:*

$$\Pr\left[\begin{matrix}\mathsf{Ver}(\mathsf{ck}, \mathsf{c}, f, \boldsymbol{y}, \pi) = 1 \\ \wedge\ \boldsymbol{y} \neq \boldsymbol{y}' \wedge \\ \mathsf{Ver}(\mathsf{ck}, \mathsf{c}, f, \boldsymbol{y}', \pi') = 1\end{matrix} : \begin{matrix}\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, 1^n) \\ (\mathsf{c}, f, \boldsymbol{y}, \pi, \boldsymbol{y}', \pi') \leftarrow \mathcal{A}(\mathsf{ck})\end{matrix}\right]$$

Succinctness is defined with respect to both the input length n and the output length m – which we name input-succinctness and output-succinctness. Some FC constructions in the literature, however, are not output-succinct. To address this, we introduce a result that allows one to obtain output-succinctness from any FC.

Theorem 1. *Let* FC *be an evaluation binding FC for* n*-to-*1 *functions in the class* $\mathcal{F}$*. Let* $\mathcal{F}'$ *be the class of functions where each* $f : \mathcal{M}^n \to \mathcal{M}^m$ *in* $\mathcal{F}'$ *is such that each of its* m *projections is a function in* $\mathcal{F}$*. Let* $H : \mathcal{M}^m \to \mathcal{M}^\ell$ *with* $\ell = \mathsf{poly}(\lambda)$ *be a collision resistant hash function. Then, for a suitably expressive* $\mathcal{F}$*, there exists an evaluation binding* FC' *for the class* $\mathcal{F}'$*.*

We prove the theorem and describe the transformation in Appendix A.1. Next, we introduce efficient amortized verification for functional commitments.

Definition 7 (Efficient Verification). *A functional commitment admits efficient verification if there exists a pair of algorithms:*

$\mathsf{Digest}(\mathsf{ck}, f) \to d_f$ *on input the commitment key* ck *and a function* $f \in \mathcal{F}$*, outputs a digest of the function* d_f*.*

$\mathsf{EffVer}(\mathsf{ck}, \mathsf{c}, \boldsymbol{y}, d_f, \pi) \to b \in \{0, 1\}$ *on input the commitment key* ck*, a commitment* c*, an output* $\boldsymbol{y}$*, an opening proof* π*, and a digest* d_f *of a function* $f \in \mathcal{F}$*, accepts (*$b = 1$*) or rejects (*$b = 0$*).*

Furthermore, d_f *is succinct, i.e.* $|d_f| \leq s_{\mathsf{FC}}(\lambda, n, m, |f|)$*, and* $\mathsf{FC.EffVer}(\mathsf{ck}, \mathsf{c}, \boldsymbol{y}, d_f, \pi)$ *runs in time* $\leq s_{\mathsf{FC}}(\lambda, n, m, |f|) + \mathsf{poly}(\lambda, m)$[4]*.*

We also introduce a notion of *local updatability* that is central to the main results of this work. A FC supports local updatability if one can update a commitment c at a position $i \in [n]$ (or more generally, at a set of positions $S \subseteq [n]$) in a succinct way. Namely, the update must be verifiable in time $s_{\mathsf{FC}}(\lambda, n, 1, 1) \cdot \mathcal{O}(|S|) = \mathcal{O}(\lambda, \log n, |S|)$. Local update soundness is defined such that given an honestly generated commitment c to $\boldsymbol{x}$, and a set of updates $\{x'_i\}_{i \in S}$ that update $\boldsymbol{x}$ to $\boldsymbol{x}'$, it must be hard to forge a valid update from c to any c' such that c' does not commit to $\boldsymbol{x}'$. For this property, we enforce that commitments c are deterministic.

A consequence of the efficiency requirement is that if the size of ck is linear in n, the update verification must only process a section ck_S of ck. This is similar to what occurs for somewhere extractable commitments (see Definition 3).

Definition 8 (Local updatability). *A Functional Commitment* FC *is locally updatable if there exists a pair of algorithms* $(\mathsf{Upd}, \mathsf{VerUpd})$ *as follows:*

[4] The term $\mathsf{poly}(\lambda, m)$ appears since the EffVer algorithm needs to at least read the output $\boldsymbol{y}$, that has length m.

$\mathsf{FC.Upd}(\mathsf{ck}, \mathsf{aux}, S, \{x'_i\}_{i\in S}) \to (\mathsf{c}', \mathsf{aux}', \pi)$ *on input the commitment key* ck*, auxiliary information*[5] aux*, a set of positions* $S \subseteq [n]$*, and updates* $\{x'_i\}_{i\in S}$*, outputs an updated deterministic commitment* c'*, updated auxiliary input* aux'*, and an update proof* π*.*

$\mathsf{FC.VerUpd}(\mathsf{ck}_S, S, \mathsf{c}, \{x_i\}_{i\in S}, \mathsf{c}', \{x'_i\}_{i\in S}, \pi) \to 0/1$ *on input a section of the commitment key* ck_S*, a commitment* c*, a set of positions* $S \subseteq [n]$*, inputs* $\{x_i\}_{i\in S}$*, updates* $\{x'_i\}_{i\in S}$*, an updated deterministic commitment* c' *and an update proof* π*, accepts (outputs 1) or rejects (outputs 0).*

Let $\boldsymbol{x}' \leftarrow \mathtt{Up}(\boldsymbol{x}, \{x'_i\}_{i\in S})$ *be a function that updates* $\boldsymbol{x}$ *to* $\boldsymbol{x}'$*, i.e., outputs a vector* $\boldsymbol{x}'$ *that contains* x_i *at every coordinate* $i \notin S$*, and* x'_i *at every* $i \in S$*. Then, these algorithms must satisfy the following properties.*

Correctness. *For any* $n \in \mathbb{N}$*, any* $f : \mathcal{M}^n \to \mathcal{M}^m$ *in the class* $\mathcal{F}$*, any* $\boldsymbol{x} \in \mathcal{M}^n$*, any set* $S \subseteq [n]$*, and any set* $\{x'_i\}_{i\in S}$ *such that* $x'_i \in \mathcal{M}\ \forall i \in S$*, we have:*

$$\Pr\left[\begin{array}{l|r} \mathsf{VerUpd}(\mathsf{ck}_S, S, \mathsf{c}, \{x_i\}, \mathsf{c}', \{x'_i\}, \pi) = 1 & \mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, 1^n) \\ \wedge\ (\mathsf{c}', \mathsf{aux}') = \mathsf{Com}(\mathsf{ck}, \boldsymbol{x}') & (\mathsf{c}, \mathsf{aux}) \leftarrow \mathsf{Com}(\mathsf{ck}, \boldsymbol{x}) \\ & \boldsymbol{x}' \leftarrow \mathtt{Up}(\boldsymbol{x}, \{x'_i\}_{i\in S}) \\ & (\mathsf{c}', \mathsf{aux}', \pi) \leftarrow \\ & \mathsf{FC.Upd}(\mathsf{ck}, \mathsf{aux}, S, \{x'_i\}) \end{array}\right] = 1.$$

Soundness. *For any PPT adversary* $\mathcal{A}$*, the following probability is* $\mathsf{negl}(\lambda)$*:*

$$\Pr\left[\begin{array}{l|r} \mathsf{VerUpd}(\mathsf{ck}_S, S, \mathsf{c}, \{x_i\}, \mathsf{c}', \{x'_i\}, \pi) = 1 & \mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, 1^n) \\ \wedge\ \mathsf{c}' \neq \mathsf{Com}(\mathsf{ck}, \boldsymbol{x}') & (\mathsf{c}, \mathsf{aux}) \leftarrow \mathsf{Com}(\mathsf{ck}, \boldsymbol{x}) \\ & (S, \{x'_i\}, \mathsf{c}', \pi) \leftarrow \mathcal{A}(\mathsf{ck}, \mathsf{c}, \mathsf{aux}) \\ & \boldsymbol{x}' \leftarrow \mathtt{Up}(\boldsymbol{x}, \{x'_i\}_{i\in S}) \end{array}\right].$$

Succinctness. *For any admissible parameters, the update proof* $|\pi| \leq s_{\mathsf{FC}}(\lambda, n, 1, 1) \cdot \mathcal{O}(|S|)$*. Besides,* $\mathsf{FC.VerUpd}$ *runs in time bounded by* $s_{\mathsf{FC}}(\lambda, n, 1, 1) \cdot \mathcal{O}(|S|)$*.*

Additive Homomorphism and Updatability. Most previous FC constructions in the literature do not explicitly state a local updatability property, even though many present it naturally. One such way to achieve local updatability, such as in the FC schemes in [CFT22, BCFL23], is via additive homomorphism. In short, an FC supports additive homomorphism if, given commitments $\mathsf{c}_1, \ldots, \mathsf{c}_m$ to $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$, there exists an efficient addition algorithm that produces a commitment c to $\sum_{i=1}^m \boldsymbol{x}_i$. For further details, we refer to ([CFT22], Appendix A.4).

Chainable Functional Commitments. Some functional commitment schemes may offer useful composability properties such as *chainability*. A chainable functional commitment (CFC) [BCFL23] is an extension of FCs that allows some party to commit to multiple inputs $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ and then open to *a commitment of* $\boldsymbol{y} = f(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m)$, i.e, the output $\boldsymbol{y}$ remains in committed form. We refer to [BCFL23] for the exact syntax, which is a straightforward generalization of Definition 6.

Note that a CFC generically implies an FC by simply adding a proof that the output commitment indeed opens to $\boldsymbol{y}$.

[5] We note that in some algebraic schemes, only the section of aux corresponding to the set S may be needed.

3 Multi-Key Homomorphic Signatures

In this section, we recall the definition of Multi-Key Homomorphic Signatures (MKHS) [FMNP16].

As explained in the introduction, a MKHS allows each signer to sign a set of messages $\{m_{\mathsf{id},i}\}$ so that an evaluator can compute a function f on messages signed by different users and to produce a signature that certifies the correctness of the result. Since the verifier does not see the original inputs one must carefully define what does it mean that a value y is the correct output of a function f on *some* signed messages. Following the work of Gennaro and Wichs [GW13] on (single-key) homomorphic authenticators, even in the multi-key setting one can use the notion of *labeled programs*. Informally speaking, this means that a user id signs each message $m_{\mathsf{id},i}$ along with a "tag" τ_i and, in the MKHS case, her identity id. The pair $\ell_i = (\mathsf{id}, \tau_i)$ is called the "label" and is a unique identifier of the signed message. To verify an output y, one checks the signature not only w.r.t. the function f but also with the labels (ℓ_i) of its inputs—what is called a labeled program $\mathcal{P}$. This way, a successful verification of the tuple $(\mathcal{P} = (f, \ell_1, \ldots, \ell_n), y, \sigma_{f,y})$ means that y is the correct output of f on some messages signed by the corresponding user with label $\ell_1, \ldots, \ell_n$ respectively.

Definition 9 (Labeled Programs for MKHS [FMNP16]). *A labeled program $\mathcal{P}$ is a tuple $(f, \ell_1, \ldots, \ell_n)$ such that $f : \mathcal{M}^n \to \mathcal{M}^m$ is a function of n variables (e.g., a circuit) and $\ell_i \in \mathcal{L}$ is a label for the i-th input of f. Let $f_{id} : \mathcal{M} \to \mathcal{M}$ be the identity function and $\ell \in \mathcal{L}$ be any label. We denote by $\mathcal{I}_\ell = (f_{id}, \ell)$ the identity program with label ℓ. Labeled programs can be composed as follows: given $\mathcal{P}_1, \ldots, \mathcal{P}_k$ and a function $g : \mathcal{M}^t \to \mathcal{M}^m$, the composed program, denoted $\mathcal{P}^* = g(\mathcal{P}_1, \ldots, \mathcal{P}_k)$, is the one obtained by evaluating g on the collection of t outputs of $\mathcal{P}_1, \ldots, \mathcal{P}_k$. The labeled inputs of $\mathcal{P}^*$ are the distinct labeled inputs of $\mathcal{P}_1, \ldots \mathcal{P}_k$, where inputs with the same label are converted to a single input. A program $\mathcal{P} = (f, \ell_1, \ldots, \ell_n)$ can be expressed as the composition of n identity programs, i.e., $\mathcal{P} = f(\mathcal{I}_{\ell_1}, \ldots, \mathcal{I}_{\ell_n})$.*

In MKHS, each label ℓ is a pair (id, τ) where $\mathsf{id} \in \mathcal{ID}$ is a user's identity and $\tau \in \mathcal{T}$ is a tag; thus the label space is $\mathcal{L} = \mathcal{ID} \times \mathcal{T}$. We denote $\mathsf{id} \in \mathcal{P}$ if there is at least one label of the program $\mathcal{P}$ with identity id, i.e., for $\mathcal{P} = (f, \ell_1, \ldots, \ell_n)$, $\mathsf{id} \in \mathcal{P}$ iff there exists $\ell_i = (\mathsf{id}_i, \tau_i)$ such that $\mathsf{id}_i = \mathsf{id}$.

Definition 10 (Multi-Key Homomorphic Signature). *Let $\mathcal{F}$ be a family of functions, $\mathcal{ID}$ an identity space, and $\mathcal{T}$ a tag space. A Multi-Key Homomorphic Signature scheme for a family of functions $\mathcal{F}$, identity space $\mathcal{ID}$, and tag space $\mathcal{T}$ is a tuple of algorithms* $\mathsf{MKHS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Eval}, \mathsf{Ver})$ *such that:*

$\mathsf{Setup}(1^\lambda, \mathcal{F}, \mathcal{ID}, \mathcal{T}) \to \mathsf{pp}$**:** *On input the security parameter λ and descriptions of $\mathcal{F}, \mathcal{ID}, \mathcal{T}$, the setup algorithm outputs public parameters pp. We assume pp to be an input of all subsequent algorithms, even if not specified.*

$\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{sk}, \mathsf{vk})$**:** *On input the public parameters pp, the key generation algorithm outputs a* secret *signing key sk and a* public *verification key vk.*

$\mathsf{Sign}(\mathsf{sk}, m, \ell) \to \sigma$**:** *On input a signing key sk, a label $\ell = (\mathsf{id}, \tau) \in \mathcal{L}$, and a message $m \in \mathcal{M}$, the signing algorithm outputs a signature σ.*

$\mathsf{Eval}(f, (\mathcal{P}_i, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}_i}, m_i, \sigma_i)_{i \in [n]}) \to \sigma_{f,y}$**:** *Given a function $f \in \mathcal{F}$ with n inputs, and for each input i a triple consisting of a labeled program $\mathcal{P}_i$, the set of corresponding verification keys $\{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}_i}$, a message m_i and a signature σ_i, the evaluation algorithm outputs a new signature $\sigma_{f,y}$.*

$\mathsf{Ver}(\mathcal{P}, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}}, \boldsymbol{y}, \sigma_{f,y}) \to b$**:** *On input a labeled program $\mathcal{P} = (f, \ell_1, \ldots, \ell_n)$, the set of verification keys $\{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}}$ of the users involved in $\mathcal{P}$, a value $\boldsymbol{y} \in \mathcal{M}^m$, and a signature $\sigma_{f,y}$, the verification algorithm outputs 0 (reject) or 1 (accept).*

A MKHS scheme should have authentication and evaluation correctness. The former says that a freshly generated signature on (ℓ, m) verifies correctly for m as the output of the identity program $\mathcal{I}_\ell$.

Definition 11 (Authentication correctness). *For all public parameters* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{F}, \mathcal{ID}, \mathcal{T})$*, keypair* $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$*, label* $\ell \in \mathcal{L}$*, message* $m \in \mathcal{M}$*, and identity program* $\mathcal{I}_\ell$*, if* $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m, \ell)$ *then* $\mathsf{Ver}(\mathcal{I}_\ell, \mathsf{vk}, m, \sigma) = 1$ *holds with overwhelming probability.*

Evaluation correctness instead says, roughly, that running Eval with a function f on a tuple of valid signatures produces a new valid signature for the output. We consider two classes of MKHS schemes: *single-hop* and *multi-hop*. Single-hop MKHS are schemes where Eval can only be executed on signatures produced by Sign. In this case, evaluation correctness ensures that, given a function f and signatures $(\sigma_1, \ldots, \sigma_n)$ such that each σ_i verifies for m_i as the output of $\mathcal{I}_{\ell_i}$, Eval produces a signature that verifies for $\boldsymbol{y} = f(m_1, \ldots, m_n)$ as the output of the labeled program $\mathcal{P} = (f, \ell_1, \ldots, \ell_n)$.

Definition 12 (Single-Hop Evaluation correctness). *Consider any public parameters* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{F}, \mathcal{ID}, \mathcal{T})$*, any set* $\{(\mathsf{vk}_i, \sigma_i, m_i, \ell_i)\}_{i \in [n]}$ *such that, for every* $i \in [n]$*,* vk_i *is honestly generated and* $\mathsf{Ver}(\mathcal{I}_{\ell_i}, \mathsf{vk}_i, m_i, \sigma_i) = 1$*, and any function* $f \in \mathcal{F}$*. If* $\boldsymbol{y} = f(m_1, \ldots, m_n)$*,* $\mathcal{P} = (f, \ell_1, \ldots, \ell_n)$*, and* $\sigma_{f,y} = \mathsf{Eval}(f, (\mathcal{I}_{\ell_i}, \mathsf{vk}_i, m_i, \sigma_i)_{i \in [n]})$ *then* $\mathsf{Ver}(\mathcal{P}, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}}, \boldsymbol{y}, \sigma_{f,y}) = 1$ *with overwhelming probability.*

Multi-hop MKHS instead allow to execute Eval on signatures produced by previous executions of Eval. In this case, evaluation correctness ensures that, given a function f and triples $(\sigma_1, \ldots, \sigma_n)$ such that each σ_i verifies for m_i as the output of $\mathcal{I}_{\ell_i}$, Eval produces a signature that verifies for $\boldsymbol{y} = f(m_1, \ldots, m_n)$ as the output of the labeled program $\mathcal{P} = (f, \ell_1, \ldots, \ell_n)$.

Definition 13 (Multi-Hop Evaluation correctness). *Consider any public parameters* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{F}, \mathcal{ID}, \mathcal{T})$*, any* $(\mathcal{P}_i, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}_i}, m_i, \sigma_i)_{i \in [n]}$ *such that all the verification keys are honestly generated and, for every* $i \in [n]$*,* $\mathsf{Ver}((\mathcal{P}_i, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}_i}, m_i, \sigma_i)_{i \in [n]}) = 1$*, and any function* $f \in \mathcal{F}$*. If* $\boldsymbol{y} = f(m_1, \ldots, m_n)$*,* $\mathcal{P} = f(\mathcal{P}_1, \ldots, \mathcal{P}_n)$*, and* $\sigma_{f,y} = \mathsf{Eval}(f, (\mathcal{P}_i, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}_i}, m_i, \sigma_i)_{i \in [n]})$ *then with overwhelming probability* $\mathsf{Ver}(\mathcal{P}, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}}, \boldsymbol{y}, \sigma_{f,y}) = 1$*.*

Next, we define *succinctness*, which is the property that makes MKHS a nontrivial primitive to realize. Intuitively, a MKHS is succinct if the size of the signatures generated by Eval is much shorter than the input size of the evaluated function, e.g., polylogarithmic.

Definition 14 (Succinctness). *Let* $s_{\mathsf{MKHS}} : \mathbb{N}^4 \to \mathbb{N}$ *be a function. A MKHS scheme* MKHS *for a class of functions* $\mathcal{F}$ *is* s_{MKHS}*-succinct if for every honestly generated parameters* pp*, keys and signatures, and any function* $f : \mathcal{M}^n \to \mathcal{M}^m$*,* $f \in \mathcal{F}$*, the output* $\sigma_{f,y}$ *of* $\mathsf{Eval}(f, \cdot)$ *is of size* $|\sigma_{f,y}| \leq s_{\mathsf{MKHS}}(\lambda, n, m, |f|)$*. Additionally, we say that* MKHS *is* succinct *if there exists a fixed function* $s_{\mathsf{MKHS}}(\lambda, n, m, |f|) = \mathsf{poly}(\lambda, \log n, \log m, o(|f|))$*.*

We note that our succinctness definition is stronger than the one originally proposed in [FMNP16] which allowed signatures to grow linearly (or polynomially) in the number t of distinct users involved in the computation, but still logarithmically in the total number of inputs.

3.1 Security

The security notion of multi-key homomorphic signatures intuitively models the fact that an adversary, who can query signatures on messages of its choice to multiple users, can only produce valid signatures that are either the ones it received, or ones that are obtained by correctly executing the evaluation algorithm on genuine signatures. The adversary may also corrupt users to obtain their secret keys, yet the alleged forgery cannot involve verification keys of corrupted users.

Definition 15 (Unforgeability). *Consider the security experiment denoted by* $\mathsf{HomUF\text{-}CMA}_{\mathcal{A},\mathsf{MKHS}}(1^\lambda)$ *in Figure 3 between an adversary* $\mathcal{A}$ *and a challenger* $\mathcal{CH}$. *A* MKHS *scheme is unforgeable (*HomUF-CMA-*secure) if, for all PPT adversaries* $\mathcal{A}$, *we have* $\Pr[\mathsf{HomUF\text{-}CMA}_{\mathcal{A},\mathsf{MKHS}}(1^\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Game $\mathsf{HomUF\text{-}CMA}_{\mathcal{A},\mathsf{MKHS}}(1^\lambda)$:

Setup: The challenger $\mathcal{CH}$ proceeds as follows:
- Initialize empty lists $\mathsf{L}_{\mathsf{ID}}, \mathsf{L}_{\mathsf{Corr}}, \mathsf{L}_{\mathsf{Sig}} \leftarrow \emptyset$ and generate $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{F}, \mathcal{ID}, \mathcal{T})$.
- Run $\mathcal{A}(\mathsf{pp})$. Next, $\mathcal{A}$ can make the following queries adaptively.

KeyGen queries $\mathcal{O}^{\mathsf{KeyGen}}(\mathsf{id})$**:** If $\mathsf{id} \notin \mathsf{L}_{\mathsf{ID}}$, generate $(\mathsf{vk}_{\mathsf{id}}, \mathsf{sk}_{\mathsf{id}}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, update $\mathsf{L}_{\mathsf{ID}} = \mathsf{L}_{\mathsf{ID}} \cup \{\mathsf{id}\}$, and return $\mathsf{vk}_{\mathsf{id}}$ to $\mathcal{A}$.

Signing queries $\mathcal{O}^{\mathsf{Sign}}(\ell, m)$**:** Given $\ell = (\mathsf{id}, \tau)$:
- If $(\ell, \cdot) \notin \mathsf{L}_{\mathsf{Sig}}$, compute $\sigma_\ell \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{id}}, \ell, m)$, update $\mathsf{L}_{\mathsf{Sig}} := \mathsf{L}_{\mathsf{Sig}} \cup (\ell, m)$, and return σ_ℓ to $\mathcal{A}$.
- Else, if $(\ell, \cdot) \in \mathsf{L}_{\mathsf{Sig}}$, ignore the query.

Corruption queries $\mathcal{O}^{\mathsf{Corr}}(\mathsf{id})$**:** if $\mathsf{id} \in \mathsf{L}_{\mathsf{ID}}$ and $\mathsf{id} \notin \mathsf{L}_{\mathsf{Corr}}$, update $\mathsf{L}_{\mathsf{Corr}} \leftarrow \mathsf{L}_{\mathsf{Corr}} \cup \mathsf{id}$, and return $\mathsf{sk}_{\mathsf{id}}$

Forgery: At the end of the game, $\mathcal{A}$ returns a tuple $(\mathcal{P}^*, \boldsymbol{y}^*, \sigma^*)$ where $\mathcal{P}^* = (f^*, \ell_1^*, \ldots, \ell_n^*)$.

Game output: Return 1 if and only if $\mathsf{Ver}(\mathcal{P}^*, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}^*}, \boldsymbol{y}^*, \sigma^*) = 1$, $\{\mathsf{id} \in \mathcal{P}^*\} \cap \mathsf{L}_{\mathsf{Corr}} = \emptyset$, and one of the following cases occurs:
- Type 1: $\exists j \in [n]$ such that $(\ell_j^*, \cdot) \notin \mathsf{L}_{\mathsf{Sig}}$ (i.e., $\mathcal{A}$ never made a query with label ℓ_j^*).
- Type 2: $\forall i \in [n] : (\ell_i^*, m_i) \in \mathsf{L}_{\mathsf{Sig}}$ but $\boldsymbol{y}^* \neq f^*(m_1, \ldots, m_n)$

Fig. 3: Security experiment $\mathsf{HomUF\text{-}CMA}_{\mathcal{A},\mathsf{MKHS}}(1^\lambda)$.

The above notion of security, introduced by Fiore et al. [FMNP16], is *adaptive* insofar as the adversary can make corruption queries at any point in the game. This notion is stronger than the non-adaptive security achieved by the construction in [FMNP16], where the adversary can perform corruption queries only on identities for which no signature query had already been performed.

3.2 Amortized efficiency

We give a definition of amortized efficiency for MKHS schemes. The issue is that in the basic syntax of MKHS (and HS too) the verifier should read the description of the program $\mathcal{P}$ which may take the same running time as the computation to be verified, especially in a model of computation such as circuits. To address this, we consider the case in which one can preprocess the labeled program $\mathcal{P}$, independently of the signature to be verified, and to reuse it. However, we observe that preprocessing the entire tuple $\mathcal{P} = (f, \ell_1, \ldots, \ell_n)$ would not give any benefit because in MKHS labels are unique, and thus preprocessing a function f for the evaluation on a set of labels $\ell_1, \ldots, \ell_n$ cannot be reusable. Therefore we model preprocessing via two algorithms: one for the function f

and one for the input labels $\ell_1, \ldots, \ell_n$ and verification keys $\{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}$, which benefits when running the same function f on different set of signed inputs or when executing different functions on the same set of signed inputs.

Definition 16 (Amortized efficiency). *An MKHS scheme satisfies amortized efficiency if there is a triple of algorithms* $(\mathsf{PrepFunc}, \mathsf{PrepLabels}, \mathsf{EffVer})$ *such that:*

- *For any labeled program* $\mathcal{P} = (f, \ell_1, \ldots, \ell_n)$*, verification keys* $\{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}$*, output* $\boldsymbol{y}$ *and signature* $\sigma_{f,y}$ *such that* $\mathsf{Ver}(\mathcal{P} = (f, \ell_1, \ldots, \ell_n), \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}, \boldsymbol{y}, \sigma_{f,y}) = 1$ *it holds that:*
$$\mathsf{EffVer}(\mathsf{PrepLabels}(\mathsf{pp}, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}, (\ell_1, \ldots, \ell_n)), \mathsf{PrepFunc}(\mathsf{pp}, f), \boldsymbol{y}, \sigma_{f,y}) = 1$$
- *Given* $\mathsf{vk}_{\boldsymbol{\ell}} \leftarrow \mathsf{PrepLabels}(\mathsf{pp}, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}, (\ell_1, \ldots, \ell_n))$ *and* $d_f \leftarrow \mathsf{PrepFunc}(\mathsf{pp}, f)$*, the running time of* $\mathsf{EffVer}(\mathsf{vk}_{\boldsymbol{\ell}}, d_f, \boldsymbol{y}, \sigma_{f,y})$ *is bounded by* $s_{\mathsf{MKHS}}(\lambda, n, m, |f|) \cdot m = \mathsf{poly}(\lambda, \log n, m, \log |f|)$*.*

Finally, we note that previous work on (single-key) homomorphic signatures [CFW14, GVW15] used a different preprocessing approach based on assuming that labels have a structure $\ell = (\Delta, \tau)$ consisting of a dataset identifier Δ (e.g., a filename) and a tag.[6] Then they allow preprocessing the circuit along with tags in order to reuse it to verify computations on different datasets. In comparison, our preprocessing notion is more flexible and, by allowing arbitrary labels, implies the one from previous work.

3.3 Context Hiding

Informally speaking, a MKHS is context-hiding if signatures on outputs do not reveal information on the inputs of the function. In our work, we adapt to the multi-key setting the context-hiding definition for HS of [CFN15, full version], which in turn generalizes the one in [GVW15].

Definition 17 (Context-Hiding MKHS). *A MKHS supports context-hiding if there exist additional PPT procedures* $\tilde{\sigma} \leftarrow \mathsf{Hide}(\mathcal{P}, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}, y, \sigma)$ *and* $\mathsf{HVer}(\mathcal{P}, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}, y, \sigma)$ *such that:*

- *Correctness: For any tuple* $(\mathcal{P}, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}, y, \sigma)$ *such that* $\{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}$ *are honestly generated and* $\mathsf{Ver}(\mathcal{P}, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}, y, \sigma) = 1$*, we have that* $\mathsf{HVer}(\mathcal{P}, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}, y, \mathsf{Hide}(\mathcal{P}, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}, y, \sigma)) = 1$*.*
- *Unforgeability: The signature scheme is secure when we replace the original verification algorihtm* Ver *with* HVer *in the security game.*
- *Context-Hiding: There is a simulator* $\mathsf{Sim} = (\mathsf{Sim}_{\mathsf{Setup}}, \mathsf{Sim}_{\mathsf{Sig}})$ *such that for any PPT (stateful) distinguisher* $\mathcal{D}$ *running in the experiments* $\{\mathsf{CtxtHiding}^b\}_{b=0,1}$ *defined below, it holds*
$$\left|\Pr[\mathsf{CtxtHiding}^0_{\mathcal{D},\mathsf{MKHS}}(\lambda) = 1] - \Pr[\mathsf{CtxtHiding}^1_{\mathcal{D},\mathsf{MKHS}}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda)$$

$\mathsf{CtxtHiding}^0_{\mathcal{D},\mathsf{MKHS}}(\lambda)$	$\mathsf{CtxtHiding}^1_{\mathcal{D}}(\lambda)$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{F}, \mathcal{ID}, \mathcal{T})$	$(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{Sim}_{\mathsf{Setup}}(1^\lambda, \mathcal{F}, \mathcal{ID}, \mathcal{T})$
$(f, (\mathcal{P}_i, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}_i}, m_i, \sigma_i)_{i\in[n]}) \leftarrow \mathcal{D}(\mathsf{pp})$	$(f, (\mathcal{P}_i, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}_i}, m_i, \sigma_i)_{i\in[n]}) \leftarrow \mathcal{D}(\mathsf{pp})$
$b \leftarrow \wedge_{i\in[n]} \mathsf{Ver}(\mathcal{P}_i, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}_i}, m_i, \sigma_i)$	$b \leftarrow \wedge_{i\in[n]} \mathsf{Ver}(\mathcal{P}_i, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}_i}, m_i, \sigma_i)$
$y \leftarrow f(m_1, \ldots, m_n)$	$y \leftarrow f(m_1, \ldots, m_n)$
$\mathcal{P} \leftarrow f(\mathcal{P}_1, \ldots, \mathcal{P}_n)$	$\mathcal{P} \leftarrow f(\mathcal{P}_1, \ldots, \mathcal{P}_n)$
$\sigma \leftarrow \mathsf{Eval}(f, (\mathcal{P}_i, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}_i}, m_i, \sigma_i)_{i\in[n]})$	
$\tilde{\sigma} \leftarrow \mathsf{Hide}(\mathcal{P}, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}, y, \sigma))$	$\tilde{\sigma} \leftarrow \mathsf{Sim}_{\mathsf{Sig}}(\mathsf{td}, \mathcal{P}, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}, y))$
$b' \leftarrow \mathcal{D}(\tilde{\sigma})$	$b' \leftarrow \mathcal{D}(\tilde{\sigma})$
return $b \wedge b'$	**return** $b \wedge b'$

[6] Though not formalized, this is the same notion used in the MKHS scheme of [FMNP16].

Generic Context-Hiding solution via NIZKs. We state a simple result showing that any MKHS with amortized verification can be compiled, via the use of a NIZK scheme, into one that has context-hiding.

Theorem 2. *Let* MKHS *be a MKHS scheme with amortized efficiency, and let* Π *be a knowledge-sound NIZK for the NP relation* $R_{\mathsf{MKHS}} = \{((\mathsf{vk}_{\boldsymbol{\ell}}, d_f, y); \sigma_{f,y}) : \mathsf{EffVer}(\mathsf{vk}_{\boldsymbol{\ell}}, d_f, y, \sigma_{f,y}) = 1\}$*. Then there exists a context-hiding MKHS scheme* MKHS^* *for the same class of functions supported by* MKHS*.*

The proof is rather straightforward and based on the idea of using the NIZK to prove the existence of a valid signature. The amortized efficiency requirement ensures that the scheme remains succinct even if the NIZK is not succinct. A proof sketch is given below.

Proof (Sketch). The algorithms of MKHS^* are the same as those of MKHS except that $\mathsf{MKHS}^*.\mathsf{Setup}$ runs $\mathsf{MKHS}.\mathsf{Setup}$ and additionally generates a common reference string for Π. Then the algorithm Hide runs Π's prover on a valid $((\mathsf{vk}_{\boldsymbol{\ell}}, d_f, y); \sigma_{f,y}) \in R_{\mathsf{MKHS}}$ and sets $\tilde{\sigma}$ as the resulting NIZK proof. In turn, HVer executes Π's verifier on $(\mathsf{vk}_{\boldsymbol{\ell}}, d_f, y)$ and $\tilde{\sigma}$. Correctness is straightforward. The succinctness of MKHS^* is based on the succinctness of MKHS and the fact that EffVer running time is $\mathsf{poly}(\lambda, \log n, \log |f|)$; therefore, even if the size of the NIZK proof depended on the size of the statement, it would be still succinct. For the unforgeability of MKHS^* we rely on the fact that Π is an argument of knowledge, which allows us to use its extractor to get a MKHS signature σ from the NIZK proof $\tilde{\sigma}$ so that from a forgery for MKHS^* we can get one for MKHS. Finally, the context-hiding property follows by the zero-knowledge property of Π.

4 Our MKHS Construction

In this section we present our main result, that is the construction of a fully succinct MKHS.

Our scheme MKHS relies on four building blocks: a functional commitment FC, a digital signature scheme Σ, a somewhere extractable BARG for NP BARG, and a somewhere extractable commitment SEC. MKHS allows the evaluation of the same functions supported by FC, and it supports arbitrary identities and tags, i.e., $\mathcal{T} = \mathcal{ID} = \{0,1\}^\lambda$. We denote messages by m_i and labels by ℓ_i and assume that $|m_i| = \mathsf{poly}(\lambda)$ for a fixed polynomial.

As described in the technical overview, the main idea of our construction is to combine a BARG proof to attest the validity of each signature-message pair, and a FC proof to show the correct evaluation of f on $m_1, \ldots, m_n$, which are committed in c. Moreover, to connect both proofs, our construction verifies the correctness of c inside the BARG circuit $\mathcal{C}$, by starting with an empty commitment, and iteratively building a commitment to $m_1, \ldots, m_n$. We remark that the FC scheme must be updatable, and also deterministic, such that we can test commitment equality.

We describe the construction in Figure 4 and summarize its main properties in Theorem 3. For ease of exposition, in this scheme we focus on single-hop evaluation and do not consider context-hiding. We show in Section 5.1 how to achieve multi-hop sequential composition by employing chainable FCs instead of FCs. Also, we recall that context-hiding can be achieved via NIZKs following Theorem 2.

Theorem 3. *Let* FC *be a deterministic and updatable functional commitment scheme for a class of functions* $\mathcal{F} : \mathcal{M}^n \to \mathcal{M}^m$*,* BARG *a somewhere-extractable batch argument for NP,* SEC *a somewhere extractable commitment, and* Σ *a* EUF-CMA*-secure signature scheme for messages in* $\mathcal{M} \times \{0,1\}^{2\lambda}$*.*

$\underline{\mathsf{MKHS.Setup}(1^\lambda, 1^n, \mathcal{F})}$:

- Calculate the required circuit size $|\mathcal{C}|$ given $n, \mathcal{F}, \lambda$.
- Calculate the required block size B from λ. Note that $B = \mathsf{poly}(\lambda)$.
- $\mathsf{crs} \leftarrow \mathsf{BARG.Setup}(1^\lambda, n, 1^{|\mathcal{C}|})$.
- $\mathsf{dk} \leftarrow \mathsf{SEC.Setup}(1^\lambda, n, B)$
- $\mathsf{ck} \leftarrow \mathsf{FC.Setup}(1^\lambda, 1^n)$.
- Output $\mathsf{pp} \leftarrow (\mathsf{crs}, \mathsf{dk}, \mathsf{ck})$.

$\underline{\mathsf{MKHS.KeyGen}(1^\lambda)}$:

- Output $(\mathsf{vk}, \mathsf{sk}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\lambda)$.

$\underline{\mathsf{MKHS.Sign}(\mathsf{sk}, \ell, m)}$:

- $\sigma \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}, m|\ell)$.
- Output σ.

$\underline{\mathsf{MKHS.Eval}(\mathsf{pp}, f, (\ell_i, \mathsf{vk}_i, m_i, \sigma_i)_{i\in[n]})}$:

- Parse $\mathsf{pp} := (\mathsf{crs}, \mathsf{ck}, \mathsf{dk})$.
- $(\mathsf{c}, \mathsf{aux}) \leftarrow \mathsf{FC.Com}(\mathsf{ck}, m_1, \ldots, m_n)$.
- $\pi_f \leftarrow \mathsf{FC.Open}(\mathsf{ck}, f, \mathsf{aux})$.
- $(\mathsf{c}_0, \mathsf{aux}_0) \leftarrow \mathsf{FC.Com}(\mathsf{ck}, \mathbf{0})$.
- For every $i \in [n]$, compute $(\mathsf{c}_i, \mathsf{aux}_i, \pi_i) \leftarrow \mathsf{FC.Upd}(\mathsf{ck}, \mathsf{aux}_{i-1}, i, m_i)$.
  Note that each c_i is a commitment to the *partial* vector $(m_1, \ldots, m_i, 0, \ldots, 0)$. Note also that $\mathsf{c}_n = \mathsf{c}$.
- Compute a somewhere extractable commitment to all partial commitments $(\mathsf{c}_w, \mathsf{aux}_w) \leftarrow \mathsf{SEC.Com}(\mathsf{dk}, (\mathsf{c}_1, \ldots, \mathsf{c}_n))$.
- For every $i \in [n]$, compute local opening proofs $o_i \leftarrow \mathsf{SEC.Open}(\mathsf{dk}, \mathsf{aux}_w, i)$ to each c_i.
- Compute a BARG proof $\pi_\sigma \leftarrow \mathsf{BARG.Prove}(\mathsf{crs}, \mathcal{C}, \{\mathtt{x}_i, \mathtt{w}_i\}_i)$ for circuit $\mathcal{C}(\mathtt{x}_i, \mathtt{w}_i)$ as described in Figure 5.
- Output $\sigma_{f,y} = (\mathsf{c}, \pi_f, \pi_\sigma, \mathsf{c}_w)$.

$\underline{\mathsf{MKHS.Ver}(\mathsf{pp}, \mathcal{P}, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}, \boldsymbol{y}, \sigma_{f,y})}$:

- Parse $\mathcal{P} := (f, \ell_1, \ldots, \ell_n)$ and $\{\ell_i := (\mathsf{id}_i, \tau_i)\}$.
- If $\mathcal{P} = (f_{id}, \ell_1)$ then check that $\Sigma.\mathsf{Ver}(\mathsf{vk}_{\mathsf{id}_1}, \boldsymbol{y}|\ell_1, \sigma_{f,y}) = 1$.
- Else, parse $\sigma_{f,y} := (\mathsf{c}, \pi_f, \pi_\sigma, \mathsf{c}_w)$.
- Let $\mathsf{c}_0 \leftarrow \mathsf{FC.Com}(\mathsf{ck}, \mathbf{0})$
- Compute the circuit $\mathcal{C}$ in Figure 5, hardcoding $\mathsf{c}, \mathsf{c}_w, \mathsf{c}_0$.
- Given $\{\mathsf{vk}_i\}_i := \{\mathsf{vk}_{\mathsf{id}_i}\}_i$ and $\{\ell_i\}_i$, $\{\mathsf{ck}_i\}_i$, $\{\mathsf{dk}_i\}_i$, define $\mathtt{x}_i = (\mathsf{vk}_i, \mathsf{ck}_i, \mathsf{dk}_i, \mathsf{dk}_{i-1}, \ell_i, i)$.
- Check that $\mathsf{FC.Ver}(\mathsf{ck}, \mathsf{c}, f, \boldsymbol{y}, \pi_f) = 1$.
- Check that $\mathsf{BARG.Ver}(\mathsf{crs}, \mathcal{C}, \{\mathtt{x}_i\}_i, \pi_\sigma) = 1$.
- Output 1 if both checks pass.

Fig. 4: Construction of a succinct multi-key homomorphic signature scheme MKHS from a functional commitment FC, a BARG for NP BARG, a somewhere extractable commitment SEC and a digital signature Σ.

Description of $\mathcal{C}(\mathtt{x}, \mathtt{w})$:
Hardwired: $\mathsf{c}, \mathsf{c}_w, \mathsf{c}_0$
Statement: $\mathtt{x} = (\mathsf{vk}_i, \mathsf{ck}_i, \mathsf{dk}_i, \mathsf{dk}_{i-1}, \ell_i, i)$
Witness: $\mathtt{w} = (m_i, \sigma_i, \pi_i, \mathsf{c}_{i-1}, \mathsf{c}_i, o_{i-1}, o_i)$
Circuit:

- If $i = 1$, check that $\mathsf{c}_{i-1} = \mathsf{c}_0$ and skip the SEC verification check for $i - 1$.
- If $i = n$, check that $\mathsf{c}_i = \mathsf{c}$
- Check that:

$$\begin{aligned} &\Sigma.\mathsf{Ver}(\mathsf{vk}_i, m_i | \ell_i, \sigma_i) = 1 \\ \wedge\ &\mathsf{FC.VerUpd}(\mathsf{ck}_i, i, \mathsf{c}_{i-1}, 0, \mathsf{c}_i, m_i, \pi_i) = 1 \\ \wedge\ &\mathsf{SEC.Ver}(\mathsf{dk}_i, \mathsf{c}_w, i, o_i, \mathsf{c}_i) = 1 \\ \wedge\ &\mathsf{SEC.Ver}(\mathsf{dk}_{i-1}, \mathsf{c}_w, i-1, o_{i-1}, \mathsf{c}_{i-1}) = 1 \end{aligned}$$

Fig. 5: Description of the BARG circuit $\mathcal{C}$.

Then, the construction MKHS *in Figure 4 is an adaptively-secure multi-key homomorphic signature for $\mathcal{F}$.*

Moreover, given that the following conditions are satisfied:

- BARG *has $s_{\mathsf{BARG}}(\lambda, |\mathcal{C}|, k)$ succinct proofs.*
- FC *has $s_{\mathsf{FC}}(\lambda, n, m, |f|)$ succinct opening proofs and commitments, and admits succinct local verification where $\mathsf{VerUpd}(\mathsf{ck}_i, i, \mathsf{c}, 0, \mathsf{c}', m_i, \pi_i)$ runs in time bounded by $s_{\mathsf{FC}}(\lambda, n, 1, 1)$.*
- SEC *admits local verification with $s_{\mathsf{SEC}}(\lambda, n, B)$ succinctness.*

Then, MKHS *has succinct signatures of size $|\sigma_{f,y}| = s_{\mathsf{MKHS}}(\lambda, n, m, |f|)$, where, for $|\mathcal{C}| = s_{\mathsf{FC}}(\lambda, n, 1, \lambda) + s_{\mathsf{SEC}}(\lambda, n, \lambda)$, we have (up to constant factors),*

$$s_{\mathsf{MKHS}}(\lambda, n, m, |f|) = s_{\mathsf{BARG}}(\lambda, |\mathcal{C}|, n) + s_{\mathsf{FC}}(\lambda, n, m, |f|) + s_{\mathsf{SEC}}(\lambda, n, \lambda).$$

Proof. Authentication correctness follows directly by the correctness of Σ. Evaluation correctness follows from the correctness of all the building blocks.

For succinctness, observe that the four additive factors in the expression for $|\sigma_{f,y}|$ correspond to the sizes of $\pi_\sigma, \pi_f, \mathsf{c}, \mathsf{c}_w$, respectively. To calculate the expression for $s_{\mathsf{MKHS}}(\lambda, n, m, |f|)$, note that the block size of the SEC is $B = \mathsf{poly}(\lambda, \log n)$. Then, note that all keys, commitments, and openings involved in $\mathcal{C}$ are of size $s_{\mathsf{FC}}(\lambda, n, 1, \lambda) + s_{\mathsf{SEC}}(\lambda, n, \lambda)$, as well as the running time of the FC.VerUpd, SEC.Ver and Σ.Ver algorithms. Hence, $|\mathcal{C}| = s_{\mathsf{FC}}(\lambda, n, 1, \lambda) + s_{\mathsf{SEC}}(\lambda, n, \lambda)$.

We prove security in Section 4.2.

4.1 Efficient Verification

If FC has amortized efficient verification, then it is possible to preprocess the function f. Similarly, if BARG has amortized efficient verification, it is possible to preprocess the labels ℓ_i and the respective verification keys. We describe the corresponding preprocessing algorithms MKHS.PrepFunc and MKHS.PrepLabels, as well as the efficient verification algorithm MKHS.EffVer, in Figure 6.

$\underline{\mathsf{MKHS.PrepFunc}(\mathsf{pp}, f)}$:
- Parse $\mathsf{pp} := (\mathsf{crs}, \mathsf{ck}, \mathsf{dk})$.
- Output $d_F \leftarrow \mathsf{FC.Digest}(\mathsf{ck}, f)$.

$\underline{\mathsf{MKHS.PrepLabels}(\mathsf{pp}, (\mathsf{vk}_i, \ell_i)_{i\in[n]})}$:
- Parse $\mathsf{pp} := (\mathsf{crs}, \mathsf{ck}, \mathsf{dk})$.
- Given $\{\mathsf{vk}_i\}_i$, $\{\ell_i\}_i$, $\{\mathsf{ck}\}_i$, $\{\mathsf{dk}\}_i$, define $\mathtt{x}_i = (\mathsf{vk}_i, \ell_i, \mathsf{ck}_i, \mathsf{dk}_i)$.
- Output $\mathsf{vk}_\ell \leftarrow \mathsf{BARG.PreVer}(\mathsf{crs}, \{\mathtt{x}_i\}_i)$.

$\underline{\mathsf{MKHS.EffVer}(\mathsf{vk}_\ell, d_F, \boldsymbol{y}, \sigma_{f,y})}$:
- Parse $\sigma_{f,y} := (\mathsf{c}, \pi_f, \pi_\sigma, \mathsf{c}_w)$.
- Let $\mathsf{c}_0 \leftarrow \mathsf{FC.Com}(\mathsf{ck}, \mathbf{0})$
- Check that $\mathsf{FC.EffVer}(\mathsf{ck}, \mathsf{c}, d_F, \boldsymbol{y}, \pi_f) = 1$
- Compute the BARG circuit $\mathcal{C}$, hardcoding $\mathsf{c}, \mathsf{c}_w, \mathsf{c}_0$.
- Check that $\mathsf{BARG.EffVer}(\mathsf{crs}, \mathcal{C}, \mathsf{vk}_\ell, \pi_\sigma) = 1$
- Output 1 iff both checks pass.

Fig. 6: Efficient verification algorithms for our construction of a multi-key homomorphic signature scheme MKHS.

We summarize the efficient verification properties in the following corollary of Theorem 3. The proof follows from the definitions of efficient verification for BARGs (Definition 5) and for FCs (Definition 7).

Corollary 1. *If* BARG *and* FC *admit efficient verification, the* MKHS *scheme from Figure 4 with the algorithms in Figure 6 also satisfies efficient verification, i.e., the running time of* $\mathsf{MKHS.EffVer}(\mathsf{vk}_\ell, d_F, \boldsymbol{y}, \sigma_{f,y})$ *is bounded by* $s_{\mathsf{MKHS}}(\lambda, n, m, |f|) \cdot m$.

We note that the efficient verification property of our scheme is flexible, in the sense that we introduce separate algorithms for preprocessing the function PrepFunc and for the labels PrepLabels. Therefore, if the BARG satisfies efficient verification but the FC does not (or vice-versa), our MKHS admits pre-processing only the labels (or the function).

4.2 Proof of Security

Let $\mathcal{A}$ be an adversary in the security experiment $\mathsf{HomUF\text{-}CMA}_{\mathcal{A},\mathsf{MKHS}}(1^\lambda)$ for our MKHS construction. In this game, the adversary has access to a signing oracle $\mathcal{O}^{\mathsf{Sign}}$ such that, for $\ell = (\mathsf{id}, \tau)$, then $\mathcal{O}^{\mathsf{Sign}}(\ell, m)$ outputs $\sigma_\ell \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}_{\mathsf{id}}, m|\ell)$. It also has access to a key generation $\mathcal{O}^{\mathsf{KeyGen}}$ and a corruption $\mathcal{O}^{\mathsf{Corr}}$ oracle. Finally, $\mathcal{A}$ produces an alleged forgery $(\mathcal{P}^*, \boldsymbol{y}^*, \sigma^*)$ where $\mathcal{P}^* = (f^*, \ell_1^*, \ldots, \ell_n^*)$. We recall that there are two possible types of forgeries, that we define formally as the following events.

- $\mathsf{TYPE}_1 := \exists j \in [n], (\ell_j^*, \cdot) \notin \mathsf{L}_{\mathsf{Sig}}$. Namely, there exists some index j such that $\mathcal{A}$ never queried $(\ell_j^*, \cdot)$ to the signing oracle.
- $\mathsf{TYPE}_2 := \forall i \in [n], (\ell_i^*, m_i) \in \mathsf{L}_{\mathsf{Sig}} \ \wedge \ \boldsymbol{y}^* \neq f^*(m_1, \ldots, m_n)$. Namely, $\mathcal{A}$ asked all queries (ℓ_i^*, m_i) to the signing oracle, but cheated at computing $\boldsymbol{y}^*$.

For both types of forgeries we can partition on whether the forgery is a fresh signature (i.e., $\mathcal{P} = \mathcal{I}_\ell$) or an evaluated one. In the event of type 2 forgeries, for our scheme we can also partition over the event '$\mathsf{c}^* = \mathsf{FC.Com}(\mathsf{ck}, m_1, \ldots, m_n)$', where c^* is the (deterministic) commitment included in σ^*.

Let also VER be the event that verification passes and that no user involved in a labeled program is corrupted, i.e.,

$$\mathsf{VER} := \mathsf{MKHS.Ver}(\mathsf{pp}, \mathcal{P}^*, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}^*}, \boldsymbol{y}^*, \sigma^*) = 1 \wedge \{\mathsf{id} \in \mathcal{P}^*\} \cap \mathsf{L}_{\mathsf{Corr}} = \emptyset.$$

We define 4 experiments, UF_1, UF_2, UF_3, and UF_4:

- UF_1 outputs 1 iff $\mathsf{VER} \wedge \mathcal{P} \neq \mathcal{I}_\ell \wedge \mathsf{TYPE}_1$.
- UF_2 outputs 1 iff $\mathsf{VER} \wedge \mathcal{P} \neq \mathcal{I}_\ell \wedge \mathsf{TYPE}_2 \wedge (\mathsf{c}^* = \mathsf{FC.Com}(\mathsf{ck}, m_1, \ldots, m_n))$.
- UF_3 outputs 1 iff $\mathsf{VER} \wedge \mathcal{P} \neq \mathcal{I}_\ell \wedge \mathsf{TYPE}_2 \wedge (\mathsf{c}^* \neq \mathsf{FC.Com}(\mathsf{ck}, m_1, \ldots, m_n))$.
- UF_4 outputs 1 iff $\mathsf{VER} \wedge \mathcal{P} = \mathcal{I}_\ell \wedge (\mathsf{TYPE}_1 \vee \mathsf{TYPE}_2)$.

Overall, we partitioned the probability space so that, by the union bound, for any PPT adversary $\mathcal{A}$ we have that $\Pr[\mathsf{HomUF\text{-}CMA}_{\mathcal{A},\mathsf{MKHS}}(\lambda) = 1] \leq \sum_{k=1}^{4} \Pr[\mathsf{UF}_{k,\mathcal{A}}(\lambda) = 1]$. We separate the proof in lemmas that bound the probability that $\mathcal{A}$ wins in each of the experiments.

Lemma 1. *For any PPT adversary $\mathcal{A}$ making at most $Q = \mathsf{poly}(\lambda)$ queries to the key generation oracle and that can produce a valid forgery in UF_1, there exist PPT adversaries $\mathcal{B}_{sind}, \mathcal{B}_{sExt}, \mathcal{B}_{\text{EUF-CMA}}$ against the BARG setup indistinguishability, somewhere extractability and the* EUF-CMA *property of the digital signature scheme Σ, such that:*

$$\Pr[\mathsf{UF}_{1,\mathcal{A}}(\lambda) = 1] \leq \\ n \cdot \left(\mathsf{Adv}^{\text{sind}}_{\mathsf{BARG},\mathcal{B}_{sind}}(\lambda) + \mathsf{Adv}^{\text{sext}}_{\mathsf{BARG},\mathcal{B}_{sExt}}(\lambda) + Q \cdot \mathsf{Adv}^{\text{euf-cma}}_{\Sigma,\mathcal{B}_{\text{EUF-CMA}}}(\lambda)\right).$$

Proof. We first define WIN_1 as the winning event of UF_1:

$$\mathsf{WIN}_1 := \begin{cases} \exists j \in [n] : (\ell_j^* = (\mathsf{id}_j^*, \tau_j^*), \cdot) \notin \mathsf{L}_{\mathsf{Sig}} \wedge \mathsf{id}_j^* \notin \mathsf{L}_{\mathsf{Corr}} \\ \wedge\ \mathsf{BARG.Ver}(\mathsf{crs}, \mathcal{C}, \{\mathsf{x}_i^*\}_i, \pi_\sigma^*) = 1 \end{cases}.$$

Notice that WIN_1 is implied by $\mathsf{VER} \wedge \mathsf{TYPE}_1$, and that we have suppressed unnecesary checks that will not be used in the proof of this lemma. Based on this winning condition, we define a series of hybrid games $\mathsf{Hyb}^0, \mathsf{Hyb}^1, \mathsf{Hyb}^2, \mathsf{Hyb}^3$ described in Figure 7.

$\underline{\mathsf{Hyb}^0}$**:** As described above, this game is a simplified version of UF_1 where we omit unnecessary outputs from the adversary and from the winning condition.

$\underline{\mathsf{Hyb}^1}$**:** To transition from Hyb^0 to Hyb^1, since the choice of j^* is uniform over $[n]$, we have that $j = j^*$ with probability $\frac{1}{n}$. As a result we have that:

$$\Pr[\mathsf{Hyb}^1_{\mathcal{A}}(\lambda) = 1] \geq \frac{1}{n} \Pr[\mathsf{Hyb}^0_{\mathcal{A}}(\lambda) = 1].$$

$\underline{\mathsf{Hyb}^2}$**:** In this game, the only difference with Hyb^1 is that the BARG setup is set in trapdoor mode at position j^*. Then, we have that if $\mathcal{A}$ interpolates between Hyb^1 and Hyb^2, we can construct an adversary $\mathcal{B}_{sind}$ against BARG setup indistinguishability property such that

$$\Pr[\mathsf{Hyb}^2_{\mathcal{A}}(\lambda) = 1] \leq \Pr[\mathsf{Hyb}^1_{\mathcal{A}}(\lambda) = 1] + \mathsf{Adv}^{\text{sind}}_{\mathsf{BARG},\mathcal{B}_{sind}}(\lambda).$$

$\mathsf{Hyb}^0_{\mathcal{A}}(\lambda)$:	$\mathsf{Hyb}^1_{\mathcal{A}}(\lambda)$:
$\mathsf{crs} \leftarrow \mathsf{BARG.Setup}()$ $\mathsf{dk} \leftarrow \mathsf{SEC.Setup}()$ $\mathsf{ck} \leftarrow \mathsf{FC.Setup}()$ $(\mathcal{P}^*, \pi^*_\sigma, \{\mathtt{x}^*_i\}_i) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{crs}, \mathsf{dk}, \mathsf{ck})$ **Output 1 iff:** $\mathsf{WIN}_1 = 1$	$j^* \leftarrow\$ [n]$ $\mathsf{crs} \leftarrow \mathsf{BARG.Setup}()$ $\mathsf{dk} \leftarrow \mathsf{SEC.Setup}()$ $\mathsf{ck} \leftarrow \mathsf{FC.Setup}()$ $(\mathcal{P}^*, \pi^*_\sigma, \{\mathtt{x}^*_i\}_i) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{crs}, \mathsf{dk}, \mathsf{ck})$ **Output 1 iff:** $\mathsf{WIN}_1 = 1$ $\wedge (j^* = j)$
$\mathsf{Hyb}^2_{\mathcal{A}}(\lambda)$:	$\mathsf{Hyb}^3_{\mathcal{A}}(\lambda)$:
$j^* \leftarrow\$ [n]$ $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{BARG.TdSetup}(j^*)$ $\mathsf{dk} \leftarrow \mathsf{SEC.Setup}()$ $\mathsf{ck} \leftarrow \mathsf{FC.Setup}()$ $(\mathcal{P}^*, \pi^*_\sigma, \{\mathtt{x}^*_i\}_i) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{crs}, \mathsf{dk}, \mathsf{ck})$ **Output 1 iff:** $\mathsf{WIN}_1 = 1$ $\wedge (j^* = j)$	$j^* \leftarrow\$ [n]$ $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{BARG.TdSetup}(j^*)$ $\mathsf{dk} \leftarrow \mathsf{SEC.Setup}()$ $\mathsf{ck} \leftarrow \mathsf{FC.Setup}()$ $\bar{\mathtt{w}}_{j^*} \leftarrow \mathsf{BARG.Ext}(\mathsf{td}, \mathcal{C}, \pi^*_\sigma)$ $(\mathcal{P}^*, \pi^*_\sigma, \{\mathtt{x}^*_i\}_i) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{crs}, \mathsf{dk}, \mathsf{ck})$ **Output 1 iff:** $\mathsf{WIN}_1 = 1$ $\wedge (j^* = j)$ $\wedge\, \mathcal{C}(\mathtt{x}^*_{j^*}, \bar{\mathtt{w}}_{j^*}) = 1$

Fig. 7: Games $\mathsf{Hyb}^0, \mathsf{Hyb}^1, \mathsf{Hyb}^2, \mathsf{Hyb}^3$ for the proof of Lemma 1. We highlight changes between games and omit inputs to Setup for succinctness.

$\underline{\mathsf{Hyb}^3}$**:** If $\mathcal{A}$ outputs 1 against Hyb^2 but outputs 0 against Hyb^3, it must be the case that:

- $\mathsf{BARG.Ver}(\mathsf{crs}, \mathcal{C}, \{\mathtt{x}_i\}_i, \pi_{\sigma^*}) = 1$,
- $\mathcal{C}(\mathtt{x}_{j^*}, \bar{\mathtt{w}}_{j^*}) \neq 1$ where $\bar{\mathtt{w}}_{j^*}$ is obtained from $\mathsf{BARG.Ext}(\mathsf{td}, \mathcal{C}, \pi^*_\sigma)$.

Then, we can use $\mathcal{A}$ to construct an adversary $\mathcal{B}_{sExt}$ against BARG somewhere extractability such that:

$$\Pr[\mathsf{Hyb}^3_{\mathcal{A}}(\lambda) = 1] \leq \Pr[\mathsf{Hyb}^2_{\mathcal{A}}(\lambda) = 1] + \mathsf{Adv}^{\mathrm{sext}}_{\mathsf{BARG}, \mathcal{B}_{sExt}}(\lambda).$$

Finally, we proceed to bound the advantage of $\mathcal{A}$ in Hyb^3. Recall that $\mathcal{A}$ can make at most $Q = \mathsf{poly}(\lambda)$ queries to the key generation oracle $\mathcal{O}^{\mathsf{KeyGen}}$. We use $\mathcal{A}$ to build an algorithm $\mathcal{B}_{\text{EUF-CMA}}$ that breaks the existential unforgeability of Σ. $\mathcal{B}_{\text{EUF-CMA}}$ simulates the game Hyb^3 to $\mathcal{A}$, proceeding as follows:

1. $\mathcal{B}_{\text{EUF-CMA}}$ receives the verification key vk^* from the EUF-CMA challenger.
2. $\mathcal{B}_{\text{EUF-CMA}}$ starts by uniformly sampling $j^* \leftarrow\$ [n]$ and $q^* \leftarrow\$ [Q]$. It initializes empty lists $\mathsf{L}_{\mathsf{ID}}, \mathsf{L}_{\mathsf{Sig}} \leftarrow \emptyset$. Then $\mathcal{B}_{\text{EUF-CMA}}$ runs the setup algorithm for $\mathsf{BARG}, \mathsf{FC}, \mathsf{MKHS}$, setting $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{BARG.TdSetup}(j^*)$. It then sends the public parameters pp to $\mathcal{A}$.
3. Whenever $\mathcal{A}$ makes a query to $\mathcal{O}^{\mathsf{KeyGen}}(\mathsf{id})$:
   - If the query is the q^*-th query, let $\mathsf{vk}_{\mathsf{id}} = \mathsf{vk}^*$ and return $\mathsf{vk}_{\mathsf{id}}$ to $\mathcal{A}$.
   - Otherwise, let $(\mathsf{vk}_{\mathsf{id}}, \mathsf{sk}_{\mathsf{id}}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\lambda)$.

4. Whenever $\mathcal{A}$ makes a query to $\mathcal{O}^{\mathsf{Sign}}(m, \ell = (\mathsf{id}, \tau))$:
   - If $(\ell, m) \notin \mathsf{L}_{\mathsf{Sig}}$:
     - If $\mathsf{id} = \mathsf{id}_{q^*}$: forward the query to the EUF-CMA oracle $\sigma \leftarrow \mathcal{O}^{\mathsf{Sign}}(m|\ell)$, update $\mathsf{L}_{\mathsf{Sig}} := \mathsf{L}_{\mathsf{Sig}} \cup (\ell, m)$ and then send σ to $\mathcal{A}$.
     - Otherwise, if $\mathsf{id} \neq \mathsf{id}_{q^*}$: compute $\sigma_\ell \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{id}}, \ell, m)$, update $\mathsf{L}_{\mathsf{Sig}} := \mathsf{L}_{\mathsf{Sig}} \cup (\ell, m)$, and return σ_ℓ to $\mathcal{A}$.
   - Else, if $(\ell, m) \in \mathsf{L}_{\mathsf{Sig}}$, ignore the query.
5. Whenever $\mathcal{A}$ makes a query to $\mathcal{O}^{\mathsf{Corr}}(\mathsf{id})$:
   - if $\mathsf{id} = \mathsf{id}_{q^*}$, abort
   - else, if $\mathsf{id} \in \mathsf{L}_{\mathsf{ID}}$ and $\mathsf{id} \notin \mathsf{L}_{\mathsf{Corr}}$, update $\mathsf{L}_{\mathsf{Corr}} \leftarrow \mathsf{L}_{\mathsf{Corr}} \cup \mathsf{id}$, and return $\mathsf{sk}_{\mathsf{id}}$.
6. At the end of the game $\mathcal{A}$ outputs $(\mathcal{P}^*, y^*, \sigma^*)$. $\mathcal{B}_{\text{EUF-CMA}}$ checks that $\mathsf{BARG.Ver}(\mathsf{crs}, \mathcal{C}, \{\mathtt{x}_i^*\}_i, \pi_\sigma^*) = 1$ and that $(\ell_{j^*}^* = (\mathsf{id}_{j^*}^*, \tau_{j^*}^*), \cdot) \notin \mathsf{L}_{\mathsf{Sig}}$ and $\mathsf{id}_{j^*}^* \notin \mathsf{L}_{\mathsf{Corr}}$. Additionally, it checks that $\mathsf{id}_{j^*}^* = \mathsf{id}_{q^*}$. If any of these checks does not pass, $\mathcal{B}_{\text{EUF-CMA}}$ aborts. Otherwise it computes $\bar{\mathtt{w}}_{j^*} \leftarrow \mathsf{BARG.Ext}(\mathsf{td}, \mathcal{C}, \{\mathtt{x}_i^*\}_{i\in[n]}, \pi_\sigma^*)$ and parses $\bar{\sigma}_{j^*}$ and $\bar{m}_{j^*}$ from $\bar{\mathtt{w}}_{j^*}$. At the end $\mathcal{B}_{\text{EUF-CMA}}$ outputs $(\bar{m}_{j^*}|\ell_{j^*}^*, \bar{\sigma}_{j^*})$ as its forgery.

By construction, conditioned on $\mathsf{id}_{j^*}^* = \mathsf{id}_{q^*}$, algorithm $\mathcal{B}_{\text{EUF-CMA}}$ perfectly simulates an execution of Hyb^3 to $\mathcal{A}$. Note that, overall, the probability of not aborting during the simulation is at least $1/Q$, since the winning condition guarantees that there exists at least one identity that remains uncorrupted. If all guesses are correct, as $\mathcal{C}$ is explicitly checking $\Sigma.\mathsf{Ver}(\mathsf{vk}^*, \bar{m}_{j^*}|\ell_{j^*}^*, \bar{\sigma}_{j^*}) = 1$, it means that $\bar{\sigma}_{j^*}$ is a valid signature on $\bar{m}_{j^*}|\ell_{j^*}^*$.

Thus with probability at least $\frac{1}{Q} \cdot \Pr[\mathsf{Hyb}_{\mathcal{A}}^3(\lambda) = 1]$, $\mathcal{B}_{\text{EUF-CMA}}$ outputs a valid EUF-CMA forgery. In summary,

$$\Pr[\mathsf{Hyb}_{\mathcal{A}}^3(1^\lambda) = 1] \leq Q \cdot \mathsf{Adv}_{\Sigma, \mathcal{B}_{\text{EUF-CMA}}}^{\text{euf-cma}}(\lambda).$$

Lemma 2. *For any PPT adversary $\mathcal{A}$ that can produce a valid forgery against UF_2, there exists a PPT adversary $\mathcal{B}$ against evaluation binding of the functional commitment scheme FC such that*

$$\Pr[\mathsf{UF}_{2,\mathcal{A}}(\lambda) = 1] \leq \mathsf{Adv}_{\mathsf{FC},\mathcal{B}}^{\text{evbind}}(\lambda).$$

Proof. As in the proof of Lemma 1, we first define a wining event WIN_2 as a simplification of the winning condition of UF_2 which only includes the checks that are relevant for the reduction.

$$\mathsf{WIN}_2 := \begin{cases} \forall i \in [n], (\ell_i^*, m_i) \in \mathsf{L}_{\mathsf{Sig}} \\ \wedge\, \boldsymbol{y}^* \neq f^*(m_1, \ldots, m_n) \\ \wedge\, \mathsf{c}^* = \mathsf{FC.Com}(\mathsf{ck}, (m_1, \ldots, m_n)) \\ \wedge\, \mathsf{FC.Ver}(\mathsf{ck}, \mathsf{c}^*, \boldsymbol{y}^*, f^*, \pi_{f^*}^*) = 1 \end{cases}.$$

We describe how to build an efficient algorithm $\mathcal{B}$ that breaks the evaluation binding of FC.

1. $\mathcal{B}$ receives a commitment key ck by the challenger of the evaluation binding game.
2. $\mathcal{B}$ initialize empty lists $\mathsf{L}_{\mathsf{ID}}, \mathsf{L}_{\mathsf{Sig}} \leftarrow \emptyset$. Then $\mathcal{B}$ runs $\mathsf{crs} \leftarrow \mathsf{BARG.Setup}(1^\lambda, n, 1^{|\mathcal{C}|})$ and $\mathsf{dk} \leftarrow \mathsf{SEC.Setup}(1^\lambda, n, B)$ and sends $\mathsf{pp} \leftarrow (\mathsf{crs}, \mathsf{dk}, \mathsf{ck})$ to $\mathcal{A}$.
3. $\mathcal{B}$ simulates all of $\mathcal{A}$'s queries to $\mathcal{O}^{\mathsf{KeyGen}}, \mathcal{O}^{\mathsf{Sign}}, \mathcal{O}^{\mathsf{Corr}}$ by using knowledge of the secret keys, and updates the list $\mathsf{L}_{\mathsf{Sig}}$ every time a fresh $\mathcal{O}^{\mathsf{Sign}}(\ell^*, m)$ query is made by $\mathcal{A}$.
4. At the end of the simulation, $\mathcal{A}$ outputs $(\mathcal{P}^*, \mathsf{c}^*, \pi_{f^*}^*)$ (we ignore the remaining outputs). $\mathcal{B}$ parses $(f^*, (\ell_1^*, \ldots, \ell_n^*))$ from $\mathcal{P}^*$, and retrieves the messages $m_1, \ldots, m_n$ associated to labels $\ell_1^*, \ldots, \ell_n^*$ from $\mathsf{L}_{\mathsf{Sig}}$.

5. Finally, $\mathcal{B}$ computes the honest output $\boldsymbol{y} = f^*(m_1, \ldots, m_n)$, and an honest FC opening proof to $\boldsymbol{y}$ as $\pi_{f^*} \leftarrow \mathsf{FC.Open}(\mathsf{ck}, (m_1, \ldots, m_n), f^*)$. Then, $\mathcal{B}$ outputs $(\mathsf{c}^*, f^*, \boldsymbol{y}, \pi_{f^*}, \boldsymbol{y}^*, \pi^*_{f^*})$.

By construction, $\mathcal{B}$ perfectly simulates an execution of the MKHS game for $\mathcal{A}$. Also, note that if $\mathcal{A}$ is a successful adversary against UF_2, then by the WIN_2 event, the messages $m_1, \ldots, m_n$ retrieved from $\mathsf{L}_{\mathsf{Sig}}$ must be the same ones that are committed under c^*. As c^* and $\boldsymbol{y}$ are honest, we have that $\mathsf{FC.Ver}(\mathsf{ck}, \mathsf{c}^*, \boldsymbol{y}, f^*, \pi_{f^*}) = 1$.

Thus, $\mathcal{B}$'s output is a valid output in the FC evaluation binding game. To summarize,

$$\Pr[\mathsf{UF}_{2,\mathcal{A}}(\lambda) = 1] \leq \mathsf{Adv}^{\text{evbind}}_{\mathsf{FC},\mathcal{B}}(\lambda).$$

Lemma 3. *For any PPT adversary $\mathcal{A}$ that wins in the UF_3 game, there exists a tuple of PPT adversaries $(\mathcal{B}_1, \ldots, \mathcal{B}_6)$ such that*

$$\Pr[\mathsf{UF}_{3\mathcal{A}}(\lambda) = 1] \leq n \cdot \Big[\mathsf{Adv}^{\text{sind}}_{\mathsf{BARG},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\text{sext}}_{\mathsf{BARG},\mathcal{B}_2}(\lambda) + 2 \cdot \mathsf{Adv}^{\text{sind}}_{\mathsf{SEC},\mathcal{B}_3}(\lambda) + \mathsf{Adv}^{\text{sext}}_{\mathsf{SEC},\mathcal{B}_4}(\lambda) + Q \cdot \mathsf{Adv}^{\text{eufcma}}_{\Sigma,\mathcal{B}_5}(\lambda) + \mathsf{Adv}^{\text{updbind}}_{\mathsf{FC},\mathcal{B}_6}(\lambda)\Big].$$

Proof. For $\mathcal{A}$ to win in event UF_3, it must have crafted a type 2 forgery $\boldsymbol{y}^* \neq f^*(m_1, \ldots, m_n)$ such that $\forall i \in [n], (\ell^*_i, m_i) \in \mathsf{L}_{\mathsf{Sig}}$, and such that $\{\mathsf{id} \in \mathcal{P}^*\} \cap \mathsf{L}_{\mathsf{Corr}} = \emptyset$. Besides, the commitment c^* to the messages must not be honestly computed, $\mathsf{c}^* \neq \mathsf{FC.Com}(\mathsf{ck}, \boldsymbol{m})$. We will prove the lemma through a long sequence of hybrid sub-games $\mathsf{Hyb}^{1,0}, \ldots, \mathsf{Hyb}^{n,8}, \mathsf{Hyb}^{n,8*}$. First of all, we describe the following winning event:

$$\mathsf{WIN}_3 := \begin{cases} \forall i \in [n], (\ell^*_i, m_i) \in \mathsf{L}_{\mathsf{Sig}} \\ \wedge \{\mathsf{id} \in \mathcal{P}^*\} \cap \mathsf{L}_{\mathsf{Corr}} = \emptyset \\ \wedge \mathsf{BARG.Ver}(\mathsf{crs}, \mathcal{C}, \{\mathtt{x}_i\}_i, \pi_\sigma) = 1 \\ \wedge \mathsf{c}^* \neq \mathsf{FC.Com}(\mathsf{ck}, \boldsymbol{m}) \end{cases}.$$

As in previous lemmas, note that WIN_3 only includes a subset of the checks in $\mathsf{MKHS.Ver}$, as the other conditions (in particular, FC verification) are not relevant for this lemma. Based on this winning condition, we introduce an initial $\mathsf{Hyb}^{1,0}$ in Figure 8 as a simplification of UF_3 where we omit the outputs $\pi_f, \boldsymbol{y}^*$ from the adversary. As the winning condition in $\mathsf{Hyb}^{1,0}$ is less strict than in UF_3 while the pre-conditions remain the same, any adversary winning in UF_3 also wins in $\mathsf{Hyb}^{1,0}$. Hence, $\Pr[\mathsf{UF}_{3\mathcal{A}}(\lambda) = 1] \leq \Pr[\mathsf{Hyb}^{1,0}_{\mathcal{A}}(\lambda) = 1]$.

Games $\mathsf{Hyb}^{1,j}$: We formally introduce the hybrid games in Figures 8, 9, 10, and 11. We progress through the hybrids below.

$\mathsf{Hyb}^{1,1}$**:** The transition from $\mathsf{Hyb}^{1,0}$ to $\mathsf{Hyb}^{1,1}$, where we switch $\mathsf{BARG.Setup}$ to trapdoor mode $\mathsf{BARG.TdSetup}(1)$ at index 1, follows easily by the setup indistinguishability property of BARG. We have that there exists a PPT adversary $\mathcal{B}_1$ against BARG setup indistinguishability such that

$$\Pr[\mathsf{Hyb}^{1,0}_{\mathcal{A}}(\lambda) = 1] \leq \Pr[\mathsf{Hyb}^{1,1}_{\mathcal{A}}(\lambda) = 1] + \mathsf{Adv}^{\text{sind}}_{\mathsf{BARG},\mathcal{B}_1}(\lambda).$$

$\mathsf{Hyb}^{1,2}$**:** In this step, we additionally extract from BARG at position 1 and abort if $\mathcal{C}(\mathtt{x}_1, \bar{\mathtt{w}}_1) \neq 1$ for the extracted witness $\bar{\mathtt{w}}_1$. The witness is given by $\bar{\mathtt{w}}_1 = (\bar{m}_1, \bar{\sigma}_1, \bar{\pi}_1, \bar{\mathsf{c}}_0, \bar{\mathsf{c}}_1, \bar{o}_0, \bar{o}_1)$, where $\bar{\mathsf{c}}_0$ and $\bar{o}_0$ are irrelevant for the proof. It follows that there exists a PPT adversary $\mathcal{B}_2$ against BARG somewhere extractability such that

$$\Pr[\mathsf{Hyb}^{1,1}_{\mathcal{A}}(\lambda) = 1] \leq \Pr[\mathsf{Hyb}^{1,2}_{\mathcal{A}}(\lambda) = 1] + \mathsf{Adv}^{\text{sext}}_{\mathsf{BARG},\mathcal{B}_2}(\lambda).$$

$\mathsf{Hyb}^{1,0}_{\mathcal{A}}(\lambda)$:	$\mathsf{Hyb}^{1,1}_{\mathcal{A}}(\lambda)$:
$\mathsf{crs} \leftarrow \mathsf{BARG.Setup}()$	$(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{BARG.TdSetup}(1)$
$\mathsf{dk} \leftarrow \mathsf{SEC.Setup}()$	$\mathsf{dk} \leftarrow \mathsf{SEC.Setup}()$
$\mathsf{ck} \leftarrow \mathsf{FC.Setup}()$	$\mathsf{ck} \leftarrow \mathsf{FC.Setup}()$
$(\mathcal{P}^*, \{\mathtt{x}_i^*\}_i, \mathsf{c}^*, \pi_\sigma, \mathsf{c}_w)$	$(\mathcal{P}^*, \{\mathtt{x}_i^*\}_i, \mathsf{c}^*, \pi_\sigma, \mathsf{c}_w)$
$\leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{crs}, \mathsf{dk}, \mathsf{ck})$	$\leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{crs}, \mathsf{dk}, \mathsf{ck})$
Output 1 iff $\mathsf{WIN}_3 = 1$	**Output 1 iff** $\mathsf{WIN}_3 = 1$

Fig. 8: Games $\mathsf{Hyb}^{1,0}, \mathsf{Hyb}^{1,1}$ for the proof of Lemma 3. We highlight changes between games and omit inputs to Setup for succinctness.

$\mathsf{Hyb}^{1,2}_{\mathcal{A}}(\lambda)$:	$\mathsf{Hyb}^{1,3}_{\mathcal{A}}(\lambda)$:
$(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{BARG.TdSetup}(1)$	$(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{BARG.TdSetup}(1)$
$\mathsf{dk} \leftarrow \mathsf{SEC.Setup}()$	$(\mathsf{dk}, \mathsf{td}_c) \leftarrow \mathsf{SEC.TdSetup}(1)$
$\mathsf{ck} \leftarrow \mathsf{FC.Setup}()$	$\mathsf{ck} \leftarrow \mathsf{FC.Setup}()$
$(\mathcal{P}^*, \{\mathtt{x}_i^*\}_i, \mathsf{c}^*, \pi_\sigma, \mathsf{c}_w)$	$(\mathcal{P}^*, \{\mathtt{x}_i^*\}_i, \mathsf{c}^*, \pi_\sigma, \mathsf{c}_w)$
$\leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{crs}, \mathsf{dk}, \mathsf{ck})$	$\leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{crs}, \mathsf{dk}, \mathsf{ck})$
$\bar{\mathtt{w}}_1 \leftarrow \mathsf{BARG.Ext}(\mathsf{td}, \mathcal{C}, \pi_\sigma)$	$\bar{\mathtt{w}}_1 \leftarrow \mathsf{BARG.Ext}(\mathsf{td}, \mathcal{C}, \pi_\sigma)$
Output 1 iff:	**Output 1 iff:**
$\mathsf{WIN}_3 = 1$	$\mathsf{WIN}_3 = 1$
$\wedge\, \mathcal{C}(\mathtt{x}_1, \bar{\mathtt{w}}_1) = 1$	$\wedge\, \mathcal{C}(\mathtt{x}_1, \bar{\mathtt{w}}_1) = 1$

Fig. 9: Games $\mathsf{Hyb}^{1,2}, \mathsf{Hyb}^{1,3}$ for the proof of Lemma 3.

$\mathsf{Hyb}^{1,3}$**:** In this game, we set $\mathsf{SEC.Setup}$ extractable at index 1. We have that there exists an adversary $\mathcal{B}_4$ against SEC setup indistinguishability such that

$$\Pr[\mathsf{Hyb}^{1,2}_{\mathcal{A}}(\lambda) = 1] \leq \Pr[\mathsf{Hyb}^{1,3}_{\mathcal{A}}(\lambda) = 1] + \mathsf{Adv}^{\mathrm{sind}}_{\mathsf{SEC}}(\lambda).$$

$\mathsf{Hyb}^{1,4}$**:** In this game, we extract $\hat{\mathsf{c}}_1 \leftarrow \mathsf{SEC.Ext}(\mathsf{td}_c, \mathsf{c}_w)$ and abort if $\hat{\mathsf{c}}_1 \neq \bar{\mathsf{c}}_1$. To prove the transition from the previous game, note that, by definition, $\mathcal{C}(\mathtt{x}_1, \bar{\mathtt{w}}_1) = 1$ implies that $\mathsf{SEC.Ver}(\mathsf{dk}_1, \mathsf{c}_w, 1, \bar{\mathsf{c}}_1, \bar{o}_1) = 1$. Hence, if $\mathsf{SEC.Ext}(\mathsf{td}_c, \mathsf{c}_w, i^*) \neq \bar{\mathsf{c}}_{i^*}$, then we can create an adversary $\mathcal{B}_3$ against SEC somewhere extractability. We have that

$$\Pr[\mathsf{Hyb}^{1,3}_{\mathcal{A}}(\lambda) = 1] \leq \Pr[\mathsf{Hyb}^{1,4}_{\mathcal{A}}(\lambda) = 1] + \mathsf{Adv}^{\mathrm{sext}}_{\mathsf{SEC},\mathcal{B}_3}(\lambda).$$

$\mathsf{Hyb}^{1,5}$**:** In this game, we add the requirement that the extracted $\bar{m}_1 \in \bar{\mathtt{w}}_1$ equals the honest m_1, where m_1 is the message that $\mathcal{A}$ queries to the $\mathcal{O}^{\mathsf{Sign}}$ oracle on label $\ell_1^* \in \mathcal{P}^*$. By definition of $\mathcal{C}$, we have that $\mathcal{C}(\mathtt{x}_1, \bar{\mathtt{w}}_1) = 1$ and therefore $\Sigma.\mathsf{Ver}(\mathsf{vk}_1, \bar{m}_1 | \ell_1^*, \bar{\sigma}_1) = 1$. If $m_1 \neq \bar{m}_1$, then $\mathcal{A}$ must have produced a signature forgery $(\bar{m}_1 | \ell_1^*, \bar{\sigma}_1)$ for key vk_1.
In a more careful analysis, we bound the probability of this event by constructing an adversary $\mathcal{B}_5$ against the unforgeability of the signature scheme Σ. $\mathcal{B}_5$ runs on input a verification key vk^* from the EUF-CMA challenger; it chooses a random index $q^* \leftarrow\!\!\$\ [Q]$, where $Q = \mathsf{poly}(\lambda)$ is the

$\mathsf{Hyb}^{1,4}_{\mathcal{A}}(\lambda)$:	$\mathsf{Hyb}^{1,5}_{\mathcal{A}}(\lambda)$:
$(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{BARG.TdSetup}(1)$	$(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{BARG.TdSetup}(1)$
$(\mathsf{dk}, \mathsf{td}_c) \leftarrow \mathsf{SEC.TdSetup}(1)$	$(\mathsf{dk}, \mathsf{td}_c) \leftarrow \mathsf{SEC.TdSetup}(1)$
$\mathsf{ck} \leftarrow \mathsf{FC.Setup}()$	$\mathsf{ck} \leftarrow \mathsf{FC.Setup}()$
$(\mathcal{P}^*, \{\mathtt{x}^*_i\}_i, \mathsf{c}^*, \pi_\sigma, \mathsf{c}_w)$	$(\mathcal{P}^*, \{\mathtt{x}^*_i\}_i, \mathsf{c}^*, \pi_\sigma, \mathsf{c}_w)$
$\leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{crs}, \mathsf{dk}, \mathsf{ck})$	$\leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{crs}, \mathsf{dk}, \mathsf{ck})$
$\bar{\mathtt{w}}_1 \leftarrow \mathsf{BARG.Ext}(\mathsf{td}, \mathcal{C}, \pi_\sigma)$	$\bar{\mathtt{w}}_1 \leftarrow \mathsf{BARG.Ext}(\mathsf{td}, \mathcal{C}, \pi_\sigma)$
$\hat{\mathsf{c}}_1 \leftarrow \mathsf{SEC.Ext}(\mathsf{td}_c, \mathsf{c}_w)$	$\hat{\mathsf{c}}_1 \leftarrow \mathsf{SEC.Ext}(\mathsf{td}_c, \mathsf{c}_w)$
Output 1 iff:	**Output 1 iff:**
$\mathsf{WIN}_3 = 1$	$\mathsf{WIN}_3 = 1$
$\wedge\, \mathcal{C}(\mathtt{x}_1, \bar{\mathtt{w}}_1) = 1$	$\wedge\, \mathcal{C}(\mathtt{x}_1, \bar{\mathtt{w}}_1) = 1$
$\wedge\, \hat{\mathsf{c}}_1 = \bar{\mathsf{c}}_1$	$\wedge\, \hat{\mathsf{c}}_1 = \bar{\mathsf{c}}_1$
	$\wedge\, \bar{m}_1 = m_1$

Fig. 10: Games $\mathsf{Hyb}^{1,4}, \mathsf{Hyb}^{1,5}$ for the proof of Lemma 3.

number of queries that $\mathcal{A}$ can make to the $\mathcal{O}^{\mathsf{KeyGen}}$ oracle, and then it adaptively simulates all $\mathcal{O}^{\mathsf{KeyGen}}$, $\mathcal{O}^{\mathsf{Sign}}$ and $\mathcal{O}^{\mathsf{Corr}}$ queries for $\mathcal{A}$ as follows:

– For the i-th query to $\mathcal{O}^{\mathsf{KeyGen}}(\mathsf{id})$, if $i = q^*$, return $\mathsf{vk}_{\mathsf{id}} = \mathsf{vk}^*$, otherwise generate a keypair $(\mathsf{vk}_{\mathsf{id}}, \mathsf{sk}_{\mathsf{id}}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\lambda)$ and return $\mathsf{vk}_{\mathsf{id}}$.
– All $\mathcal{O}^{\mathsf{Sign}}((\mathsf{id}, \cdot), \cdot)$ queries such that $\mathsf{id} = \mathsf{id}_{q^*}$ are answered using the $\mathcal{O}^{\mathsf{Sign}}(\cdot)$ oracle of the EUF-CMA challenger, where all the remaining queries are answered by using the secret key $\mathsf{sk}_{\mathsf{id}}$, which is known to $\mathcal{B}_5$.
– If $\mathcal{A}$ makes a query $\mathcal{O}^{\mathsf{Corr}}(\mathsf{id})$ such that $\mathsf{id} = \mathsf{id}_{q^*}$ abort, otherwise return the corresponding secret key $\mathsf{sk}_{\mathsf{id}}$.

After $\mathcal{A}$ outputs $(\mathcal{P}^*, \{\mathtt{x}^*_i\}_i, \mathsf{c}^*, \pi_\sigma, \mathsf{c}_w)$, $\mathcal{B}_5$ parses the labels $\ell^*_1 = (\mathsf{id}^*_1, \cdot)$ in $\mathcal{P}^*$ and aborts if $\mathsf{id}^*_1 \neq \mathsf{id}_{q^*}$.
If $\mathcal{B}_5$ does not abort, then the simulation is perfect and it must be that $(\bar{m}_1 | \ell^*_1, \bar{\sigma}_1)$ is a valid forgery for the EUF-CMA game. This holds as the WIN_3 condition enforces that $\mathcal{A}$ is only allowed to output labels ℓ^*_i such that (ℓ^*_i, m_i) was queried to $\mathcal{O}^{\mathsf{Sign}}$. Moreover, WIN_3 also enforces that vk_1 is not a corrupted key. Namely, id^*_1 is one of the non-corrupted keys and thus the probability that $\mathcal{B}_5$ does not abort is $1/Q$. Thus, we conclude:

$$\Pr[\mathsf{Hyb}^{1,4}_{\mathcal{A}}(\lambda) = 1] \leq \Pr[\mathsf{Hyb}^{1,5}_{\mathcal{A}}(\lambda) = 1] + Q \cdot \mathsf{Adv}^{\mathrm{eufcma}}_{\Sigma, \mathcal{B}_5}(\lambda).$$

$\underline{\mathsf{Hyb}^{1,6}}$**:** In this game, we compute the honest partial commitment $\mathsf{c}_1 \leftarrow \mathsf{FC.Com}(\mathsf{ck}, (m_1, 0, \ldots, 0))$, and require that $\bar{\mathsf{c}}_1 = \mathsf{c}_1$, and by extension, that $\hat{\mathsf{c}}_1 = \mathsf{c}_1$. This step follows by the updatability of FC, since $\mathcal{C}(\mathtt{x}_1, \bar{\mathtt{w}}_1) = 1$ only holds if $\mathsf{FC.VerUpd}(\mathsf{ck}_1, 1, \bar{\mathsf{c}}_0, 0, \bar{\mathsf{c}}_1, \bar{m}_1, \bar{\pi}_1) = 1$. As $\bar{m}_1 = m_1$, if $\bar{\mathsf{c}}_1 \neq \mathsf{c}_1$ then we break FC updatability soundness. Hence, there exists an adversary $\mathcal{B}_6$ against FC updatability such that

$$\Pr[\mathsf{Hyb}^{1,5}_{\mathcal{A}}(\lambda) = 1] \leq \Pr[\mathsf{Hyb}^{1,6}_{\mathcal{A}}(\lambda) = 1] + \mathsf{Adv}^{\mathrm{updbind}}_{\mathsf{FC}, \mathcal{B}_6}(\lambda).$$

$\mathsf{Hyb}^{1,6}_{\mathcal{A}}(\lambda)$:	$\mathsf{Hyb}^{1,7}_{\mathcal{A}}(\lambda)$:
$(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{BARG.TdSetup}(1)$	$(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{BARG.TdSetup}(1)$
$(\mathsf{dk}, \mathsf{td}_c) \leftarrow \mathsf{SEC.TdSetup}(1)$	$(\mathsf{dk}, \mathsf{td}_c) \leftarrow \mathsf{SEC.TdSetup}(1)$
$\mathsf{ck} \leftarrow \mathsf{FC.Setup}()$	$\mathsf{ck} \leftarrow \mathsf{FC.Setup}()$
$(\mathcal{P}^*, \{\mathtt{x}_i^*\}_i, \mathsf{c}^*, \pi_\sigma, \mathsf{c}_w)$	$(\mathcal{P}^*, \{\mathtt{x}_i^*\}_i, \mathsf{c}^*, \pi_\sigma, \mathsf{c}_w)$
$\leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{crs}, \mathsf{dk}, \mathsf{ck})$	$\leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{crs}, \mathsf{dk}, \mathsf{ck})$
$\bar{\mathtt{w}}_1 \leftarrow \mathsf{BARG.Ext}(\mathsf{td}, \mathcal{C}, \pi_\sigma)$	
$\hat{\mathsf{c}}_1 \leftarrow \mathsf{SEC.Ext}(\mathsf{td}_c, \mathsf{c}_w)$	$\hat{\mathsf{c}}_1 \leftarrow \mathsf{SEC.Ext}(\mathsf{td}_c, \mathsf{c}_w)$
$\mathsf{c}_1 \leftarrow \mathsf{FC.Com}(\mathsf{ck}, (m_1, \mathbf{0}))$	$\mathsf{c}_1 \leftarrow \mathsf{FC.Com}(\mathsf{ck}, (m_1, \mathbf{0}))$
Output 1 iff:	**Output 1 iff:**
$\mathsf{WIN}_3 = 1$	$\mathsf{WIN}_3 = 1$
$\wedge\, \mathcal{C}(\mathtt{x}_1, \bar{\mathtt{w}}_1) = 1$	$\wedge\, \hat{\mathsf{c}}_1 = \mathsf{c}_1$
$\wedge\, \hat{\mathsf{c}}_1 = \bar{\mathsf{c}}_1 = \mathsf{c}_1$	
$\wedge\, \bar{m}_1 = m_1$	

Fig. 11: Games $\mathsf{Hyb}^{1,6}, \mathsf{Hyb}^{1,7}$ for the proof of Lemma 3.

$\mathsf{Hyb}^{1,7}$**:** This game is a simplification of $\mathsf{Hyb}^{1,6}$ where we no longer extract from BARG, and hence we no longer have $\mathcal{C}(\mathtt{x}_1, \bar{\mathtt{w}}_1) = 1 \wedge \bar{m}_1 = m_1 \wedge \hat{\mathsf{c}}_1 = \bar{\mathsf{c}}_1$ in the winning condition. We have that

$$\Pr[\mathsf{Hyb}^{1,6}_{\mathcal{A}}(\lambda) = 1] = \Pr[\mathsf{Hyb}^{1,7}_{\mathcal{A}}(\lambda) = 1 \wedge \mathcal{C}(\mathtt{x}_1, \bar{\mathtt{w}}_1) = 1 \wedge \bar{m}_1 = m_1 \wedge \hat{\mathsf{c}}_1 = \bar{\mathsf{c}}_1] \\ \leq \Pr[\mathsf{Hyb}^{1,7}_{\mathcal{A}}(\lambda) = 1].$$

Note, this simplification makes the winning condition of $\mathsf{Hyb}^{1,7}$ independent of BARG extraction, which is crucial for changing the extraction index in the subsequent hybrid.

Games $\mathsf{Hyb}^{i,j}$ for $2 \leq i < n$: We introduce the hybrid games in Figures 12, 13, 14. First of all, we analyze the step from $\mathsf{Hyb}^{1,7}$ to $\mathsf{Hyb}^{2,1}$. Then, we analyze the generic step from $\mathsf{Hyb}^{i-1,9}$ to $\mathsf{Hyb}^{i,1}$ for $i > 2$, and proceed with the remaining hybrids.

$\mathsf{Hyb}^{2,1}$**:** In the transition from $\mathsf{Hyb}^{1,7}$ to $\mathsf{Hyb}^{2,1}$, we simply switch $\mathsf{BARG.TdSetup}(1)$ to the following index $\mathsf{BARG.TdSetup}(2)$. The step again follows by the setup indistinguishability of BARG. We have that

$$\Pr[\mathsf{Hyb}^{1,7}_{\mathcal{A}}(\lambda) = 1] \leq \Pr[\mathsf{Hyb}^{2,1}_{\mathcal{A}}(\lambda) = 1] + \mathsf{Adv}^{\mathsf{sind}}_{\mathsf{BARG},\mathcal{B}_1}(\lambda).$$

$\mathsf{Hyb}^{i,1}$**:** In the transition from $\mathsf{Hyb}^{i-1,9}$ to $\mathsf{Hyb}^{i,1}$, we also simply switch $\mathsf{BARG.TdSetup}(i-1)$ to the following index $\mathsf{BARG.TdSetup}(i)$. As above, the setup indistinguishability of BARG implies that

$$\Pr[\mathsf{Hyb}^{i,1}_{\mathcal{A}}(\lambda) = 1] \leq \Pr[\mathsf{Hyb}^{i-1,9}_{\mathcal{A}}(\lambda) = 1] + \mathsf{Adv}^{\mathsf{sind}}_{\mathsf{BARG},\mathcal{B}_1}(\lambda).$$

$\mathsf{Hyb}^{i,2}$**,** $\mathsf{Hyb}^{1,3}$**:** These steps are identical to the respective steps for $\mathsf{Hyb}^{1,2}, \mathsf{Hyb}^{1,3}$,

$$\Pr[\mathsf{Hyb}^{i,1}_{\mathcal{A}}(\lambda) = 1] \leq \Pr[\mathsf{Hyb}^{i,3}_{\mathcal{A}}(\lambda) = 1] + \mathsf{Adv}^{\mathsf{sext}}_{\mathsf{BARG},\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\mathsf{sind}}_{\mathsf{SEC},\mathcal{B}_3}(\lambda).$$

$\mathsf{Hyb}_{\mathcal{A}}^{i,1}(\lambda)$:	$\mathsf{Hyb}_{\mathcal{A}}^{i,2}(\lambda)$:	$\mathsf{Hyb}_{\mathcal{A}}^{i,3}(\lambda)$:
$(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{BARG.TdSetup}(i)$	$(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{BARG.TdSetup}(i)$	$(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{BARG.TdSetup}(i)$
$(\mathsf{dk}, \mathsf{td}_c) \leftarrow \mathsf{SEC.Setup}(i-1)$	$(\mathsf{dk}, \mathsf{td}_c) \leftarrow \mathsf{SEC.Setup}(i-1)$	$(\mathsf{dk}, \mathsf{td}_c) \leftarrow \mathsf{SEC.Setup}(i-1)$
$\mathsf{ck} \leftarrow \mathsf{FC.Setup}()$	$\mathsf{ck} \leftarrow \mathsf{FC.Setup}()$	$\mathsf{ck} \leftarrow \mathsf{FC.Setup}()$
$(\mathcal{P}^*, \{\mathtt{x}_i^*\}_i, \mathsf{c}^*, \pi_\sigma, \mathsf{c}_w)$	$(\mathcal{P}^*, \{\mathtt{x}_i^*\}_i, \mathsf{c}^*, \pi_\sigma, \mathsf{c}_w)$	$(\mathcal{P}^*, \{\mathtt{x}_i^*\}_i, \mathsf{c}^*, \pi_\sigma, \mathsf{c}_w)$
$\leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{crs}, \mathsf{dk}, \mathsf{ck})$	$\leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{crs}, \mathsf{dk}, \mathsf{ck})$	$\leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{crs}, \mathsf{dk}, \mathsf{ck})$
$\hat{\mathsf{c}}_{i-1} \leftarrow \mathsf{SEC.Ext}(\mathsf{td}_c, \mathsf{c}_w)$	$\bar{\mathtt{w}}_i \leftarrow \mathsf{BARG.Ext}(\mathsf{td}, \mathcal{C}, \pi_\sigma)$	$\bar{\mathtt{w}}_i \leftarrow \mathsf{BARG.Ext}(\mathsf{td}, \mathcal{C}, \pi_\sigma)$
$\mathsf{c}_{i-1} \leftarrow$	$\hat{\mathsf{c}}_{i-1} \leftarrow \mathsf{SEC.Ext}(\mathsf{td}_c, \mathsf{c}_w)$	$\hat{\mathsf{c}}_{i-1} \leftarrow \mathsf{SEC.Ext}(\mathsf{td}_c, \mathsf{c}_w)$
$\mathsf{FC.Com}(\mathsf{ck}, (\boldsymbol{m}_{[1:i-1]}, \mathbf{0}))$	$\mathsf{c}_{i-1} \leftarrow$	$\mathsf{c}_{i-1} \leftarrow$
Output 1 iff:	$\mathsf{FC.Com}(\mathsf{ck}, (\boldsymbol{m}_{[1:i-1]}, \mathbf{0}))$	$\mathsf{FC.Com}(\mathsf{ck}, (\boldsymbol{m}_{[1:i-1]}, \mathbf{0}))$
$\mathsf{WIN}_3 = 1$	**Output 1 iff:**	**Output 1 iff:**
$\wedge\ \hat{\mathsf{c}}_{i-1} = \mathsf{c}_{i-1}$	$\mathsf{WIN}_3 = 1$	$\mathsf{WIN}_3 = 1$
	$\wedge\ \mathcal{C}(\mathtt{x}_i, \bar{\mathtt{w}}_i) = 1$	$\wedge\ \mathcal{C}(\mathtt{x}_i, \bar{\mathtt{w}}_i) = 1$
	$\wedge\ \hat{\mathsf{c}}_{i-1} = \mathsf{c}_{i-1}$	$\wedge\ \hat{\mathsf{c}}_{i-1} = \bar{\mathsf{c}}_{i-1} = \mathsf{c}_{i-1}$

Fig. 12: Games $\mathsf{Hyb}^{i,1}, \mathsf{Hyb}^{i,2}, \mathsf{Hyb}^{i,3}$ for the proof of Lemma 3.

$\mathsf{Hyb}_{\mathcal{A}}^{i,4}(\lambda)$:	$\mathsf{Hyb}_{\mathcal{A}}^{i,5}(\lambda)$:	$\mathsf{Hyb}_{\mathcal{A}}^{i,6}(\lambda)$:
$(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{BARG.TdSetup}(i)$	$(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{BARG.TdSetup}(i)$	$(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{BARG.TdSetup}(i)$
$(\mathsf{dk}, \mathsf{td}_c) \leftarrow \mathsf{SEC.Setup}(i-1)$	$(\mathsf{dk}, \mathsf{td}_c) \leftarrow \mathsf{SEC.TdSetup}(i)$	$(\mathsf{dk}, \mathsf{td}_c) \leftarrow \mathsf{SEC.TdSetup}(i)$
$\mathsf{ck} \leftarrow \mathsf{FC.Setup}()$	$\mathsf{ck} \leftarrow \mathsf{FC.Setup}()$	$\mathsf{ck} \leftarrow \mathsf{FC.Setup}()$
$(\mathcal{P}^*, \{\mathtt{x}_i^*\}_i, \mathsf{c}^*, \pi_\sigma, \mathsf{c}_w)$	$(\mathcal{P}^*, \{\mathtt{x}_i^*\}_i, \mathsf{c}^*, \pi_\sigma, \mathsf{c}_w)$	$(\mathcal{P}^*, \{\mathtt{x}_i^*\}_i, \mathsf{c}^*, \pi_\sigma, \mathsf{c}_w)$
$\leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{crs}, \mathsf{dk}, \mathsf{ck})$	$\leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{crs}, \mathsf{dk}, \mathsf{ck})$	$\leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{crs}, \mathsf{dk}, \mathsf{ck})$
$\bar{\mathtt{w}}_i \leftarrow \mathsf{BARG.Ext}(\mathsf{td}, \mathcal{C}, \pi_\sigma)$	$\bar{\mathtt{w}}_i \leftarrow \mathsf{BARG.Ext}(\mathsf{td}, \mathcal{C}, \pi_\sigma)$	$\bar{\mathtt{w}}_i \leftarrow \mathsf{BARG.Ext}(\mathsf{td}, \mathcal{C}, \pi_\sigma)$
		$\hat{\mathsf{c}}_i \leftarrow \mathsf{SEC.Ext}(\mathsf{td}_c, \mathsf{c}_w)$
$\mathsf{c}_{i-1} \leftarrow$	$\mathsf{c}_{i-1} \leftarrow$	$\mathsf{c}_{i-1} \leftarrow$
$\mathsf{FC.Com}(\mathsf{ck}, (\boldsymbol{m}_{[1:i-1]}, \mathbf{0}))$	$\mathsf{FC.Com}(\mathsf{ck}, (\boldsymbol{m}_{[1:i-1]}, \mathbf{0}))$	$\mathsf{FC.Com}(\mathsf{ck}, (\boldsymbol{m}_{[1:i-1]}, \mathbf{0}))$
Output 1 iff:	**Output 1 iff:**	**Output 1 iff:**
$\mathsf{WIN}_3 = 1$	$\mathsf{WIN}_3 = 1$	$\mathsf{WIN}_3 = 1$
$\wedge\ \mathcal{C}(\mathtt{x}_i, \bar{\mathtt{w}}_i) = 1$	$\wedge\ \mathcal{C}(\mathtt{x}_i, \bar{\mathtt{w}}_i) = 1$	$\wedge\ \mathcal{C}(\mathtt{x}_i, \bar{\mathtt{w}}_i) = 1$
$\wedge\ \bar{\mathsf{c}}_{i-1} = \mathsf{c}_{i-1}$	$\wedge\ \bar{\mathsf{c}}_{i-1} = \mathsf{c}_{i-1}$	$\wedge\ \bar{\mathsf{c}}_{i-1} = \mathsf{c}_{i-1}$
		$\wedge\ \bar{\mathsf{c}}_i = \hat{\mathsf{c}}_i$

Fig. 13: Games $\mathsf{Hyb}^{i,4}, \mathsf{Hyb}^{i,5}, \mathsf{Hyb}^{i,6}$ for the proof of Lemma 3.

$\mathsf{Hyb}^{i,4}$**:** This game is a simplification of $\mathsf{Hyb}^{i,3}$ where we no longer extract from SEC, and therefore, the winning condition $\mathsf{c}_{i-1} = \hat{\mathsf{c}}_{i-1}$ also vanishes. Similarly to the proof for $\mathsf{Hyb}^{1,7}$, it follows that

$$\Pr[\mathsf{Hyb}_{\mathcal{A}}^{i,3}(\lambda) = 1] \leq \Pr[\mathsf{Hyb}_{\mathcal{A}}^{i,4}(\lambda) = 1].$$

$\mathsf{Hyb}^{i,5}, \mathsf{Hyb}^{i,6}$**:** These steps are nearly identical to $\mathsf{Hyb}^{1,3}$ and $\mathsf{Hyb}^{1,4}$, where we switch the extraction index of SEC (to index i), and then use the corresponding SEC extractor. By the setup indistinguishability and somewhere extractability of SEC, it follows that we can construct adversaries

$\mathcal{B}_3$ and $\mathcal{B}_4$ such that

$$\Pr[\mathsf{Hyb}_{\mathcal{A}}^{i,4}(\lambda) = 1] \leq \Pr[\mathsf{Hyb}_{\mathcal{A}}^{i,6}(\lambda) = 1] + \mathsf{Adv}_{\mathsf{SEC},\mathcal{B}_3}^{\mathrm{sind}}(\lambda) + \mathsf{Adv}_{\mathsf{SEC},\mathcal{B}_4}^{\mathrm{sext}}(\lambda).$$

$\mathsf{Hyb}_{\mathcal{A}}^{i,7}(\lambda)$:	$\mathsf{Hyb}_{\mathcal{A}}^{i,8}(\lambda)$:	$\mathsf{Hyb}_{\mathcal{A}}^{i,9}(\lambda)$:
$(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{BARG.TdSetup}(i)$	$(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{BARG.TdSetup}(i)$	$(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{BARG.TdSetup}(i)$
$(\mathsf{dk}, \mathsf{td}_c) \leftarrow \mathsf{SEC.TdSetup}(i)$	$(\mathsf{dk}, \mathsf{td}_c) \leftarrow \mathsf{SEC.TdSetup}(i)$	$(\mathsf{dk}, \mathsf{td}_c) \leftarrow \mathsf{SEC.TdSetup}(i)$
$\mathsf{ck} \leftarrow \mathsf{FC.Setup}()$	$\mathsf{ck} \leftarrow \mathsf{FC.Setup}()$	$\mathsf{ck} \leftarrow \mathsf{FC.Setup}()$
$(\mathcal{P}^*, \{\mathtt{x}_i^*\}_i, \mathsf{c}^*, \pi_\sigma, \mathsf{c}_w)$	$(\mathcal{P}^*, \{\mathtt{x}_i^*\}_i, \mathsf{c}^*, \pi_\sigma, \mathsf{c}_w)$	$(\mathcal{P}^*, \{\mathtt{x}_i^*\}_i, \mathsf{c}^*, \pi_\sigma, \mathsf{c}_w)$
$\leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{crs}, \mathsf{dk}, \mathsf{ck})$	$\leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{crs}, \mathsf{dk}, \mathsf{ck})$	$\leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{crs}, \mathsf{dk}, \mathsf{ck})$
$\bar{\mathtt{w}}_i \leftarrow \mathsf{BARG.Ext}(\mathsf{td}, \mathcal{C}, \pi_\sigma)$	$\bar{\mathtt{w}}_i \leftarrow \mathsf{BARG.Ext}(\mathsf{td}, \mathcal{C}, \pi_\sigma)$	
$\hat{\mathsf{c}}_i \leftarrow \mathsf{SEC.Ext}(\mathsf{td}_c, \mathsf{c}_w)$	$\hat{\mathsf{c}}_i \leftarrow \mathsf{SEC.Ext}(\mathsf{td}_c, \mathsf{c}_w)$	$\hat{\mathsf{c}}_i \leftarrow \mathsf{SEC.Ext}(\mathsf{td}_c, \mathsf{c}_w)$
$\mathsf{c}_{i-1} \leftarrow$	$\mathsf{c}_{i-1} \leftarrow$	
$\mathsf{FC.Com}(\mathsf{ck}, (\boldsymbol{m}_{[1:i-1]}, \mathbf{0}))$	$\mathsf{FC.Com}(\mathsf{ck}, (\boldsymbol{m}_{[1:i-1]}, \mathbf{0}))$	$\mathsf{c}_i \leftarrow \mathsf{FC.Com}(\mathsf{ck}, (\boldsymbol{m}_{[1:i]}, \mathbf{0}))$
Output 1 iff:	$\mathsf{c}_i \leftarrow \mathsf{FC.Com}(\mathsf{ck}, (\boldsymbol{m}_{[1:i]}, \mathbf{0}))$	**Output 1 iff:**
$\mathsf{WIN}_3 = 1$	**Output 1 iff:**	$\mathsf{WIN}_3 = 1$
$\wedge\ \mathcal{C}(\mathtt{x}_i, \bar{\mathtt{w}}_i) = 1$	$\mathsf{WIN}_3 = 1$	$\wedge\ \hat{\mathsf{c}}_i = \mathsf{c}_i$
$\wedge\ \bar{\mathsf{c}}_{i-1} = \mathsf{c}_{i-1}$	$\wedge\ \mathcal{C}(\mathtt{x}_i, \bar{\mathtt{w}}_i) = 1$	
$\wedge\ \bar{\mathsf{c}}_i = \hat{\mathsf{c}}_i$	$\wedge\ \bar{\mathsf{c}}_{i-1} = \mathsf{c}_{i-1}$	
$\wedge\ \bar{m}_i = m_i$	$\wedge\ \bar{\mathsf{c}}_i = \hat{\mathsf{c}}_i = \mathsf{c}_i$	
	$\wedge\ \bar{m}_i = m_i$	

Fig. 14: Games $\mathsf{Hyb}^{i,7}, \mathsf{Hyb}^{i,8}, \mathsf{Hyb}^{i,9}$ for the proof of Lemma 3.

$\mathsf{Hyb}^{i,7}$**:** This game transition is as for $\mathsf{Hyb}^{1,5}$, where we rely on the unforgeability of Σ. The only difference is that in the reduction we must replace vk_1 by vk_i in the abort condition defined in the guessing argument. In an analog manner, it follows that

$$\Pr[\mathsf{Hyb}_{\mathcal{A}}^{i,6}(\lambda) = 1] \leq \Pr[\mathsf{Hyb}_{\mathcal{A}}^{i,7}(\lambda) = 1] + Q \cdot \mathsf{Adv}_{\Sigma,\mathcal{B}_5}^{\mathrm{eufcma}}(\lambda).$$

$\mathsf{Hyb}^{i,8}$**,** $\mathsf{Hyb}^{i,9}$**:** These steps are identical to those for $\mathsf{Hyb}^{1,6}$ and $\mathsf{Hyb}^{1,7}$, respectively. We have that, by FC updatability soundness, and by the simplification of the winning condition, there exists an adversary $\mathcal{B}_6$ such that

$$\Pr[\mathsf{Hyb}_{\mathcal{A}}^{1,7}(\lambda) = 1] \leq \Pr[\mathsf{Hyb}_{\mathcal{A}}^{1,9}(\lambda) = 1] + \mathsf{Adv}_{\mathsf{FC},\mathcal{B}_6}^{\mathrm{upbind}}(\lambda).$$

Games $\mathsf{Hyb}^{n,j}$: Games $\mathsf{Hyb}^{n,1}$ to $\mathsf{Hyb}^{n,8}$ are defined as the games $\mathsf{Hyb}^{i,1}$ to $\mathsf{Hyb}^{i,8}$, for $i = n$, in Figures 12, 13, 14, and the reduction steps are identical for these cases. To analyze the advantage of the adversary in $\mathsf{Hyb}^{n,8}$, we introduce an additional $\mathsf{Hyb}^{n,8*}$, that we compare to the former in Figure 15.

Observe that $\mathsf{Hyb}^{n,8*}$ is just a simplification of game $\mathsf{Hyb}^{n,8}$ with an easier winning condition. Hence,

$$\Pr[\mathsf{Hyb}_{\mathcal{A}}^{n,8}(\lambda) = 1] \leq \Pr[\mathsf{Hyb}_{\mathcal{A}}^{n,8*}(\lambda) = 1]$$

$\mathsf{Hyb}^{n,8}_{\mathcal{A}}(\lambda)$:	$\mathsf{Hyb}^{n,8*}_{\mathcal{A}}(\lambda)$:
$(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{BARG.TdSetup}(n)$	$(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{BARG.TdSetup}(n)$
$(\mathsf{dk}, \mathsf{td}_c) \leftarrow \mathsf{SEC.TdSetup}(n)$	$(\mathsf{dk}, \mathsf{td}_c) \leftarrow \mathsf{SEC.TdSetup}(n)$
$\mathsf{ck} \leftarrow \mathsf{FC.Setup}()$	$\mathsf{ck} \leftarrow \mathsf{FC.Setup}()$
$(\mathcal{P}^*, \{\mathtt{x}_i^*\}_i, \mathsf{c}^*, \pi_\sigma, \mathsf{c}_w)$	$(\mathcal{P}^*, \{\mathtt{x}_i^*\}_i, \mathsf{c}^*, \pi_\sigma, \mathsf{c}_w)$
$\quad \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{crs}, \mathsf{dk}, \mathsf{ck})$	$\quad \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{crs}, \mathsf{dk}, \mathsf{ck})$
$\bar{\mathtt{w}}_n \leftarrow \mathsf{BARG.Ext}(\mathsf{td}, \mathcal{C}, \pi_\sigma)$	$\bar{\mathtt{w}}_n \leftarrow \mathsf{BARG.Ext}(\mathsf{td}, \mathcal{C}, \pi_\sigma)$
$\hat{\mathsf{c}}_n \leftarrow \mathsf{SEC.Ext}(\mathsf{td}_c, \mathsf{c}_w)$	$\hat{\mathsf{c}}_n \leftarrow \mathsf{SEC.Ext}(\mathsf{td}_c, \mathsf{c}_w)$
$\mathsf{c}_{n-1} \leftarrow \mathsf{FC.Com}(\mathsf{ck}, (\boldsymbol{m}_{[1:n-1]}, \mathbf{0}))$	
$\mathsf{c}_n \leftarrow \mathsf{FC.Com}(\mathsf{ck}, (\boldsymbol{m}_{[1:n]}, \mathbf{0}))$	$\mathsf{c}_n \leftarrow \mathsf{FC.Com}(\mathsf{ck}, (\boldsymbol{m}_{[1:n]}, \mathbf{0}))$
Output 1 iff:	**Output 1 iff:**
$\quad \mathsf{WIN}_3 = 1$	$\quad \mathsf{WIN}_3 = 1$
$\quad \wedge\, \mathcal{C}(\mathtt{x}_n, \bar{\mathtt{w}}_n) = 1$	$\quad \wedge\, \mathcal{C}(\mathtt{x}_n, \bar{\mathtt{w}}_n) = 1$
$\quad \wedge\, \bar{\mathsf{c}}_{n-1} = \mathsf{c}_{n-1}$	$\quad \wedge\, \bar{\mathsf{c}}_n = \mathsf{c}_n$
$\quad \wedge\, \bar{\mathsf{c}}_n = \hat{\mathsf{c}}_n = \mathsf{c}_n$	
$\quad \wedge\, \bar{m}_n = m_n$	

Fig. 15: Games $\mathsf{Hyb}^{n,8}, \mathsf{Hyb}^{n,8*}$ for the proof of Lemma 3.

Finally, note that the conditions WIN_3, $\mathcal{C}(\mathtt{x}_n, \bar{\mathtt{w}}_n) = 1$, and $\bar{\mathsf{c}}_n = \mathsf{c}_n$ cannot occur simultaneously. The circuit $\mathcal{C}(\mathtt{x}_n, \bar{\mathtt{w}}_n) = 1$ (Figure 5) checks, for $i = n$, that $\bar{\mathsf{c}}_n = \mathsf{c}^*$. Therefore, $\mathsf{c}^* = \mathsf{c}_n$ is honestly computed, which contradicts the winning condition WIN_3. We conclude that

$$\Pr[\mathsf{Hyb}^{n,8*}_{\mathcal{A}}(\lambda) = 1] = 0.$$

Proof summary. Putting all the intermediate bounds together, we obtain the following final bound:

$$\begin{aligned}\Pr[\mathsf{UF}_{3\mathcal{A}}(\lambda) = 1] \leq\ & n \cdot \mathsf{Adv}^{\mathrm{sind}}_{\mathsf{BARG},\mathcal{B}_1}(\lambda) + n \cdot \mathsf{Adv}^{\mathrm{sext}}_{\mathsf{BARG},\mathcal{B}_2}(\lambda) \\ & + (2n-1) \cdot \mathsf{Adv}^{\mathrm{sind}}_{\mathsf{SEC},\mathcal{B}_3}(\lambda) + n \cdot \mathsf{Adv}^{\mathrm{sext}}_{\mathsf{SEC},\mathcal{B}_4}(\lambda) \\ & + n \cdot Q \cdot \mathsf{Adv}^{\mathrm{eufcma}}_{\Sigma,\mathcal{B}_5}(\lambda) + n \cdot \mathsf{Adv}^{\mathrm{updbind}}_{\mathsf{FC},\mathcal{B}_6}(\lambda).\end{aligned}$$

Lemma 4. *For any PPT adversary $\mathcal{A}$ making at most $Q = \mathsf{poly}(\lambda)$ queries to the key generation oracle and that can produce a valid forgery in UF_4, there exists a PPT adversary $\mathcal{B}_{\text{EUF-CMA}}$ against the* EUF-CMA *property of the digital signature scheme Σ, such that*

$$\Pr[\mathsf{UF}_{4,\mathcal{A}}(\lambda) = 1] \leq Q \cdot \mathsf{Adv}^{\text{euf-cma}}_{\Sigma,\mathcal{B}_{\text{EUF-CMA}}}(\lambda).$$

Proof. The proof of this lemma follows virtually the same reduction strategy to unforgeability as in the previous lemmas (the proof of the last hybrid in Lemma 1 and the proofs for hybrids $\mathsf{Hyb}^{1,4} \approx \mathsf{Hyb}^{1,5}$ in Lemma 3). Therefore we only give a sketch to highlight the main differences. The reduction starts by making a guess (which is correct with probability $1/Q$) about the index of the key generation query that gives the verification key that will be used in the forgery. If the guess is correct, a MKHS forgery (ℓ^*, y^*, σ^*) gives a signature on the message $y^*|\ell^*$. If the MKHS forgery is of type 1, then the message is new since no message with suffix ℓ^* was asked to the signing oracle (as in Lemma 1). If instead it is a MKHS forgery of type 2 then the message is new since the signing oracle was queried on $m|\ell^*$ for $m \neq y^*$, and on no other message with label ℓ^* due to the rule of the MKHS security game (as in Lemma 3).

5 Extensions and Instantiations

In this section, we extend our base MKHS construction to support sequential multi-hop evaluation. Later, we describe a variety of instantiations of MKHS from falsifiable (and standard) assumptions, obtained through BARGs, FCs and SECs introduced in previous works.

Before, we present a generic result that allows one to construct a chainable functional commitment [BCFL23] from any (suitably expressive) FC.[7] This transformation turns out useful both for achieving multi-hop evaluation and for instantiations. The idea is simple: for a committed $\boldsymbol{x}$, instead of opening to $\boldsymbol{y} = f(\boldsymbol{x})$ we open to $\mathsf{c}_y = \mathsf{FC.Com}(\mathsf{ck}, f(\boldsymbol{x}))$, which can be expressed as $\mathsf{c}_y = f'(\boldsymbol{x})$ for a function $f' = g \odot f$ (i.e., the sequential composition of f followed by g), where $g(\cdot)$ is the circuit that computes the commitment algorithm $\mathsf{FC.Com}(\mathsf{ck}, \cdot\,)$.

Theorem 4. *Let* FC *be a functional commitment scheme for a class of circuits $\mathcal{F}$ and whose commitment algorithm* FC.Com *can be computed by a circuit $g \in \mathcal{F}$. Then there exists a CFC scheme* CFC *for the class of circuits $\mathcal{F}' = \{f : g \odot f \in \mathcal{F}\}$.*

We remark that, by applying the generic CFC-to-FC transformation introduced by Balbás, Catalano, Fiore and Lai [BCFL23] for the special case of layered circuits, this result boosts any FC for bounded-depth circuits into a FC′ for *unbounded-depth* circuits, albeit the proof size of FC′ grows linearly with the circuit depth. We refer to Appendix A.2 for the proof and further details.

5.1 Multi-Hop Evaluation

We show how to adapt our MKHS construction in Figure 4 to support multi-hop evaluation of sequential functions $f^{(h)}(f^{(h-1)}(\cdots f^{(1)}(\cdot)))$. This construction relies on the same primitives as before, except that we require a *chainable* functional commitment CFC instead of a FC. We remark that, by applying Theorem 4, we can generically turn any FC for circuits into a CFC for circuits. The scheme supports the same labels and messages as the single-hop scheme.

First, we define a param structure for the input taken by the Eval algorithm with function f, such that we can distinguish whether f does a first-hop evaluation, or whether we compute over a previous output of Eval.

$$\mathsf{param} = \begin{cases} (\ell_i, \mathsf{vk}_i, m_i, \sigma_i)_{i\in[n]} & \text{if } h = 1 \\ (\mathcal{P}, \{\mathsf{vk}_i\}_{i\in\mathcal{P}}, \boldsymbol{m}^{(h-1)}, \sigma) & \text{if } h > 1 \end{cases}.$$

We introduce the scheme in Figure 16. The Setup, KeyGen, and Sign algorithms remain as in Figure 4. For security, note that the signature $\sigma_{f,y}$ is as in our single-hop MKHS, except that it includes multiple CFC commitments and opening proofs $(\pi_f^{(j)}, \mathsf{c}^{(j-1)})_{j\in[h]}$. The key observation

[7] Precisely, we can build a CFC supporting a single input commitment; this is however enough in our application of composable MKHS.

is that we can see $\bar{\pi}_f := (\pi_f^{(j)}, \mathsf{c}^{(j-1)})_{j\in[h]}$ as the opening proof for $f^{(h)} \odot \cdots \odot f^{(1)}$ on $\mathsf{c}^{(0)}$ in the generic CFC-to-FC construction from [BCFL23, Theorem 2]. Hence, from the security standpoint we can interpret the multi-hop scheme as our single-hop one instantiated with a different FC; thus the same security proof applies.

$\underline{\mathsf{MKHS.Eval}(\mathsf{pp}, f := f^{(h)}, \mathsf{param}, h)}$:

If $h = 1$:

- Parse $\mathsf{param} := (\ell_i, \mathsf{vk}_i, m_i, \sigma_i)_{i\in[n]}$.
- Output $(\mathsf{c}_w, \pi_\sigma, \pi_f^{(1)}, \mathsf{c}^{(0)}) \leftarrow \mathsf{MKHS.Eval}_0(\mathsf{pp}, f^{(1)}, (\ell_i, \mathsf{vk}_i, m_i, \sigma_i)_{i\in[n]})$.

If $h > 1$:

- Parse $\mathsf{param} := (\mathcal{P}, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}, \boldsymbol{m}^{(h-1)}, \sigma)$.
- Parse $\sigma := (\mathsf{c}_w, \pi_\sigma, (\pi_f^{(j)}, \mathsf{c}^{(j-1)})_{j\in[h-1]})$.
- Parse $\mathsf{pp} := (\mathsf{crs}, \mathsf{ck}, \mathsf{dk})$.
- Compute $\boldsymbol{m}^{(h)} \leftarrow f^{(h)}(\boldsymbol{m}^{(h-1)})$.
- $\mathsf{c}^{(h-1)} \leftarrow \mathsf{CFC.Com}(\mathsf{ck}, \boldsymbol{m}^{(h-1)})$.
- $\pi_f^{(h)} \leftarrow \mathsf{CFC.Open}(\mathsf{ck}, \boldsymbol{m}^{(h-1)}, f^{(h)})$.
- Output $\sigma_{f,y} = (\mathsf{c}_w, \pi_\sigma, (\pi_f^{(j)}, \mathsf{c}^{(j-1)})_{j\in[h]})$.

$\underline{\mathsf{MKHS.Ver}(\mathsf{pp}, \mathcal{P}, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}, \boldsymbol{y}, \sigma_{f,y})}$:

- Parse $\mathcal{P} := (f, \ell_1, \ldots, \ell_n)$ and $\{\ell_i := (\mathsf{id}_i, \tau_i)\}$.
- If $\mathcal{P} = (f_{id}, \ell_1)$ then check that $\Sigma.\mathsf{Ver}(\mathsf{vk}_{\mathsf{id}_1}, \boldsymbol{y}|\ell_1, \sigma_{f,y}) = 1$.
- Else, parse $\sigma_{f,y} := (\mathsf{c}_w, \pi_\sigma, (\pi_f^{(j)}, \mathsf{c}^{(j-1)})_{j\in[h]})$.
- Parse $\mathsf{pp} := (\mathsf{crs}, \mathsf{ck}, \mathsf{dk})$.
- Parse $f := (f^{(1)}, \ldots, f^{(h)})$.
- Compute $\mathsf{c}^{(h)} \leftarrow \mathsf{CFC.Com}(\mathsf{ck}, \boldsymbol{y})$.
- Compute $\mathsf{c}_0 \leftarrow \mathsf{FC.Com}(\mathsf{ck}, \mathbf{0})$.
- Compute the BARG circuit $\mathcal{C}$ described in Figure 5, hardcoding $\mathsf{c}^{(0)}, \mathsf{c}_w, \mathsf{c}_0$.
- Given $\{\mathsf{vk}_i\}_i := \{\mathsf{vk}_{\mathsf{id}_i}\}_i$ and $\{\ell_i\}_i$, $\{\mathsf{ck}_i\}_i$, $\{\mathsf{dk}_i\}_i$, define $\mathtt{x}_i = (\mathsf{vk}_i, \mathsf{ck}_i, \mathsf{dk}_i, \mathsf{dk}_{i-1}, \ell_i, i)$.
- $\forall\, j \in [h]$, check that $\mathsf{CFC.Ver}(\mathsf{ck}, \mathsf{c}^{(j-1)}, f^{(j)}, \mathsf{c}^{(j)}, \pi_f^{(j)}) = 1$.
- Check that $\mathsf{BARG.Ver}(\mathsf{crs}, \mathcal{C}, \{\mathtt{x}_i\}_i, \pi_\sigma) = 1$.
- Output 1 if all checks pass.

Fig. 16: MKHS.Eval and MKHS.Ver algorithms of a multi-hop succinct multi-key homomorphic signature scheme MKHS constructed from a chainable functional commitment CFC, a BARG for NP BARG, a somewhere extractable commitment SEC and a digital signature Σ. Eval_0 is the single-hop Eval from Figure 4.

5.2 Instantiations of MKHS for all functions

We describe several instantiations for our construction in Section 4 that we obtain by instantiating its main building blocks. We focus on MKHS for *all functions*, that we model as either boolean or arithmetic *circuits of unbounded depth*. We discuss the properties of the resulting schemes, in particular their succinctness and the underlying assumptions.

We give two families of MKHS instantiations: those that use non-algebraic FCs and BARGs (internally relying on correlation-intractable hash functions (CIHs) and probabilistic checkable proofs (PCPs)), and those that use algebraic constructions of these schemes. CIH + PCP based constructions offer nearly optimal asymptotic succinctness, but the concrete parameters suffer from an impractical blow-up. Algebraic BARGs and FCs have smaller concrete parameters, and although our MKHS construction makes non-black-box use of them, we believe that instantiations based on algebraic building blocks present a more promising avenue towards fully-algebraic, concretely-efficient future MKHS constructions.

MKHS for unbounded-depth circuits from CIH and PCPs. The natural choices for this family of BARGs are the constructions in [CGJ+23, CJJ22] from either subexponential DDH or LWE, respectively.[8] Their efficiency is later refined in [KLVW23].

For functional commitments, the asymptotically optimal choice is to extend the SNARG for RAM computations from [KLVW23], which can be seen as an FC for single-output boolean circuits $\mathcal{C} : \{0,1\}^n \to \{0,1\}$. Such an FC can be constructed generically from BARGs, and hence from the same assumptions as before. Extending their SNARG to a fully-fledged FC for unbounded depth-circuits is not straightforward and requires a series of observations:

- The commitment scheme underlying their SNARG is deterministic and supports efficient local updatability as it is implemented as a Merkle tree.
- Their SNARG satisfies FC evaluation binding for RAM computations with a bounded number of steps, which can be represented by single-output boolean circuits $f : \{0,1\}^n \to \{0,1\}$ of bounded depth $d_{\max}$. To boost their scheme, we can apply our generic transformation to obtain a CFC from any FC (Theorem 4). Since committing to a Merkle tree can be carried out by a circuit of $\mathsf{poly}(\lambda, \log n)$ depth, the transformation yields a CFC for boolean circuits $f : \{0,1\}^n \to \{0,1\}^n$ of bounded depth $d'_{\max} \lesssim d_{\max}$, where the opening proofs have size $|\pi_f| = \mathsf{poly}(\lambda, \log n)$.
- Given such a CFC for boolean circuits of bounded depth, one can obtain a FC for circuits of unbounded depth d by applying the generic CFC-to-FC transformation from [BCFL23], which imposes a multiplicative overhead of d on the opening size. Overall, $|\pi_f| = \mathsf{poly}(\lambda, \log n) \cdot d$.

Corollary 2. *Assuming the hardness of either (1) subexponential decisional Diffie-Hellman (DDH), or (2) learning with errors, there exists a multi-key homomorphic signature for unbounded-depth boolean circuits $\mathcal{F} = \{f : \{0,1\}^n \to \{0,1\}^m\}$ with the following properties:*

- ***Public parameters size:*** $|\mathsf{pp}| = \mathsf{poly}(\lambda, \log n)$.
- ***Signature size:*** $|\sigma_{f,y}| = \mathsf{poly}(\lambda, \log n, \log m) \cdot d$.
- ***Efficient verification:*** *Both the labels and the function can be preprocessed. The online efficient verification algorithm runs in time* $\mathsf{poly}(\lambda, \log n, m) \cdot d$.
- ***Multi-hop evaluation*** *and* ***Context-hiding****.*

MKHS for unbounded-depth circuit from algebraic schemes. Our MKHS can be instantiated over bilinear groups by using the algebraic BARG from [WW22], which relies either on the subgroup decision assumption or on the k-Lin assumption for any $k \geq 2$. In [WW22], they also present companion constructions of somewhere extractable commitments from the same assumptions. For the FC, the most natural pairing-based choice is the algebraic scheme from [BCFL23], which relies on the HiKer assumption. Notably, this FC scheme admits efficient local updatability, deterministic commitments, and supports efficient verification with preprocessing.

[8] In the same works, SECs are constructed from the same assumptions as a building block for BARGs.

Corollary 3. *Assuming the hardness of HiKer and either the subgroup decision assumption or k-Lin for $k \geq 2$, there exists a pairing-based MKHS for unbounded-depth arithmetic circuits $\mathcal{F} = \{f : \mathcal{M}^n \to \mathcal{M}^m\}$ of bounded width w with the following properties:*

- ***Public parameters size:*** $|\mathsf{pp}| = \mathcal{O}(w^5)$
- ***Signature size:*** $|\sigma_{f,y}| = \mathcal{O}(\lambda \cdot d^2) + \mathsf{poly}(\lambda)$*. In particular, the signature is fully succinct on both n and m.*
- ***Efficient verification:*** *Both the labels and the function can be preprocessed. The online efficient verification algorithm runs in time* $\mathcal{O}(\lambda \cdot d^2) + \mathsf{poly}(\lambda)$*.*
- ***Multi-hop evaluation*** *and* ***Context-hiding****.*

Thanks to a recent result by Wee and Wu [WW24], one may also replace the HiKer assumption by bilateral k-Lin. Towards a lattice-based algebraic instatiation, we remark that no lattice-based algebraic BARGs exist up to date. For FCs, a natural choice may be the lattice-based (C)FC in [BCFL23], or the scheme that results after applying the transformation of Theorem 4 to the FC in [WW23a].

Acknowledgements

This work is supported by the PICOCRYPT project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 101001283), partially supported by projects PRODIGY (TED2021-132464B-I00) and ESPADA (PID2022-142290OB-I00) funded by MCIN/AEI/10.13039/501100011033/ and the European Union NextGenerationEU / PRTR, and partially funded by Ministerio de Universidades (FPU21/00600).

References

AL11. Nuttapong Attrapadung and Benoît Libert. Homomorphic network coding signatures in the standard model. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011: 14th International Conference on Theory and Practice of Public Key Cryptography*, volume 6571 of *Lecture Notes in Computer Science*, pages 17–34. Springer, Heidelberg, March 2011.

AP19. Diego F. Aranha and Elena Pagnin. The simplest multi-key linearly homomorphic signature scheme. In Peter Schwabe and Nicolas Thériault, editors, *Progress in Cryptology - LATINCRYPT 2019: 6th International Conference on Cryptology and Information Security in Latin America*, volume 11774 of *Lecture Notes in Computer Science*, pages 280–300. Springer, Heidelberg, October 2019.

BCFL23. David Balbás, Dario Catalano, Dario Fiore, and Russell W. F. Lai. Chainable functional commitments for unbounded-depth circuits. In *Theory of Cryptography*, Cham, 2023. Springer Nature Switzerland.

BCJP24. Maya Farber Brodsky, Arka Rai Choudhuri, Abhishek Jain, and Omer Paneth. Monotone-policy aggregate signatures. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024*, pages 168–195, Cham, 2024. Springer Nature Switzerland.

BF11. Dan Boneh and David Mandell Freeman. Homomorphic signatures for polynomial functions. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 149–168. Springer, Heidelberg, May 2011.

BFKW09. Dan Boneh, David Freeman, Jonathan Katz, and Brent Waters. Signing a linear subspace: Signature schemes for network coding. In Stanislaw Jarecki and Gene Tsudik, editors, *PKC 2009: 12th International Conference on Theory and Practice of Public Key Cryptography*, volume 5443 of *Lecture Notes in Computer Science*, pages 68–87. Springer, Heidelberg, March 2009.

BGLS03. Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer, Heidelberg, May 2003.

CF13. Dario Catalano and Dario Fiore. Vector commitments and their applications. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013: 16th International Conference on Theory and Practice of Public Key Cryptography*, volume 7778 of *Lecture Notes in Computer Science*, pages 55–72. Springer, Heidelberg, February / March 2013.

CFGV13. Dario Catalano, Dario Fiore, Rosario Gennaro, and Konstantinos Vamvourellis. Algebraic (trapdoor) one-way functions and their applications. In Amit Sahai, editor, *TCC 2013: 10th Theory of Cryptography Conference*, volume 7785 of *Lecture Notes in Computer Science*, pages 680–699. Springer, Heidelberg, March 2013.

CFN15. Dario Catalano, Dario Fiore, and Luca Nizzardo. Programmable hash functions go private: Constructions and applications to (homomorphic) signatures with shorter public keys. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 254–274. Springer, Heidelberg, August 2015.

CFT22. Dario Catalano, Dario Fiore, and Ida Tucker. Additive-homomorphic functional commitments and applications to homomorphic signatures. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology – ASIACRYPT 2022, Part IV*, volume 13794 of *Lecture Notes in Computer Science*, pages 159–188. Springer, Heidelberg, December 2022.

CFW12. Dario Catalano, Dario Fiore, and Bogdan Warinschi. Efficient network coding signatures in the standard model. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012: 15th International Conference on Theory and Practice of Public Key Cryptography*, volume 7293 of *Lecture Notes in Computer Science*, pages 680–696. Springer, Heidelberg, May 2012.

CFW14. Dario Catalano, Dario Fiore, and Bogdan Warinschi. Homomorphic signatures with efficient verification for polynomial functions. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 371–389. Springer, Heidelberg, August 2014.

CGJ+23. Arka Rai Choudhuri, Sanjam Garg, Abhishek Jain, Zhengzhong Jin, and Jiaheng Zhang. Correlation intractability and SNARGs from sub-exponential DDH. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part IV*, volume 14084 of *Lecture Notes in Computer Science*, pages 635–668. Springer, Heidelberg, August 2023.

CJJ21. Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. Non-interactive batch arguments for NP from standard assumptions. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part IV*, volume 12828 of *Lecture Notes in Computer Science*, pages 394–423, Virtual Event, August 2021. Springer, Heidelberg.

CJJ22. Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. SNARGs for $\mathcal{P}$ from LWE. In *62nd Annual Symposium on Foundations of Computer Science*, pages 68–79. IEEE Computer Society Press, February 2022.

dCP23. Leo de Castro and Chris Peikert. Functional commitments for all functions, with transparent setup and from SIS. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part III*, volume 14006 of *Lecture Notes in Computer Science*, pages 287–320. Springer, Heidelberg, April 2023.

Des93. Yvo Desmedt. Computer security by redefining what a computer is. NSPW, 1993.

DGKV22. Lalita Devadas, Rishab Goyal, Yael Kalai, and Vinod Vaikuntanathan. Rate-1 non-interactive arguments for batch-NP and applications. In *63rd Annual Symposium on Foundations of Computer Science*, pages 1057–1068. IEEE Computer Society Press, October / November 2022.

FMNP16. Dario Fiore, Aikaterini Mitrokotsa, Luca Nizzardo, and Elena Pagnin. Multi-key homomorphic authenticators. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016, Part II*, volume 10032 of *Lecture Notes in Computer Science*, pages 499–530. Springer, Heidelberg, December 2016.

FN16. Dario Fiore and Anca Nitulescu. On the (in)security of SNARKs in the presence of oracles. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B: 14th Theory of Cryptography Conference, Part I*, volume 9985 of *Lecture Notes in Computer Science*, pages 108–138. Springer, Heidelberg, October / November 2016.

FP18. Dario Fiore and Elena Pagnin. Matrioska: A compiler for multi-key homomorphic signatures. In Dario Catalano and Roberto De Prisco, editors, *SCN 18: 11th International Conference on Security in Communication Networks*, volume 11035 of *Lecture Notes in Computer Science*, pages 43–62. Springer, Heidelberg, September 2018.

Fre12. David Mandell Freeman. Improved security for linearly homomorphic signatures: A generic framework. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012: 15th International Conference* *on Theory and Practice of Public Key Cryptography*, volume 7293 of *Lecture Notes in Computer Science*, pages 697–714. Springer, Heidelberg, May 2012.

Gen09. Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 169–178. ACM Press, May / June 2009.

GKKR10. Rosario Gennaro, Jonathan Katz, Hugo Krawczyk, and Tal Rabin. Secure network coding over the integers. In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010: 13th International Conference on Theory and Practice of Public Key Cryptography*, volume 6056 of *Lecture Notes in Computer Science*, pages 142–160. Springer, Heidelberg, May 2010.

Goy24. Rishab Goyal. Mutable batch arguments and applications. Cryptology ePrint Archive, Paper 2024/737, 2024. https://eprint.iacr.org/2024/737.

GU24. Romain Gay and Bogdan Ursu. On Instantiating Unleveled Fully-Homomorphic Signatures from Falsifiable Assumptions. In *International Conference on Public Key Cryptography - PKC 2024*, 2024. to appear.

GVW15. Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled fully homomorphic signatures from standard lattices. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th Annual ACM Symposium on Theory of Computing*, pages 469–477. ACM Press, June 2015.

GW11. Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd Annual ACM Symposium on Theory of Computing*, pages 99–108. ACM Press, June 2011.

GW13. Rosario Gennaro and Daniel Wichs. Fully homomorphic message authenticators. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013, Part II*, volume 8270 of *Lecture Notes in Computer Science*, pages 301–320. Springer, Heidelberg, December 2013.

GZ21. Alonso González and Alexandros Zacharakis. Fully-succinct publicly verifiable delegation from constant-size assumptions. In Kobbi Nissim and Brent Waters, editors, *TCC 2021: 19th Theory of Cryptography Conference, Part I*, volume 13042 of *Lecture Notes in Computer Science*, pages 529–557. Springer, Heidelberg, November 2021.

HKW15. Susan Hohenberger, Venkata Koppula, and Brent Waters. Universal signature aggregators. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 3–34. Springer, Heidelberg, April 2015.

HW15. Pavel Hubacek and Daniel Wichs. On the communication complexity of secure function evaluation with long output. In Tim Roughgarden, editor, *ITCS 2015: 6th Conference on Innovations in Theoretical Computer Science*, pages 163–172. Association for Computing Machinery, January 2015.

JMSW02. Robert Johnson, David Molnar, Dawn Xiaodong Song, and David Wagner. Homomorphic signature schemes. In Bart Preneel, editor, *Topics in Cryptology – CT-RSA 2002*, volume 2271 of *Lecture Notes in Computer Science*, pages 244–262. Springer, Heidelberg, February 2002.

KLVW23. Yael Kalai, Alex Lombardi, Vinod Vaikuntanathan, and Daniel Wichs. Boosting batch arguments and ram delegation. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing (STOC)*, STOC 2023, page 1545–1552, New York, NY, USA, 2023. Association for Computing Machinery.

KNYY19. Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Designated verifier/prover and preprocessing NIZKs from Diffie-Hellman assumptions. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part II*, volume 11477 of *Lecture Notes in Computer Science*, pages 622–651. Springer, Heidelberg, May 2019.

KPY19. Yael Tauman Kalai, Omer Paneth, and Lisa Yang. How to delegate computations publicly. In Moses Charikar and Edith Cohen, editors, *51st Annual ACM Symposium on Theory of Computing*, pages 1115–1124. ACM Press, June 2019.

KVZ21. Yael Tauman Kalai, Vinod Vaikuntanathan, and Rachel Yun Zhang. Somewhere statistical soundness, post-quantum security, and SNARGs. In Kobbi Nissim and Brent Waters, editors, *TCC 2021: 19th Theory of Cryptography Conference, Part I*, volume 13042 of *Lecture Notes in Computer Science*, pages 330–368. Springer, Heidelberg, November 2021.

LPJY13. Benoît Libert, Thomas Peters, Marc Joye, and Moti Yung. Linearly homomorphic structure-preserving signatures and their applications. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 289–307. Springer, Heidelberg, August 2013.

LRY16. Benoît Libert, Somindu C. Ramanna, and Moti Yung. Functional commitment schemes: From polynomial commitments to pairing-based accumulators from simple assumptions. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *ICALP 2016: 43rd International Colloquium on Automata, Languages and Programming*, volume 55 of *LIPIcs*, pages 30:1–30:14. Schloss Dagstuhl, July 2016.

LTWC18. Russell W. F. Lai, Raymond K. H. Tai, Harry W. H. Wong, and Sherman S. M. Chow. Multi-key homomorphic signatures unforgeable under insider corruption. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018, Part II*, volume 11273 of *Lecture Notes in Computer Science*, pages 465–492. Springer, Heidelberg, December 2018.

NWW23. Shafik Nassar, Brent Waters, and David J. Wu. Monotone policy bargs from bargs and additively homomorphic encryption. Cryptology ePrint Archive, Paper 2023/1967, 2023. https://eprint.iacr.org/2023/1967.

OPWW15. Tatsuaki Okamoto, Krzysztof Pietrzak, Brent Waters, and Daniel Wichs. New realizations of somewhere statistically binding hashing and positional accumulators. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015, Part I*, volume 9452 of *Lecture Notes in Computer Science*, pages 121–145. Springer, Heidelberg, November / December 2015.

RS09. Markus Rückert and Dominique Schröder. Aggregate and verifiably encrypted signatures from multilinear maps without random oracles. In Jong Hyuk Park, Hsiao-Hwa Chen, Mohammed Atiquzzaman, Changhoon Lee, Tai-hoon Kim, and Sang-Soo Yeo, editors, *Advances in Information Security and Assurance*, pages 750–759, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

SBB19. Lucas Schabhüser, Denis Butin, and Johannes Buchmann. Context hiding multi-key linearly homomorphic authenticators. In Mitsuru Matsui, editor, *Topics in Cryptology – CT-RSA 2019*, volume 11405 of *Lecture Notes in Computer Science*, pages 493–513. Springer, Heidelberg, March 2019.

SFVA21. Somayeh Dolatnezhad Samarin, Dario Fiore, Daniele Venturi, and Morteza Amini. A compiler for multi-key homomorphic signatures for turing machines. *Theor. Comput. Sci.*, 889:145–170, 2021.

WW22. Brent Waters and David J. Wu. Batch arguments for sfNP and more from standard bilinear group assumptions. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part II*, volume 13508 of *Lecture Notes in Computer Science*, pages 433–463. Springer, Heidelberg, August 2022.

WW23a. Hoeteck Wee and David J. Wu. Lattice-based functional commitments: Fast verification and cryptanalysis. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023, Part V*, volume 14442 of *Lecture Notes in Computer Science*, pages 201–235. Springer, Heidelberg, December 2023.

WW23b. Hoeteck Wee and David J. Wu. Succinct vector, polynomial, and functional commitments from lattices. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part III*, volume 14006 of *Lecture Notes in Computer Science*, pages 385–416. Springer, Heidelberg, April 2023.

WW24. Hoeteck Wee and David J. Wu. Succinct Functional Commitments for Circuits from k-Lin. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024*, pages 280–310, Cham, 2024. Springer Nature Switzerland.

A On Generic Transformations for Functional Commitments

A.1 Theorem 1: from Succinct to Compact FC and (MK)HS

We show a generic method to turn an input-succinct FC into one that is also output-succinct. The same transformation applies to (multi-key) homomorphic signatures (the adaptation is straightforward and omitted here). The idea is to execute FC on the composed function $f_H := H \odot f : \mathcal{M}^n \to \mathcal{M}^\ell$ in order to generate an opening for each of the ℓ output values of f_H. The verifier who knows the output $\boldsymbol{y} \in \mathcal{M}^m$ runs the FC verification with the function f_H and the output $H(\boldsymbol{y})$. Precisely, since we assume that FC supports only n-to-1 functions, we would consider an instantiation for n-to-ℓ functions obtained by running the opening and verification algorithms ℓ times, for the functions $\{f_{H,i}\}_{i=1..m}$ that return the i-th output bit of f_H. As one can see, the size of the opening proof of this construction is $\ell \cdot |\pi|$ where $|\pi|$ is the size of an opening in FC.

For this transformation to be correct we need the FC scheme to be sufficiently expressive in order to support the functions $H \odot f$, which may be in a class of functions larger than $\mathcal{F}$. For example, for FCs that support circuits of bounded depth one needs to increase the bound by $d_H(m)$ (i.e., the depth of H on inputs of length m). Technically, we need that each projection $f_{H,j}$, for $j = 1$ to ℓ, is in the class $\mathcal{F}$ supported by FC.

The security of this transformation relies on the evaluation binding of FC and the collision resistance of H. A proof sketch follows. Consider any adversary breaking evaluation binding of FC'. Recall that this means that we have two valid openings for $\boldsymbol{y}$ and $\boldsymbol{y}' \neq \boldsymbol{y}$. Then there are two possible cases: $H(\boldsymbol{y}) = H(\boldsymbol{y}')$ or not. In the former case we can break collision resistance of H. In the second case, there is at least an index j such that $H(\boldsymbol{y})_j \neq H(\boldsymbol{y}')_j$ and there are two valid proofs for these values w.r.t. the same function $f_{H,j}$. This case can be reduced to the evaluation binding of FC.

An interesting special case. Interestingly, the idea of this transformation can be applied even to very limited FCs, such as ones for linear maps, by means of linear hash functions such as Ajtai's. In turn, this method can be applied to existing functional commitments from lattices [dCP23, WW23b, WW23a] to obtain output-succinctness efficiently.

Let FC be an FC for n-to-1 linear forms over a ring $\mathbb{Z}_q$. Precisely, let $\mathcal{F}$ be a set of functions $\mathcal{F} = \{f : \mathbb{Z}_q^n \to \mathbb{Z}_q^m\}$ where outputs are small integers bounded (in absolute value) by some $\beta < q$. Consider Ajtai's hash function $H_{\mathbf{A}} : \mathbb{Z}^m \to \mathbb{Z}_q^\ell$ defined by $H_{\mathbf{A}}(\boldsymbol{y}) := \mathbf{A} \cdot \boldsymbol{y} \mod q$ for $\mathbf{A} \in \mathbb{Z}_q^{m \times \ell}$, which is collision-resistant for vectors of small norm. For any $f \in \mathcal{F}$ define $f_H := H_{\mathbf{A}} \odot f$, i.e., $f_H(\boldsymbol{x}) := \mathbf{A} \cdot f(\boldsymbol{x})$. Notice that $f_H : \mathbb{Z}_q^n \to \mathbb{Z}_q^\ell$ is a linear map and thus we can run FC for linear forms ℓ times, one for every output.

A.2 Theorem A.2: From FCs to Chainable FCs

To obtain a chainable FC scheme CFC from an FC scheme FC, we define CFC as follows:

- $\mathsf{CFC.Setup}(1^\lambda, 1^n) = \mathsf{FC.Setup}(1^\lambda, 1^n)$
- $\mathsf{CFC.Com}(\mathsf{ck}, \boldsymbol{x}) = \mathsf{FC.Com}(\mathsf{ck}, \boldsymbol{x})$
- $\mathsf{CFC.Open}(\mathsf{ck}, \mathsf{aux}, f) = \mathsf{FC.Open}(\mathsf{ck}, \mathsf{aux}, g \odot f)$
- $\mathsf{CFC.Ver}(\mathsf{ck}, \mathsf{c}_x, \mathsf{c}_y, f, \pi) = \mathsf{FC.Ver}(\mathsf{ck}, \mathsf{c}_x, \mathsf{c}_y, g \odot f, \pi)$.

Correctness is immediate by construction and by the definition of the class $\mathcal{F}'$.

For evaluation binding, assume by contradiction that an adversary $\mathcal{A}$ outputs a tuple $(\mathsf{c}_x, f, \mathsf{c}_y, \pi, \mathsf{c}'_y, \pi')$ that breaks the evaluation binding of CFC. Then, by construction, the tuple $(\mathsf{c}_x, g \odot f, \mathsf{c}_y, \pi, \mathsf{c}'_y, \pi')$ breaks the evaluation binding of FC.

The CFC scheme has succinctness $s_{\mathsf{CFC}}(\lambda, n, m, |f|) = s_{\mathsf{FC}}(\lambda, n, m, |g \odot f|)$, which by succinctness of FC is $= \mathsf{poly}(\lambda, \log n, \log m, o(|g \odot f|))$. To argue that this yields succinctness, i.e., $s_{\mathsf{CFC}}(\lambda, n, m, |f|) = \mathsf{poly}(\lambda, \log n, \log m, o(|f|))$, we need that $|g| = o(|f|)$. Concretely, for the sake of existing FCs it can be enough to assume that g is a circuit of depth $\mathsf{polylog}(n)$.