Multi-Key Homomorphic Signatures Unforgeable under Insider Corruption

(Full Version)

Russell W. F. Lai[1,2], Raymond K. H. Tai[1], Harry W. H. Wong[1], and Sherman S. M. Chow[1]

[1] Chinese University of Hong Kong, Hong Kong
[2] Friedrich-Alexander-Uiversität Erlangen-Nürnberg, Germany
{russell,raymondtai,whwong,sherman}@ie.cuhk.edu.hk

Abstract. Homomorphic signatures (HS) allows the derivation of the signature of the message-function pair (m, g), where $m = g(m_1, \ldots, m_K)$, given the signatures of each of the input messages m_k signed under the same key. Multi-key HS (M-HS) introduced by Fiore et al. (ASIACRYPT'16) further enhances the utility by allowing evaluation of signatures under different keys. While the unforgeability of existing M-HS notions unrealistically assumes that all signers are honest, we consider the setting where an arbitrary number of signers can be corrupted, which is typical in natural applications (*e.g.*, verifiable multi-party computation) of M-HS. Surprisingly, there is a huge gap between M-HS with and without unforgeability under corruption: While the latter can be constructed from standard lattice assumptions (ASIACRYPT'16), we show that the former must rely on non-falsifiable assumptions. Specifically, we propose a generic construction of M-HS with unforgeability under corruption from adaptive zero-knowledge succinct non-interactive arguments of knowledge (ZK-SNARK) (and other standard assumptions), and then show that such M-HS implies adaptive zero-knowledge succinct non-interactive arguments (ZK-SNARG). Our results leave open the pressing question of what level of authenticity can be guaranteed in the multi-key setting under standard assumptions.

1 Introduction

In a basic signature scheme, a signer can use a secret key to sign messages which are verifiable using the corresponding public key. The signatures are required to be unforgeable, meaning that no efficient adversaries can forge a valid signature on any message without the secret key. This requirement, however, limits the utility of the signed message. For example, without the secret key, one cannot derive a signature of a computation over the signed messages.

Homomorphic signature (HS) schemes [32] allow a third-party (public) evaluator to compute any functions from a class of admissible functions over signed messages (from a single signer), and derive signatures of the computation results. HS is a handy tool for applications which require computation on authenticated data. For example, it is useful when computationally inferior data producers (*e.g.*, sensors in Internet-of-Things) need to outsource expensive computations to a third-party (*e.g.*, the cloud), while assuring the authenticity of the computation result. Since homomorphic evaluation of messages and signatures is allowed, the standard unforgeability notion can no longer be satisfied. There are two common meaningful relaxations. The first one is considered for linear homomorphic signatures [11], where only linear functions are admissible. Unforgeability of linear HS requires that any adversary cannot derive a signature of a vector which is not a linear combination of any honestly signed vectors. This relaxation is not suitable for fully homomorphic signatures [15, 51] where all polynomials/circuits are admissible, as signatures for a wide range of messages can often be derived from just a single signed message. Thus, the second approach is to have the signature not only certify the message, but also the function that is used to compute the message. Unforgeability here means that any adversary cannot derive a signature of a message-function pair (m, g), such that m is not a function value of g evaluated over any honestly signed messages. This work considers HS for general functionality, hence we adopt the second approach.

1.1 Multi-Key Homomorphic Signatures

To further extend the utility of HS, multi-key HS (M-HS) has recently received attention [47, 49]. This extension of HS allows homomorphic evaluation of signatures signed under different keys. An evaluated signature is verifiable using a combined public key, *e.g.*, the ordered tuple consisting of all public keys of the signatures being evaluated. The multi-key setting allows multiple data producers, who do not or should not share the same key, to contribute signed data for verifiable computation. Unfortunately, existing work [47,49] only considers weaker security models (see further discussion in Section 2.2), which do not capture insider attacks from malicious contributors. In fact, a malicious signer in [49] is able to create a signature on any message-function pairs (m, g) regardless of the honest signer inputs (see Appendix B). The problem is also inherent in all existing lattice-based signatures with trapdoors.

For certain classes of computation such as the majority vote, if the M-HS scheme is not secure against insider attacks, it might be possible that a compromised signer can manipulate the voting result. This limits the usefulness of existing M-HS solutions since it is often unrealistic to assume that all contributors in a multi-party computation are honest. In view of the limitation, there is a need for a stronger notion which provides unforgeability even in the presence of corrupted signers.

1.2 Our Results

Multi-key homomorphic signatures unforgeable under insider corruption. In Section 4, we revisit the notion of multi-key homomorphic signatures (M-HS). M-HS is a generalization of homomorphic signatures which allows a public evaluator to apply a function g to transform signatures of different messages $(m_1, \ldots, m_K)$ signed under different public keys to a signature of $(g(m_1, \ldots, m_K), g)$ signed under a combined public key. Existing work [47,49] assumes all signers are honest. In contrast, we define a strong security notion of M-HS called *existential unforgeability under corruption and chosen message attack (cEUF-CMA).* The adversary can control a coalition of the evaluator and a set of malicious signers. A signature of (m, g) is a valid forgery if the resulting message m is not in the range of g restricted by the input of the honest signer. Interestingly, cEUF-CMA-security also makes sense in the single-key setting. The adversary can choose to corrupt the only honest signer, where we require that even the signer itself cannot produce a signature on (m, g) where m is not in the range of g.

Relations to existing notions. We study how cEUF-CMA-secure M-HS is related to other notions. First, we show in Section 5 that such M-HS can be constructed from adaptive zero-knowledge succinct non-interactive arguments of knowledge (ZK-SNARK). We note that there are some impossibility results regarding the security of SNARKs in the presence of oracles (O-SNARK) [25]. In particular, there exists a secure signature scheme Σ such that no candidate construction of O-SNARK satisfies adaptive proof of knowledge with respect to the signing oracle of Σ. Fortunately, there are at least two ways to circumvent this impossibility result. The first approach is to use a ZK-SNARK with a "strong" adaptive proof of knowledge property [16, 25], where the extractor takes as input an additional trapdoor and ignores the random tape of the adversary. The second approach is to use an underlying signature scheme for which there exists a secure O-SNARK [25, Section 5]. Either way, by a recursive witness extraction technique, we show that strong ZK-SNARKs implies a "poly-depth" M-HS, while using O-SNARKs yields a "constant-depth" M-HS.

Then, in Section 6.1, we show that succinct functional signatures (FS) [16] can be constructed from a cEUF-CMA-secure two-key M-HS (2-HS). Since the existence of succinct functional signatures implies the existence of succinct non-interactive arguments (SNARG), we obtain as a corollary that the existence of cEUF-CMA-secure 2-HS implies the existence of SNARG.

The above implication is somewhat unsatisfactory as it starts from ZK-SNARK and ends with SNARG (without zero-knowledge). We thus further show in Section 6.2 that the existence of cEUF-CMA-secure single-key HS implies the existence of ZK-SNARG. This makes cEUF-CMA-secure M-HS a notion sitting nicely in between ZK-SNARK and ZK-SNARG, which differ only in the existence of the knowledge extractor.

Since it is known that SNARG cannot be based on falsifiable assumptions [30], it follows that cEUF-CMA-secure M-HS must also be based on non-falsifiable assumptions. This impossibility result puts us into an unfortunate situation where, either we rely on strong assumptions for verifiable multi-party computation, or we settle for weaker authenticity guarantee. We leave open the problem of defining a security model for M-HS, which tolerates a reasonable level of corruption while being realizable with standard assumptions.

Applications. Being such a powerful primitive, cEUF-CMA-secure M-HS implies several extensions of M-HS, namely, (multi-key) delegatable homomorphic signatures (M-DHS) and (multi-key) attribute-message-homomorphic signatures (M-AMHS). As the constructions of these extensions mostly introduce more complicated syntax/functionalities without too much technicality, we only briefly describe them in Section 1.3 but omit the formal details. That said, as a case study, we use similar techniques to show how to construct distributed attribute-based signatures (D-ABS) from cEUF-CMA-secure M-HS (see Appendix A).

1.3 Extensions

We introduce two extensions which are immediate applications of cEUF-CMA-secure M-HS, but seem not to be realizable from non-corruption-resistant M-HS. Here we consider M-HS schemes which support homomorphic evaluation of labeled-data [29] (to be explained in Section 4.1). In a nutshell, such schemes ensure that data with "incompatible" labels cannot be used for computation.

Delegation. The first extension is (multi-key) delegatable homomorphic signatures (M-DHS), which may also be viewed as an extension to append-only signatures [8,33], M-DHS is motivated by the following application scenario. Suppose that multiple data producers engage in a verifiable multi-party computation. Instead of contributing the data individually, these data producers are organized to form groups called delegation chains. Similar to append-only signatures, in each of these chains, the first data producer contributes a template which is passed to each of the data producers along the chain, who fills out some of the entries in the template. The last data producer in each chain then passes the completed template to a third party evaluator, who performs expensive computation over the collection of completed templates.

M-DHS is easily realizable using cEUF-CMA-secure M-HS. To delegate, the delegator simply signs the (partially-filled) template labeled by the public key of the delegatee. By the corruption resistance of the M-HS, a delegatee cannot overwrite the template entries filled out by the delegators up the delegation chain.

Attribute-Homomorphism. Our next extension allows "attribute-homomorphism" on top of the message-homomorphism of (M-)HS. We call it (multi-key) attribute-message-homomorphic signatures (M-AMHS). Consider our running example of verifiable multi-party computation again. M-AMHS is useful when the computation not only depends on the data contributed by the data producers, but also their attributes such as trustworthiness, accuracy, and ranks. To support such a functionality, it is natural to have the authorities issue certificates to the data producers. A certificate is simply a signature of the attribute of the data producer labeled by its public key. The data producer signs its data as in M-HS, except that the evaluator now evaluates functions over both signatures produced by the data producers and the certificates issued by the authorities. By the corruption resistance of the M-HS, it is infeasible for a data producer to fake its attributes.

2 Related Work

2.1 Existing Homomorphic Signatures

Homomorphic signatures (HS) has undergone great development, notably from supporting either only addition or multiplication [9,11,18,27,28,37], to bounded-degree polynomials [10,19], and even to (leveled) fully homomorphic operations which allow evaluation of general circuits of apriori bounded depth [15,51]. Beyond unforgeability, some works also consider privacy notions such as context-hiding [1,3,4].

2.2 Existing Multi-Key Homomorphic Signatures

The study of HS was restricted to the single-key setting until the recent works of Fiore *et al.* [49] and Derler and Slamanig [47], who define multi-key homomorphic signatures with varying level of security. Independent of their work, we initiate the study of multi-key HS in the *insider corruption model.*

Fiore *et al.* [49] propose the notion of multi-key homomorphic authenticators, which is a generalization of M-HS and multi-key homomorphic MAC. They extend the HS by Gorbunov *et al.* [51] to an M-HS based on standard lattice assumptions, and introduce multi-key-homomorphic MAC based on pseudorandom functions.

While the model of Fiore *et al.* allows the adversary to corrupt signers, a forgery is valid only if it passes verification under *non-corrupted keys.* In practice, it means that if any signer involved in the computation is corrupted, the authenticity of the derived result is no longer guaranteed. Indeed, as acknowledged [49], their construction is vulnerable to insider attacks. They claimed that preventing insider attacks is impossible, by arguing that, for general functions, controlling a few inputs implies controlling the function output. We find the claim inaccurate as there is a large class of functions which may not exhibit this property, *e.g.*, functions with AND gates, majority gates, and threshold gates. Our work, in contrast, constructs M-HS which prevent insider attacks, at the cost of stronger assumptions, *i.e.*, the existence of SNARKs.

Another independent work by Derler and Slamanig [47] also defines M-HS, with a stronger security model than that of Fiore *et al.* [49] but weaker than ours. Specifically, it allows corruption of all but one signer, and the forgery must pass verification under a set of public keys including the non-corrupted one. In contrast, our model allows corruption of *all* signers, whose public keys are involved in the verification of the forgery.

Derler *et al.* [22] introduced homomorphic proxy re-authenticators (HPRA), in which data producers authenticate data under their own keys, and a proxy can evaluate functions over these data and derive a message authentication code (MAC) under a key of the receiver. On the other hand, in M-HS, homomorphic evaluation and verification of signatures can be performed publicly without any secret (*e.g.*, re-authentication and MAC keys).

2.3 Key-Homomorphism

Key-homomorphism has been studied in some earlier work in the context of threshold fully homomorphic encryption [2] and pseudorandom functions [13]. The main inspiration of considering attribute-homomorphism in our work comes from the study of Boneh *et al.* [12] in the context of key-homomorphic encryption (KHE), who formulated KHE and constructed it based on lattice assumptions. Furthermore, they used KHE to construct attribute-based encryption for general circuits with short secret keys.

Whether the name "key-homomorphic" is appropriate for their primitive is debatable, as the "public keys" in KHE are actually attributes possibly with semantic meaning. Unlike homomorphic encryption (HE) which allows homomorphic operations on the ciphertexts with respect to the plaintexts, KHE allows homomorphic operations on the ciphertexts with respect to the attributes. As the plaintexts are private while the attributes are public, KHE and HE are inherently different. For our M-AMHS, however, both messages and attributes are public. It is natural to treat attributes as messages and have the authorities sign them using M-HS.

Derler and Slamanig [47] investigate key-homomorphic signatures in the more literal setting, *i.e.*, the homomorphism is over the key space but not the attribute spaces. Their main goal is to generalize more basic primitives such as ring signatures.

Key-homomorphism in signatures is also considered in different extents in delegatable functional signatures (DFS) [6] and operational signature scheme (OSS) [5]. In the former, the evaluator must use its secret key to derive signatures. The verification algorithm then takes as input both the public key of the original signature as well as the public key of the evaluator. In the latter, the evaluation algorithm takes as input tuples consisting of an identity, a message, and a signature. It outputs another tuple to a targeted identity. DFS is constructed generically from trapdoor permutations, while OSS is constructed from indistinguishability obfuscation and one-way functions. They thus serve as proof-of-concept without giving much intuition of how to achieve key-homomorphism in signatures. Other related notions include policy-based signatures [7], in which a policy-dependent signing key can only sign messages satisfying the policy, and functional signatures [16], in which a functional signing key can only sign messages in the range of the specified function.

3 Preliminaries

Let λ be the security parameter. and let $\mathsf{negl}(\lambda)$ denote functions which are negligible in λ. For an algorithm $\mathcal{A}$, $x \leftarrow \mathcal{A}(\cdot)$ denotes assigning the output from the execution of $\mathcal{A}$ to the variable x. For a set S, $x \leftarrow S$ denotes the sampling of a uniformly random $x \in S$. The empty string and set are denoted by ϵ and ϕ respectively.

3.1 Succinct Non-Interactive Argument

Definition 1 (SNARG). *A tuple of* PPT *algorithms* $\Pi = (\mathsf{Gen}, \mathsf{Prove}, \mathsf{Vf})$ *is a* succinct non-interactive argument *(SNARG) for a language* $L \in \mathsf{NP}$ *with the witness relation* $\mathcal{R}$ *if it satisfies the following:*

- ***Completeness****: For all* x, w *such that* $\mathcal{R}(x, w) = 1$*, and for all common reference strings* $\mathsf{crs} \in \mathsf{Gen}(1^\lambda)$*, we have* $\mathsf{Vf}(\mathsf{crs}, x, \mathsf{Prove}(\mathsf{crs}, x, w)) = 1$*.*
- ***Soundness****: For all* PPT *adversaries* $\mathcal{A}$*,*

$$\Pr[\mathsf{Vf}(\mathsf{crs}, x, \pi) = 1 \ \wedge \ x \notin L : \mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda); (x, \pi) \leftarrow \mathcal{A}(\mathsf{crs})] \leq \mathsf{negl}(\lambda) .$$

- ***Succinctness****: For all* x, w *such that* $\mathcal{R}(x, w) = 1$, $\mathsf{crs} \in \mathsf{Gen}(1^\lambda)$ *and* $\pi \in \mathsf{Prove}(\mathsf{crs}, x, w)$*, there exists a universal polynomial* $p(\cdot)$ *that does not depend on the relation* $\mathcal{R}$*, such that* $|\pi| \leq p(\lambda + \log t)$*, where* t *denotes the runtime of evaluating the relation* $\mathcal{R}$*.*

Definition 2 (ZK-SNARG). *A SNARG* $\Pi = (\mathsf{Gen}, \mathsf{Prove}, \mathsf{Vf})$ *is* zero-knowledge *(ZK) if there exists a* PPT *algorithm* $\mathcal{S} = (\mathcal{S}^{\mathsf{crs}}, \mathcal{S}^{\mathsf{Prove}})$ *such that, for all* PPT *adversaries* $\mathcal{A}$*,*

$$\begin{aligned}&|\Pr[\mathcal{A}^{\mathsf{Prove}(\mathsf{crs},\cdot,\cdot)}(\mathsf{crs}) = 1 : \mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda)] - \\ &\quad \Pr[\mathcal{A}^{\mathcal{S}'(\mathsf{crs},\mathsf{td},\cdot,\cdot)}(\mathsf{crs}) = 1 : (\mathsf{crs}, \mathsf{td}) \leftarrow \mathcal{S}^{\mathsf{crs}}(1^\lambda)]| \leq \mathsf{negl}(\lambda)\end{aligned}$$

where $\mathcal{S}'(\mathsf{crs}, \mathsf{td}, x, w) = \mathcal{S}^{\mathsf{Prove}}(\mathsf{crs}, \mathsf{td}, x)$*.*

Definition 3 ((Strong) SNARK [16, 25]). *A SNARG* $\Pi = (\mathsf{Gen}, \mathsf{Prove}, \mathsf{Vf})$ *is a (strong)* succinct non-interactive argument of knowledge *(SNARK) if there exists a* PPT *algorithm* $\mathsf{E} = (\mathsf{E}^1, \mathsf{E}^2)$ *such that for all* PPT *provers* $\mathcal{A}$*, and for* every *distinguisher* $\mathcal{D}$*,*

$$\begin{aligned}&|\Pr[\mathcal{D}(\mathsf{crs}) = 1 : \mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda)] - \\ &\quad \Pr[\mathcal{D}(\mathsf{crs}) = 1 : (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{E}^1(1^\lambda)]| \leq \mathsf{negl}(\lambda) .\end{aligned}$$

Furthermore,

$$\begin{aligned}&|\Pr[\mathsf{Vf}(\mathsf{crs}, x, \pi) = 1 \wedge (x, w^*) \notin \mathcal{R} : (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{E}^1(1^\lambda), \\ &\quad (x, \pi) \leftarrow \mathcal{A}(\mathsf{crs}), w^* \leftarrow \mathsf{E}^2(\mathsf{crs}, \mathsf{td}, x, \pi)]| \leq \mathsf{negl}(\lambda)\end{aligned}$$

where the probabilities are taken over the random coins of E*. Note that the extractor is not required to take the random tape of the adversary as part of its input.*

Definition 4 (O-SNARK [25]). *A SNARG* $\Pi = (\mathsf{Gen}, \mathsf{Prove}, \mathsf{Vf})$ *is a* succinct non-interactive arguments of knowledge in the presence of oracles for $\mathbb{O}$ *(O-SNARK) for the oracle family* $\mathbb{O}$ *if for all* PPT *provers* $\mathcal{A}$*, there exists a* PPT *algorithm* $\mathsf{E}_{\mathcal{A}}$ *such that*

$$\begin{aligned}&|\Pr[\mathsf{Vf}(\mathsf{crs}, x, \pi) = 1 \ \wedge \ (x, w^*) \notin \mathcal{R} : \mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda), \\ &\quad \mathcal{O} \leftarrow \mathbb{O}; (x, \pi) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{crs}), w^* \leftarrow E_{\mathcal{A}}(\mathsf{crs}, \mathsf{qt})]| \leq \mathsf{negl}(\lambda)\end{aligned}$$

where $\mathsf{qt} = \{q_i, \mathcal{O}(q_i)\}$ *is the transcript of all oracle queries and answers made and received by* $\mathcal{A}$ *during its execution.*

3.2 Signatures

Definition 5 (Digital Signatures). *A signature scheme is a tuple of* PPT *algorithms* $\mathcal{DS}.(\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ *defined as follows:*

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$*: The key generation algorithm takes as input the security parameter* λ *and generates a key pair* $(\mathsf{pk}, \mathsf{sk})$*.*
- $\sigma \leftarrow \mathsf{Sig}(\mathsf{sk}, m)$*: The signing algorithm takes as input a secret key* sk *and a message* $m \in \{0,1\}^*$*. It outputs a signature* σ*.*
- $b \leftarrow \mathsf{Vf}(\mathsf{pk}, m, \sigma)$*: The verification algorithm takes as input a public key* pk*, a message* m*, and a signature* σ*. It outputs a bit* b*.*

Correctness. The scheme is correct if, for all $\lambda \in \mathbb{N}$*, all key pairs* $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{KGen}(1^\lambda)$*, all messages* $m \in \{0,1\}^*$*, and all signatures* $\sigma \in \mathsf{Sig}(\mathsf{sk}, m)$*, it holds that* $\mathsf{Vf}(\mathsf{pk}, m, \sigma) = 1$*.*

Definition 6 (Existential Unforgeability). *A signature scheme* $\mathcal{DS}$ *is* existentially unforgeable under chosen message attacks (EUF-CMA-secure) *if, for all* PPT *adversaries* $\mathcal{A}$*,*

$$\Pr[\mathsf{EUF\text{-}CMA}_{\mathcal{DS},\mathcal{A}}(1^\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

where $\mathsf{EUF\text{-}CMA}_{\mathcal{DS},\mathcal{A}}$ *is an experiment defined as follows:*

- *The challenger* $\mathcal{C}$ *generates* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$ *and gives* pk *to* $\mathcal{A}$*.*
- *The adversary* $\mathcal{A}$ *is given access to a signing oracle* $\mathcal{O}_{\mathsf{Sig}}(\mathsf{sk}, \cdot)$*.*
- *Eventually,* $\mathcal{A}$ *outputs a forgery* (m^*, σ^*)*.*
- *If* m^* *is not queried to the signing oracle, outputs* $\mathsf{Vf}(\mathsf{pk}, m^*, \sigma^*)$*.*
- *Otherwise, the experiment outputs* 0*.*

3.3 Functional Signatures

Definition 7 (Functional Signatures [16]). *A functional signature (FS) scheme for a message space* $\mathcal{M}$*, and a function family* $\mathcal{F} = \{f : \mathcal{D}_f \to \mathcal{M}\}$ *consists of algorithms* $\mathcal{FS}.(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$*.*

- $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathcal{FS}.\mathsf{Setup}(1^\lambda)$*: inputs the security parameter* λ*; and outputs the master secret key* msk *and master public key* mpk*.*
- $\mathsf{sk}_f \leftarrow \mathcal{FS}.\mathsf{KGen}(\mathsf{msk}, f)$*: inputs the master secret key* msk *and a function* $f \in \mathcal{F}$*; and outputs a secret key* sk_f *for* f*.*
- $(f(m), \sigma) \leftarrow \mathcal{FS}.\mathsf{Sig}(f, \mathsf{sk}_f, m)$*: inputs a function* $f \in \mathcal{F}$*, the secret key* sk_f *for the function* f*, and a message* $m \in \mathcal{D}_f$*; and outputs* $f(m)$ *and a signature of* $f(m)$*.*
- $b \leftarrow \mathcal{FS}.\mathsf{Vf}(\mathsf{mpk}, m, \sigma)$*: inputs the master public key* mpk*, a message* m*, and a signature* σ*; and outputs 1 if the signature is valid.*

Correctness. We require that a signature signed under an honestly generated secret key to be valid. Formally, for any $\lambda \in \mathbb{N}$*, any* $(\mathsf{mpk}, \mathsf{msk}) \in \mathcal{FS}.\mathsf{Setup}(1^\lambda)$*, any* $f \in \mathcal{F}$*, any* $\mathsf{sk}_f \in \mathcal{FS}.\mathsf{KGen}(\mathsf{msk}, f)$*, any* $m \in \mathcal{D}_f$*, if* $(m^*, \sigma) \leftarrow \mathcal{FS}.\mathsf{Sig}(f, \mathsf{sk}_f, m)$*, we require that* $\mathcal{FS}.\mathsf{Vf}(\mathsf{mpk}, m^*, \sigma) = 1$*.*

With a secret key of a function, one can only produce new signatures on the range of that function.

Definition 8 (Unforgeability). *An FS scheme* $\mathcal{FS}$ *is unforgeable if the advantage of any* PPT *adversary* $\mathcal{A}$ *in the following game is negligible:*

- *The challenger generates* $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathcal{FS}.\mathsf{Setup}(1^\lambda)$*, and gives* mpk *to* $\mathcal{A}$*.*
- $\mathcal{A}$ *is allowed to query a key generation oracle* $\mathcal{O}_{\mathsf{key}}$ *and a signing oracle* $\mathcal{O}_{\mathsf{sign}}$ *defined as follows. These oracles share a dictionary indexed by tuples* $(f, i) \in \mathcal{F} \times \mathbb{N}$*, whose entries are signing keys:* $\mathsf{sk}_f \leftarrow \mathcal{FS}.\mathsf{KGen}(\mathsf{msk}, f)$*. This dictionary keeps track of the keys that have been previously generated.*

- $\mathcal{O}_{\mathsf{key}}(f, i)$
  - ∗ *If there exists an entry for the key (f, i) in the dictionary, output the corresponding value, sk_f^i.*
  - ∗ *Otherwise, sample a fresh key $\mathsf{sk}_f^i \leftarrow \mathcal{FS}.\mathsf{KGen}(\mathsf{msk}, f)$, then add an entry $(f, i) \rightarrow \mathsf{sk}_f^i$ to the dictionary and output sk_f^i.*
- $\mathcal{O}_{\mathsf{sign}}(f, i, m)$
  - ∗ *If there exists an entry for the key (f, i) in the dictionary, output $\sigma \leftarrow \mathcal{FS}.\mathsf{Sig}(f, \mathsf{sk}_f^i, m)$.*
  - ∗ *Otherwise, sample a fresh key $\mathsf{sk}_f^i \leftarrow \mathcal{FS}.\mathsf{KGen}(\mathsf{msk}, f)$, then add it to the entry (f, i) of the dictionary, and output $\sigma \leftarrow \mathcal{FS}.\mathsf{Sig}(f, \mathsf{sk}_f^i, m)$.*

– *$\mathcal{A}$ wins if it can produce (m^*, σ) such that:*
- $\mathcal{FS}.\mathsf{Vf}(\mathsf{mpk}, m^*, \sigma) = 1$*;*
- *There does not exist m such that $m^* = f(m)$ for any f which was sent as a query to the $\mathcal{O}_{\mathsf{key}}$ oracle;*
- *There does not exist a query (f, m) to $\mathcal{O}_{\mathsf{sign}}$ where $m^* = f(m)$.*

We require the signatures on a message generated by different secret key to be indistinguishable even if master signing key and the secret keys are given.

Definition 9 (Function-Privacy). *An FS scheme $\mathcal{FS}$ is function-private if the advantage of any* PPT *adversary $\mathcal{A}$ in the following game is negligible:*

– *The challenger honestly generates* $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathcal{FS}.\mathsf{Setup}(1^\lambda)$*, and gives* mpk *and* msk *(w.l.o.g. this includes the randomness used in* Setup*) to $\mathcal{A}$.*
– *$\mathcal{A}$ chooses a function f_0 and receives an honestly generated secret key $\mathsf{sk}_{f_0} \leftarrow \mathcal{FS}.\mathsf{KGen}(\mathsf{msk}, f_0)$.*
– *$\mathcal{A}$ chooses a second function f_1 for which $|f_0| = |f_1|$ (where padding can be useful if there is a known upper bound) and receives an honestly generated secret key $\mathsf{sk}_{f_1} \leftarrow \mathcal{FS}.\mathsf{KGen}(\mathsf{msk}, f_1)$.*
– *$\mathcal{A}$ chooses a pair of value m_0, m_1 s.t. $|m_0| = |m_1|$ and $f_0(m_0) = f_1(m_1)$.*
– *The challenger selects a random bit $b \leftarrow \{0, 1\}$ and generates a signature on the image message $m' = f_0(m_0) = f_1(m_1)$ using secret key sk_{f_b}, and gives the resulting signature $\sigma \leftarrow \mathcal{FS}.\mathsf{Sig}(f, \mathsf{sk}_{f_b}, m_b)$ to $\mathcal{A}$.*
– *$\mathcal{A}$ outputs a bit b', and wins the game if $b' = b$.*

We require the signature size to be independent of the size $|m|$ of the input to the function, and the size $|f|$ of the description of f.

Definition 10 (Succinctness). *An FS scheme $\mathcal{FS}$ is succinct, if there exists a polynomial $s(\cdot)$ such that for every $\lambda \in \mathbb{N}$, $f \in \mathcal{F}$, $m \in \mathcal{D}_f$, it holds with probability* 1 *over* $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathcal{FS}.\mathsf{Setup}(1^\lambda)$*;* $\mathsf{sk}_f \leftarrow \mathcal{FS}.\mathsf{KGen}(\mathsf{msk}, f)$*;* $(f(m), \sigma) \leftarrow \mathcal{FS}.\mathsf{Sig}(f, \mathsf{sk}_f, m)$ *that the resulting signature on $f(m)$ has size $|\sigma| \leq s(\lambda, |f(m)|)$.*

4 Multi-key Homomorphic Signatures under Corruption

Our aim is to define and construct multi-key homomorphic signatures (M-HS) which is unforgeable under insider corruption, and study its relation to existing notions. M-HS allows an arbitrary number of signers to generate keys and sign messages independently. In a simplified setting where messages are not labeled, suppose that each signer k signs a message m_k using its secret key sk_k, resulting in a set of signatures $\{\sigma_k\}$. An evaluator can then publicly evaluate a function g over the message-signature pairs (m_k, σ_k) to derive a signature of (m, g) where $m = g(m_1, \ldots, m_K)$. Syntactically, M-HS generalizes the normal homomorphic signatures (HS) since it reduces to HS when a single party owns all K secret keys.

In the multi-signer setting, we must carefully analyze unforgeability when the adversary can corrupt some signers or even maliciously generate some key pairs. Such an insider attack is unnatural in HS since there is only one signer and hence one signing key involved with a signature. We formulate the unforgeability against insider corruption, which requires that such group of corrupt signers cannot produce signatures of (m, g), where the message m is outside the range of the function g restricted by the inputs of the uncorrupted signers. Security against insider attack is especially useful when the output of the function cannot be fully controlled by a few inputs, *e.g.*, functions with AND, majority, and threshold gates. To illustrate the meaning of a forgery, consider the following configuration: Let $g(m_1, \dots, m_K) = \prod_{k=1}^{K} m_k$ be the product function and $m_k \in \mathbb{F}$ for some field $\mathbb{F}$. As long as $m_k = 0$ for some uncorrupted signer k, the adversary is unable to produce a signature of (m, g) where $m \neq 0$.

More interestingly, this requirement actually still makes sense even when there is only one signer who is also the adversary. In this case, unforgeability against insider corruption implies that even the only signer cannot produce a signature of (m, g) if there does not exist m' such that $m = g(m')$. Furthermore, if the signature scheme is context-hiding, the signature of (m, g) can be regarded as an adaptive zero-knowledge succinct non-interactive argument (ZK-SNARG) of the NP language $\{(m, g) : \exists m' \text{ s.t. } m = g(m')\}$ as long as g is efficiently computable.

4.1 Notation

Labeled Programs. Labeled programs are (implicitly) used in various homomorphic signature schemes. Formally, a labeled program $\mathcal{P} = (f, \ell_1, \dots, \ell_n)$ consists of a function $f : \mathcal{M}^n \to \mathcal{M}$ and a set of input labels $\ell_1, \dots, \ell_n$, where ℓ_i is a label for the i-th input of f. A composed program $\mathcal{P}^* = g(\mathcal{P}_1, \dots, \mathcal{P}_t)$ can be constructed by evaluating a function $g : \mathcal{M}^t \to \mathcal{M}$ on the outputs of some labeled programs $\mathcal{P}_1, \dots, \mathcal{P}_t$. Inputs with the same label are grouped together and converted to a single input and the input labels of $\mathcal{P}^*$ are defined as all distinct labels of $\mathcal{P}_1, \dots, \mathcal{P}_t$. An identity program $\mathcal{I}_\ell = (f_{id}, \ell)$ is defined as a labeled program with an identity function $f_{id} : \mathcal{M} \to \mathcal{M}$ and a input label ℓ. A labeled program $\mathcal{P} = (f, \ell_1, \dots, \ell_n)$ can be expressed as the composition of n identity programs $\mathcal{P} = f(\mathcal{I}_{\ell_1}, \dots, \mathcal{I}_{\ell_n})$.

Following previous work [49], we assume every user has an identity id, and their keys are associated to id. To identify users in the multi-key setting using labeled programs, we associate a message to a label $\ell = (\mathsf{id}, \tau)$, where id is a user identity, and τ is a tag.

4.2 Definitions

Syntax. A multi-key homomorphic signature scheme (M-HS) with N-hop evaluation consists of the PPT algorithms $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf}, \mathsf{Eval})$ defined as follows:

– $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ inputs the security parameter λ. It outputs a public parameter pp which is an input to all algorithms implicitly. The public parameter defines the maximum hop of evaluation N, meaning it is not possible to apply Eval on signatures that have been evaluated for N times. The public parameter also defines the identity space $\mathcal{ID}$, the tag space $\mathcal{T}$, the message space $\mathcal{M}$, the class $\mathcal{G}$ of admissible functions. The label space $\mathcal{L} := \mathcal{ID} \times \mathcal{T}$ is defined as the cartesian product of $\mathcal{ID}$ and $\mathcal{T}$. For a labeled program $\mathcal{P} = (f, \ell_1, \dots, \ell_n)$ with labels $\ell_i = (\mathsf{id}_i, \tau_i)$, we use $\mathsf{id} \in \mathcal{P}$ as compact notation for $\mathsf{id} \in \{\mathsf{id}_1, \dots, id_n\}$.
– $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(\mathsf{pp})$ inputs the public parameter. It outputs the public key pk and secret key sk. When an algorithm takes sk as input, we assume its corresponding pk is also taken as input implicitly.
– $\sigma \leftarrow \mathsf{Sig}(\mathsf{sk}, \ell, m)$ inputs the secret key sk, a label $\ell = (\mathsf{id}, \tau) \in \mathcal{L}$, and a message $m \in \mathcal{M}$. It outputs a signature σ. Without loss of generality, we assume the signature is of the form $\sigma = (0, \sigma')$, where 0 indicates it is a fresh signature.
– $\sigma \leftarrow \mathsf{Eval}(g, (\mathcal{P}_k, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}}, m_k, \sigma_k)_{k \in [K]})$ inputs a function $g \in \mathcal{G}$ and, from each contributor, a labeled program $\mathcal{P}_k$, the corresponding public keys $\{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}_k}$, a message m_k, and a signature σ_k, where $k \in [K]$. It outputs a signature σ, certifying that message m is the output of $\mathcal{P} = g(\mathcal{P}_1, \dots, \mathcal{P}_k)$ over some signed labeled messages. Without loss of generality, we assume the signature takes the form $\sigma = (n, \sigma')$, where n indicates that the signature has undergone n hops of evaluation.
– $b \leftarrow \mathsf{Vf}(\mathcal{P}, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}}, m, \sigma)$ inputs a labeled program $\mathcal{P}$, the corresponding public keys $\{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}}$, a message $m \in \mathcal{M}$, and a signature σ. It outputs a bit $b \in \{0, 1\}$, indicating if message m is the output of evaluating $\mathcal{P}$ over some signed labeled messages.

Correctness. Roughly, we require a honestly generated signature $\sigma \leftarrow \mathsf{Sig}(\mathsf{sk}, \ell, m)$ verifies for m as the output of the identity program $\mathcal{I}_\ell$. In addition, we require that if for all $i \in [K]$, σ_i verifies for m_i as the output of a labeled program $\mathcal{P}_i$, then the signature $\sigma \leftarrow \mathsf{Eval}(g, (\mathcal{P}_k, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}}, m_k, \sigma_k)_{k \in [K]})$ verifies for $g(m_1, \ldots, m_k)$ as the output of the composed program $g(\mathcal{P}_1, ..., \mathcal{P}_k)$. Formally, the correctness of an M-HS scheme is defined as follows:

– Signing Correctness: For any $\mathsf{pp} \in \mathsf{Setup}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{KGen}(\mathsf{pp})$, $\ell = (\mathsf{id}, \tau) \in \mathcal{L}$, $m \in \mathcal{M}$, and $\sigma \in \mathsf{Sig}(\mathsf{sk}, \ell, m)$, it holds that $\mathsf{Vf}(\mathcal{I}_\ell, \mathsf{pk}_{\mathsf{id}}, m, \sigma) = 1$.
– Evaluation Correctness: Furthermore, for any $K \in \mathsf{poly}(\lambda)$, any $\mathcal{P}_k$, $\{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}_k}$, m_k, and $\sigma_k = (n_k, \sigma'_k)$ such that $\mathsf{Vf}(\mathcal{P}_k, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}_k}, m_k, \sigma_k) = 1$ where $k \in [K]$, $n_k \leq N - 1$, $g \in \mathcal{G}$, and $\sigma \in \mathsf{Eval}(g, (\mathcal{P}_k, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}}, m_k, \sigma_k)_{k \in [K]})$, it holds that $\mathsf{Vf}(\mathcal{P}, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}}, m, \sigma) = 1$, where $\mathcal{P} = g(\mathcal{P}_1, \ldots, \mathcal{P}_k)$.

Unforgeability. For unforgeability against insider corruption, we require that if some signers are corrupted, they cannot produce a signature (*Type-I forgery*) disrespecting the inputs of honest signers. For example, for a product function $g(m_1, \ldots, m_K) = \prod_{k=1}^{K} m_k$ and $m_k \in \mathbb{Z}$, as long as $m_k = 0$ for some honest signer k, no adversary can forge a signature of $(1, g)$[3]. Even if all signers are corrupted, they cannot produce a signature (*Type-II forgery*) on (m, g) such that m is outside the output range of the function g. For instance, if $g(m) = 0$ for all message m, then no adversary can produce a signature of $(1, g)$.

Formally, we consider the following security game $\mathsf{cEUF\text{-}CMA}$ (*existential unforgeability under corruption and chosen message attack*) between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

– The challenger $\mathcal{C}$ runs $\mathsf{pp} \in \mathsf{Setup}(1^\lambda)$ and gives pp to $\mathcal{A}$. $\mathcal{C}$ initialize empty signing dictionary $D_{\mathsf{Sig}} = \emptyset$ and empty corruption dictionary $D_{\mathsf{Corr}} = \emptyset$.
– The adversary $\mathcal{A}$ is given access to the following oracles, which can be queried adaptively:
  - Signing oracle: $\mathcal{A}$ queries (ℓ, m) where $\ell = (\mathsf{id}, \tau) \in \mathcal{L}$ is a label and $m \in \mathcal{M}$ is a message. If (ℓ, m) is the first query with identity id, $\mathcal{C}$ generates keys $(\mathsf{pk}_{\mathsf{id}}, \mathsf{sk}_{\mathsf{id}}) \leftarrow \mathsf{KGen}(\mathsf{pp})$ and gives $\mathsf{pk}_{\mathsf{id}}$ to $\mathcal{A}$. If $(\ell, m) \notin D_{\mathsf{Sig}}$, $\mathcal{C}$ computes $\sigma_\ell \leftarrow \mathsf{Sig}(\mathsf{sk}_{\mathsf{id}}, \ell, m)$, returns σ_ℓ to $\mathcal{A}$ and updates $D_{\mathsf{Sig}} \leftarrow D_{\mathsf{Sig}} \cup (\ell, m)$, else $\mathcal{C}$ just ignores the query.
  - Corruption oracle: $\mathcal{A}$ queries $\mathsf{id} \in \mathcal{ID}$. If id is not queried to signing oracle and corruption oracle before, $\mathcal{C}$ generates keys $(\mathsf{pk}_{\mathsf{id}}, \mathsf{sk}_{\mathsf{id}}) \leftarrow \mathsf{KGen}(\mathsf{pp})$, gives $(\mathsf{pk}_{\mathsf{id}}, \mathsf{sk}_{\mathsf{id}})$ to $\mathcal{A}$ and updates $D_{\mathsf{Corr}} \leftarrow D_{\mathsf{Corr}} \cup \mathsf{id}$. Else If $\mathsf{id} \notin D_{\mathsf{Corr}}$, $\mathcal{C}$ returns the secret key $\mathsf{sk}_{\mathsf{id}}$ and updates $D_{\mathsf{Corr}} \leftarrow D_{\mathsf{Corr}} \cup \mathsf{id}$. If $\mathsf{id} \in D_{\mathsf{Corr}}$, $\mathcal{C}$ just ignores the query.
– The adversary $\mathcal{A}$ outputs a labeled program $\mathcal{P}^* = (g^*, \ell_1^*, \ldots, \ell_K^*)$, a message m^*, and a signature σ^*. The experiment outputs 1 if and only if $\mathsf{Vf}(\mathcal{P}^*, \{\mathsf{pk}_{\mathsf{id}}^*\}_{\mathsf{id} \in \mathcal{P}^*}, m^*, \sigma^*) = 1$,
  and $(\mathcal{P}^*, m^*, \sigma^*)$ is a forgery of either of the following types:
  - *Type-I*:
    * Denote $T = \{(\mathsf{id}_i^*, i)\}_{\mathsf{id}_i^* \in \mathcal{P}^* \setminus D_{\mathsf{Corr}}}$ as the set of honest signers involved in $\mathcal{P}^*$ and the index of corresponding input. Note that $(\mathsf{id}_i^*, i) \in T$ means that the i-th input of P^* is contributed by the honest signer id_i^*. We require $T \neq \emptyset$. Meaning at least one of the singer involved in $\mathcal{P}^*$ is honest.
    * *Type-I-I*:
      · Denote $S = \{i\}_{(\mathsf{id}_i^*, i) \in T}$ as a set collecting the index of inputs contributed from honest signers. Let $M_i = \{m\}_{(\ell_i^*, m) \in D_{\mathsf{Sig}}}$, denote $g^*(\{M_i\}_{i \in S})$ as a set of possible outputs of g^* when all the inputs of g^* with index $i \in S$ are restricted to the set M_i. We require $\forall\ i \in S$, $M_i \neq \emptyset$, but $m^* \notin g^*(\{M_i\}_{i \in S})$. Meaning m^* is not the correct output of P^* when executed over messages previously authenticated by the honest signers.
    * *Type-I-II*:
      · There exists a label $\ell' = (\mathsf{id}', \tau') \in \mathcal{P}^*$ such that $(\mathsf{id}', \cdot) \in T$ and $(\ell', \cdot) \notin D_{\mathsf{Sig}}$. Meaning $\mathcal{A}$ never made a query with label ℓ'.
  - *Type-II*: $m^* \notin g^*(\cdot)$. Meaning that it is impossible to obtain m^* from $\mathcal{P}^*$.

[3] Formally, a forgery would be certifying a tuple $(\mathcal{P}, 1)$, where $\mathcal{P} = (g, \tau_1, \ldots, \tau_K)$, instead of $(1, g)$.

We say that a M-HS scheme is unforgeable under corruption (cEUF-CMA-secure) if, for all PPT adversaries $\mathcal{A}$, we have $\Pr\{\mathsf{cEUF\text{-}CMA}_{\mathcal{HS},\mathcal{A}} = 1\} \leq \mathsf{negl}(\lambda)$.

We say that the scheme is unforgeable (EUF-CMA-secure) if $\mathcal{A}$ is not allowed to issue any corruption query.

Note that by not allowing the adversary to query two messages with same label to the signing oracle and requiring $\forall \mathsf{id} \in \mathcal{P}^*$, $\mathsf{id} \notin D_{\mathsf{Corr}}$, meaning all signer involved in $\mathcal{P}^*$ are honest, we can get back the definition of previous work [49] in the single database setting [4].

Context Hiding. We require an M-HS scheme to be weakly context hiding, such that the signature on an evaluated message does not reveal information about the function inputs. The property is "weak" since the functionality is not hidden. This is inherent to our notion as the symbolic labeled program is required for verification, as well as to existing homomorphic signatures supporting functionalities beyond linear functions. In the context of verifiable multi-party computation, function inputs should be hidden while the function itself should remain public. Therefore, in this context, weak context-hiding is a more suitable property when compared to a variant which requires the fresh signature to be indistinguishable from the evaluated one, although the latter provides stronger privacy.

Formally, an M-HS scheme $\mathcal{HS}$ is said to be weakly context hiding, if there exists a simulator $\mathcal{S} = (\mathcal{S}^{\mathsf{Setup}}, \mathcal{S}^{\mathsf{Sig}})$ such that for any PPT adversaries $\mathcal{A}$, we have

$$\left|\Pr[\mathsf{ContextHiding}^0_{\mathcal{HS},\mathcal{S},\mathcal{A}}(1^\lambda) = 1] - \Pr[\mathsf{ContextHiding}^1_{\mathcal{HS},\mathcal{S},\mathcal{A}}(1^\lambda) = 1]\right| \leq \mathsf{negl}(\lambda)$$

where for $b \in \{0,1\}$ $\mathsf{ContextHiding}^b_{\mathcal{HS},\mathcal{S},\mathcal{A}}$ are experiments defined in Figure 1.

```
$\mathsf{ContextHiding}^0_{\mathcal{HS},\mathcal{S},\mathcal{A}}(1^\lambda)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$(g, (\mathcal{P}_k, m_k, \sigma_k)_{k=1}^K, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp})$
$b_k \leftarrow \mathsf{Vf}(\mathcal{P}_k, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}_k}, m_k, \sigma_k)\ \forall k \in [K]$

$\sigma \leftarrow \mathsf{Eval}(g, (\mathcal{P}_k, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}, m_k, \sigma_k)_{k=1}^K)$
$b' \leftarrow \mathcal{A}(\mathsf{st}, \sigma)$
return $\wedge_k b_k \wedge b'$
```

```
$\mathsf{ContextHiding}^1_{\mathcal{HS},\mathcal{S},\mathcal{A}}(1^\lambda)$

$(\mathsf{pp}, \mathsf{td}) \leftarrow \mathcal{S}^{\mathsf{Setup}}(1^\lambda)$
$(g, (\mathcal{P}_k, m_k, \sigma_k)_{k=1}^K, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp})$
$b_k \leftarrow \mathsf{Vf}(\mathcal{P}_k, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}_k}, m_k, \sigma_k)\ \forall k \in [K]$
$\mathcal{P} \leftarrow g(\mathcal{P}_1, \ldots, \mathcal{P}_K),\ m \leftarrow g(m_1, \ldots, m_K)$
$\sigma \leftarrow \mathcal{S}^{\mathsf{Sig}}(\mathsf{td}, \mathcal{P}, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}, m)$
$b' \leftarrow \mathcal{A}(\mathsf{st}, \sigma)$
return $\wedge_k b_k \wedge b'$
```

Fig. 1. Context hiding experiments of M-HS

Succinctness. We require the signature size to be independent of the sizes of the inputs to, the descriptions of, and the output of the labeled program.

Formally, an M-HS scheme is succinct if there exists a polynomial $s(\cdot)$, s.t. for any $K \in \mathbb{N}$, $\{\mathcal{P}_k, m_k, \sigma_k\}_{k\in[K]}$, $g \in \mathcal{G}$, and $\sigma \in \mathsf{Eval}(g, (\mathcal{P}_k, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}, m_k, \sigma_k)_{k=1}^K)$, $|\sigma| \leq s(\lambda)$ where λ is the security parameter.

5 Construction

We construct M-HS with unforgeability under corruption generically from ordinary signatures and ZK-SNARKs, which can be seen as a multi-key generalization of the folklore construction of HS. We formalize

[4] To recover their definition in multiple datasets setting, we need to add dataset identifier in our definition. Since one can always include the dataset identifier in the label, and restrict a labeled program to be computed on inputs with the same dataset identifier, we just omit the dataset identifier in this paper.

the following idea. Signatures are produced freshly using an ordinary signature scheme. To perform a homomorphic evaluation, the evaluator proves that it possesses a set of signatures on messages, and the evaluation of a function on these messages produces the resulting message.

We use a family of proof systems recursively by using the proofs (the evaluated signatures) as witnesses to compute other proofs for further homomorphic evaluation.[5] The family of proof systems corresponds to a family of languages, which in turn is parameterized by the number of hops n the signature has been evaluated. A statement $(\mathcal{P}, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}, m)$ is contained in the n-th language denoted by L_n, if $\mathcal{P}$ is of hop n, and for some K and for each $k \in [K]$: 1) there exists a valid signature (a proof) σ_k on some tuple $(\mathcal{P}_k, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}_k}, m_k)$ (which is a statement in the language L_{n-1}, 2) $\mathcal{P} = g(\mathcal{P}_1, \ldots, \mathcal{P}_K)$ for some function g, 3) m is the output of g with inputs $m_1, \ldots, m_K$. Despite recursion, if each proof is succinct, the recursively generated proofs, and hence signatures, are also succinct.

Concretely, we define the family of proof systems and languages as follows. For each $n \in [N]$, let $\mathit{\Pi}_n$ be a proof system for the following NP language L_n:

$$L_1 = \left\{ \begin{array}{l} (\phi, \mathcal{P}, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}, m) : \\ \exists\ (g, (\mathcal{I}_{\ell_k}, m_k, \sigma_k)_{k\in[K]})\ \text{ s.t.} \\ \mathcal{P} = g(\mathcal{I}_{\ell_1}, \ldots, \mathcal{I}_{\ell_K})\ \wedge\ m = g(m_1, \ldots, m_K)\ \wedge \\ \forall k \in [K], \sigma_k = (0, \sigma'_k)\ \wedge\ \mathcal{DS}.\mathsf{Vf}(\mathsf{pk}_{\mathsf{id}_k}, (\ell_k, m_k), \sigma'_k) = 1 \end{array} \right\}$$

$$L_n = \left\{ \begin{array}{l} (\{\mathsf{crs}_i\}_{i=1}^{n-1}, \mathcal{P}, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}, m) : \\ \exists\ (g, (\mathcal{P}_k, m_k, \sigma_k)_{k\in[K]})\ \text{ s.t.} \\ \mathcal{P} = g(\mathcal{P}_1, \ldots, \mathcal{P}_K)\ \wedge\ m = g(m_1, \ldots, m_K)\ \wedge \\ \forall k \in [K], \sigma_k = (n_k, \sigma'_k)\ \wedge\ \mathit{\Pi}_{n-1}.\mathsf{Vf}(\mathsf{crs}_{n_k}, (\{\mathsf{crs}_i\}_{i=1}^{n_k-1}, \mathcal{P}_k, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}_k}, m_k), \sigma'_k) \end{array} \right\}$$

Figure 2 formally shows our generic construction of multi-key homomorphic signature scheme $\mathcal{HS}$ from a digital signature scheme $\mathcal{DS}$ and a proof system $\mathit{\Pi}$. Its correctness follows directly from the correctness of $\mathcal{DS}$ and $\mathit{\Pi}$.

Next, we prove that $\mathcal{HS}$ is unforgeable against insider corruption. If the adversary outputs a signature (a proof) of a tuple $(\mathcal{P}^*, m^*)$ such that m^* is outside the range of the evaluation of $\mathcal{P}^*$ restricted by the inputs of the honest signer, either a proof can be extracted for a statement outside L_n, for some n, which breaks the soundness of $\mathit{\Pi}_n$, or a forgery of $\mathcal{DS}$ verifiable under the public of the honest signer can be extracted, which breaks the unforgeability of $\mathcal{DS}$.

Theorem 1. *If one-way function exists, and $\mathit{\Pi}_n$ is a sound strong SNARK (Definition 3) for all $n \in [N]$, $\mathcal{HS}$ is unforgeable under corruption.*

Proof. EUF-CMA-secure digital signatures can be constructed from one-way functions [36, 40]. Thus, we suppose that $\mathcal{DS}$ is EUF-CMA-secure.

Suppose there exists an adversary $\mathcal{A}_{\mathcal{HS}}$ that produces a forgery in $\mathcal{HS}$ with non-negligible probability. We show how to construct an adversary $\mathcal{A}$ that uses $\mathcal{A}_{\mathcal{HS}}$ to break the soundness of $\mathit{\Pi}_n$ or produce a forgery of $\mathcal{DS}$.

$\mathcal{A}$ first guesses whether the forgery given by $\mathcal{A}_{\mathcal{HS}}$ will be of type-I (which breaks the unforgeability of $\mathcal{DS}$) forge or type-II (which breaks the soundness of $\mathit{\Pi}_n$).

Suppose $\mathcal{A}$ guesses it is a type-I forgery, we write $\mathcal{A}$ as $\mathcal{A}_{\mathcal{DS}}$. $\mathcal{A}_{\mathcal{DS}}$ acts as a challenger in the $\mathsf{cEUF\text{-}CMA}$ game of $\mathcal{HS}$. $\mathcal{A}_{\mathcal{DS}}$ obtains from its challenger the public key $\mathsf{pk}_{\mathcal{DS}}$. It generates for each n $(\mathsf{crs}_n, \mathsf{td}_n) \leftarrow \mathit{\Pi}_n.\mathsf{E}^1(1^\lambda)$, a simulated crs_n for $\mathit{\Pi}_n$, together with a trapdoor td_n, and forwards the public parameters $\mathsf{pp} = (1^\lambda, \mathsf{crs}_0, \ldots, \mathsf{crs}_N)$ to $\mathcal{A}_{\mathcal{HS}}$, where $\mathsf{crs}_0 := \phi$. Then $\mathcal{A}_{\mathcal{DS}}$ initialize empty signing dictionary $D_{\mathsf{Sig}} = \emptyset$ and empty corruption dictionary $D_{\mathsf{Corr}} = \emptyset$.

$\mathcal{A}_{\mathcal{HS}}$ makes two types of queries:

[5] Homomorphic encryption with targeted malleability [14] also used similar techniques.

```
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
    $\mathsf{crs}_0 := \phi$
    $\mathsf{crs}_n \leftarrow \Pi_n.\mathsf{Gen}(1^\lambda)\ \forall n \in [N]$
    return $\mathsf{pp} = (1^\lambda, \{\mathsf{crs}_n\}_{n=0}^N)$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(\mathsf{pp})$
    $(\mathsf{pk}_{\mathcal{DS}}, \mathsf{sk}_{\mathcal{DS}}) \leftarrow \mathcal{DS}.\mathsf{KGen}(1^\lambda)$
    return $(\mathsf{pk}, \mathsf{sk}) := (\mathsf{pk}_{\mathcal{DS}}, \mathsf{sk}_{\mathcal{DS}})$

$b \leftarrow \mathsf{Vf}(\mathcal{P}, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}}, m, \sigma)$
    parse $\sigma = (n, \sigma')$
    $b \leftarrow 0$
    if $n = 0\ \wedge\ \mathcal{P} = \mathcal{I}_\ell$ then
        parse $\ell = (\mathsf{id}, \tau)$
        $b \leftarrow \mathcal{DS}.\mathsf{Vf}(\mathsf{pk}_{\mathsf{id}}, (\ell, m), \sigma')$
    elseif $n \in [N]$ then
        $x := (\{\mathsf{crs}_i\}_{i=1}^{n-1}, \mathcal{P}, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}}, m)$
        $b \leftarrow \Pi_n.\mathsf{Vf}(\mathsf{crs}_n, x, \sigma')$
    endif
    return $b$

$\sigma \leftarrow \mathsf{Sig}(\mathsf{sk}, \ell, m)$
    $\sigma' \leftarrow \mathcal{DS}.\mathsf{Sig}(\mathsf{sk}_{\mathcal{DS}}, (\ell, m))$
    return $\sigma := (0, \sigma')$

$\sigma \leftarrow \mathsf{Eval}(g, (\mathcal{P}_k, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}_k}, m_k, \sigma_k)_{k \in [K]})$
    foreach $k \in [K]$ do
        parse $\sigma_k = (n_k, \sigma'_k)$
    endfor
    $n := \max_k(n_k)$
    $\mathcal{P} \leftarrow g(\mathcal{P}_1, \ldots, \mathcal{P}_K)$
    $m = g(m_1, \ldots, m_K)$
    $x := (\{\mathsf{crs}_i\}_{i=1}^{n}, \mathcal{P}, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}}, m)$
    $w := (g, (\mathcal{P}_k, m_k, \sigma_k)_{k \in [K]})$
    $\sigma' \leftarrow \Pi_{n+1}.\mathsf{Prove}(\mathsf{crs}_{n+1}, x, w)$
    return $\sigma := (n+1, \sigma')$
```

Fig. 2. Construction of M-HS from ZK-SNARK

- Signing query: $\mathcal{A}_{\mathcal{HS}}$ queries (ℓ, m) where $\ell = (\mathsf{id}, \tau) \in \mathcal{L}$ and $m \in \mathcal{M}$. $\mathcal{A}_{\mathcal{DS}}$ first randomly picks a value $q \in [Q]$ where Q is the maximum number of distinct identities $\mathcal{A}_{\mathcal{HS}}$ is allowed to query (to either oracle). If $(\ell = (\mathsf{id}, \tau), m)$ is the first query with identity id, and $\mathcal{A}_{\mathcal{HS}}$ has queried $q-1$ distinct identities (to either oracle) before, then $\mathcal{A}_{\mathcal{DS}}$ sets $\hat{\mathsf{id}} = \mathsf{id}$, $\mathsf{pk}_{\hat{\mathsf{id}}} = \mathsf{pk}_{\mathcal{DS}}$ and gives it to $\mathcal{A}_{\mathcal{HS}}$. Else if $(\ell = (\mathsf{id}, \tau), m)$ is the first query with identity id, $\mathcal{A}_{\mathcal{DS}}$ generates keys $(\mathsf{pk}_{\mathsf{id}}, \mathsf{sk}_{\mathsf{id}}) \leftarrow \mathsf{KGen}(\mathsf{pp})$ and gives $\mathsf{pk}_{\mathsf{id}}$ to $\mathcal{A}$. If $(\ell, m) \notin D_{\mathsf{Sig}}$, and if $\ell = (\hat{\mathsf{id}}, \cdot)$, $\mathcal{A}_{\mathcal{DS}}$ forwards (ℓ, m) to its signing oracle to get σ'_ℓ, else $\mathcal{A}_{\mathcal{DS}}$ computes $\sigma'_\ell \leftarrow \mathsf{Sig}(\mathsf{sk}_{\mathsf{id}}, \ell, m)$. In either case, $\mathcal{A}_{\mathcal{DS}}$ returns $\sigma_\ell = (0, \sigma'_\ell)$ to $\mathcal{A}_{\mathcal{HS}}$ and updates $D_{\mathsf{Sig}} \leftarrow D_{\mathsf{Sig}} \cup (\ell, m)$. If $(\ell, m) \in D_{\mathsf{Sig}}$, $\mathcal{A}_{\mathcal{DS}}$ just ignores the query.
- Corruption query: $\mathcal{A}_{\mathcal{HS}}$ queries $\mathsf{id} \in \mathcal{ID}$. If id is not queried to signing oracle and corruption oracle before, $\mathcal{A}_{\mathcal{DS}}$ generates keys $(\mathsf{pk}_{\mathsf{id}}, \mathsf{sk}_{\mathsf{id}}) \leftarrow \mathsf{KGen}(\mathsf{pp})$, gives $(\mathsf{pk}_{\mathsf{id}}, \mathsf{sk}_{\mathsf{id}})$ to $\mathcal{A}$ and updates $D_{\mathsf{Corr}} \leftarrow D_{\mathsf{Corr}} \cup \mathsf{id}$. If $\mathsf{id} = \hat{\mathsf{id}}$, $\mathcal{A}_{\mathcal{DS}}$ just aborts. Else if $\mathsf{id} \notin D_{\mathsf{Corr}}$, $\mathcal{A}_{\mathcal{DS}}$ returns the secret key $\mathsf{sk}_{\mathsf{id}}$ and updates $D_{\mathsf{Corr}} \leftarrow D_{\mathsf{Corr}} \cup \mathsf{id}$. If $\mathsf{id} \in D_{\mathsf{Corr}}$, $\mathcal{A}_{\mathcal{DS}}$ just ignores the query.

Suppose $\mathcal{A}$ guesses correctly, $\mathcal{A}_{\mathcal{HS}}$ will output, as an alleged forgery of $\mathcal{HS}$, a labeled program $\mathcal{P}^* = (g^*, \ell_1^*, \ldots, \ell_K^*)$, a message m^*, and a signature $\sigma^* = (n^*, \sigma')$ such that $\mathsf{Vf}(\mathcal{P}^*, \{\mathsf{pk}_{\mathsf{id}}^*\}_{\mathsf{id} \in \mathcal{P}^*}, m^*, \sigma^*) = 1$, and $T = \{\mathsf{id}_i^*, i\}_{\mathsf{id}_i^* \in \mathcal{P}^* \setminus D_{\mathsf{Corr}}}$ is not empty. For type-I forgery, $\mathcal{A}_{\mathcal{HS}}$ must query the identity of honest signers to the signing oracle, therefore with non-negligible probability, $(\hat{\mathsf{id}}, \cdot) \in T$. If for all $i \in S$, $M_i \neq \emptyset$, but $m^* \notin g^*(\{\{m\}_{m \in M_i}\}_{i \in S})$ where $S = \{i\}_{(\mathsf{id}_i^*, i) \in T}$ and $M_i = \{m\}_{(\ell_i^*, m) \in D_{\mathsf{Sig}}}$, meaning m^* is not the correct output of P^* when executed over messages previously authenticated by the honest signers, then it is a type-I-I forgery. Else if there exists a label $\ell' = (\mathsf{id}', \tau') \in \mathcal{P}^*$ such that $(\mathsf{id}', \cdot) \in T$ and $(\ell', \cdot) \notin D_{\mathsf{Sig}}$, meaning $\mathcal{A}$ never made a query with label ℓ', then it is a type-I-II forgery. In either case, $\mathcal{A}_{\mathcal{DS}}$ runs $\Pi_n.\mathsf{E}^2$, the extractor of ZK-SNARK for L_n, recursively from $n = n^*$ to $n = 1$, so that it recovers a set of label-message-signature tuples $\{((\ell_k^*, m_k^*), \sigma_k^*)\}$, all pass the verification of $\mathcal{DS}$. With non-negligible probability, there exists a tuple

$((\ell' = (\hat{\mathsf{id}}, \tau'), m'), \sigma') \in \{((\ell_k^*, m_k^*), \sigma_k^*)\}$ such that $(\ell', m') \notin D_{\mathsf{Sig}}$. Suppose that is the case, $((\ell', m'), \sigma')$ is a valid forgery to $\mathcal{DS}$.

Note that by Definition 3 each extractor $\Pi_n.\mathsf{E}^2$ works for all provers and does not take as input the random tape of the prover, which is $\Pi_{n+1}.\mathsf{E}^2$ in our case. Therefore the extraction of each layer contributes an additive, instead of multiplicative, overhead to the runtime of the overall extraction. We can therefore afford a polynomially large number of hops N.

We remark that each extraction succeeds with overwhelming probability $1-u$, where $u \in \mathsf{negl}(\lambda)$. When we have $p \in \mathsf{poly}(\lambda)$ number of recursive extractions, the success probability of the final extraction is:

$$(1-u)^p \geq 1 - \sum_{i=2k+1}^{p} \binom{p}{i} u^i \geq 1 - \sum_{i=2k+1}^{p} (\frac{\mathrm{e}}{i} pu)^i \in 1 - \mathsf{negl}(\lambda)$$

Note that we have used the inequality $\binom{p}{i} \leq (\frac{ep}{i})^i$ where e is the base of natural logarithm.

Suppose $\mathcal{A}$ guesses type-II. $\mathcal{A}$ further guesses $i \in [N]$ such that the given forgery can be used to break the soundness of Π_i. We write $\mathcal{A}$ as $\mathcal{A}_{\Pi_i}$, who acts as a challenger in the $\mathsf{cEUF\text{-}CMA}$ game of $\mathcal{HS}$.

$\mathcal{A}_{\Pi_i}$ obtains from its challenger the common reference strings crs. It sets $\mathsf{crs}_i = \mathsf{crs}$. It generates for each $n \in \{[N]\backslash i\}$, $(\mathsf{crs}_n, \mathsf{td}_n) \leftarrow \Pi_n.\mathsf{E}^1(1^\lambda)$, a simulated crs_n for Π_n, together with a trapdoor td_n. It forwards the public parameters $\mathsf{pp} = (1^\lambda, \mathsf{crs}_0, \ldots, \mathsf{crs}_N)$ to $\mathcal{A}_{\mathcal{HS}}$, where $\mathsf{crs}_0 := \phi$.

$\mathcal{A}_{\mathcal{HS}}$ makes two types of queries:

- Signing query: $\mathcal{A}_{\mathcal{HS}}$ queries (ℓ, m) where $\ell = (\mathsf{id}, \tau) \in \mathcal{L}$ and $m \in \mathcal{M}$. If (ℓ, m) is the first query with identity id, $\mathcal{A}_{\Pi_i}$ generates keys $(\mathsf{pk}_{\mathsf{id}}, \mathsf{sk}_{\mathsf{id}}) \leftarrow \mathsf{KGen}(\mathsf{pp})$ and gives $\mathsf{pk}_{\mathsf{id}}$ to $\mathcal{A}_{\mathcal{HS}}$. If $(\ell, m) \notin D_{\mathsf{Sig}}$, $\mathcal{A}_{\Pi_i}$ computes $\sigma_\ell \leftarrow \mathsf{Sig}(\mathsf{sk}_{\mathsf{id}}, \ell, m)$, returns σ_ℓ to $\mathcal{A}_{\mathcal{HS}}$ and updates $D_{\mathsf{Sig}} \leftarrow D_{\mathsf{Sig}} \cup (\ell, m)$, else $\mathcal{A}_{\Pi_i}$ just ignores the query.
- Corruption query: $\mathcal{A}_{\mathcal{HS}}$ queries $\mathsf{id} \in \mathcal{ID}$. If id is not queried to signing oracle and corruption oracle before, $\mathcal{A}_{\Pi_i}$ generates keys $(\mathsf{pk}_{\mathsf{id}}, \mathsf{sk}_{\mathsf{id}}) \leftarrow \mathsf{KGen}(\mathsf{pp})$, gives $(\mathsf{pk}_{\mathsf{id}}, \mathsf{sk}_{\mathsf{id}})$ to $\mathcal{A}_{\mathcal{HS}}$ and updates $D_{\mathsf{Corr}} \leftarrow D_{\mathsf{Corr}} \cup \mathsf{id}$. Else If $\mathsf{id} \notin D_{\mathsf{Corr}}$, $\mathcal{A}_{\Pi_i}$ returns the secret key $\mathsf{sk}_{\mathsf{id}}$ and updates $D_{\mathsf{Corr}} \leftarrow D_{\mathsf{Corr}} \cup \mathsf{id}$. If $\mathsf{id} \in D_{\mathsf{Corr}}$, $\mathcal{A}_{\Pi_i}$ just ignores the query.

If $\mathcal{A}$ guesses correctly, $\mathcal{A}_{\mathcal{HS}}$ will output, as an alleged forgery of $\mathcal{HS}$, a labeled program $\mathcal{P}^* = (g^*, \ell_1^*, \ldots, \ell_K^*)$, a message m^*, and a signature $\sigma^* = (n^*, \sigma')$ such that $\mathsf{Vf}(\mathcal{P}^*, \{\mathsf{pk}_{\mathsf{id}}^*\}_{\mathsf{id} \in \mathcal{P}^*}, m^*, \sigma^*) = 1$, but $m^* \notin g^*(\cdot)$. $\mathcal{A}_{\Pi_i}$ runs $\Pi_n.\mathsf{E}^2$, the extractor of ZK-SNARK for L_n, recursively from $n = n^*$ to $n = i+1$, so that it recovers a set of a set of labeled program-message-signature tuples $\{(\mathcal{P}_k^*, m_k^*, \sigma_k^*)\}$, all of which pass the verification of Π_i. With non-negligible probability, there exists a tuple $(\mathcal{P}' = (g', \ell_1, \ldots, \ell_t), m', \sigma') \in \{(\mathcal{P}_k^*, m_k^*, \sigma_k^*)\}$ such that $m' \notin g'(\cdot)$. Suppose that is the case, $(\mathcal{P}', \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}'}, m', \sigma')$ breaks the soundness of Π_i. Note that $n \in [N]$ where $N \in \mathsf{poly}(\lambda)$, $\mathcal{A}$ has non-negligible probability of guessing it correctly.

Theorem 2. *Assume one-way function exists. If Π_n is a sound O-SNARK with respect to the signing oracle of $\mathcal{DS}$ (Definition 4) for all $n \in [N]$ where N is a constant, then $\mathcal{HS}$ is unforgeable under corruption. Note that in this case $\mathcal{HS}$ only supports constant-hop (N) evaluation.*

Proof. The proof of this theorem is exactly the same as the proof of unforgeability from strong SNARK (Theorem 1), except that extractors that are dependent to the provers are used. Specifically, $\mathcal{A} := \mathcal{A}_{n^*}$ acts as the prover for the extractor $\Pi_{n^*}.\mathsf{E}^2_{\mathcal{A}}$, and an extractor $\Pi_n.\mathsf{E}^2_{\mathcal{A}_n} := \mathcal{A}_{n-1}$ in the upper layer acts as the prover for the extractor $\Pi_{n-1}.\mathsf{E}^2_{\mathcal{A}_{n-1}}$ in the lower layer. Note that for all $n \in [N]$, the same signing oracle for $\mathcal{DS}$ is required. Therefore, with the transcript of signing oracle queries, the set of extractors $\Pi_n.\mathsf{E}^2_{\mathcal{A}_n}$ for the recursive language is able to extract the witnesses. Note that the runtime of $\Pi_n.\mathsf{E}^2_{\mathcal{A}_n}$ may depend on the runtime of $\mathcal{A}_n$. In general, $\Pi_n.\mathsf{E}^2_{\mathcal{A}_n}$ may run $\mathcal{A}_n$ as a black-box polynomially-many times. In the worst case, suppose $n^* = N$. In this case, even if N is as small as logarithmic, the total runtime of recursively running the set of extractors $\Pi_n.\mathsf{E}^2_{\mathcal{A}_n}$ might become exponential, as the extractors needs to take the provers (the extractor in the layer above) as input, each of which contributes a multiplicative polynomial overhead to the extraction time. We thus restrict N to be a constant.

Candidate Constructions of Strong SNARKs and O-SNARKs. As shown by Fiore *et al.* [25], there are a few candidates of O-SNARK. Computationally-Sound proofs of Micali [41] can be use as O-SNARK without putting any restrictions on the underlying signature scheme in our construction. If we require the underlying digital signatures to be hash and sign signatures and model the hash as random oracle, than all SNARKs can be used as O-SNARKs. In the standard model, if we require the message space of the signature scheme to be properly bounded and require the adversary to query almost the entire message space, or we require the adversary to issue oracle queries before seeing the common reference string, then all SNARKs can be used as O-SNARKs.

On the other hand, as far as we know, no strong SNARK candidate is known, although the notion has been used in the literature [16]. For example, in recent SNARK constructions [42–45] based on knowledge of exponents or certain extractability assumptions, the extractor needs to run the prover as a black-box. This does not affect our overall results in the sense that, constant-hop M-HS constructed from O-SNARKs is sufficient to imply functional signatures and ZK-SNARGs.

Theorem 3. *If Π_n is adaptive zero knowledge for all $n \in [N]$, $\mathcal{HS}$ is weakly context hiding.*

Proof. Π_n is adaptive zero-knowledge, so there exists a simulator $\mathcal{S}_{\Pi_n} = (\mathcal{S}^{\mathsf{crs}}_{\Pi_n}, \mathcal{S}^{\mathsf{Prove}}_{\Pi_n})$ which simulates a proof π_n for any instance in L_n. To construct a simulator $\mathcal{S}_{\mathcal{HS}}$ for $\mathcal{HS}$, we define $\mathcal{S}^{\mathsf{Setup}}_{\mathcal{HS}}$ which simulates the common reference strings crs_n using $\mathcal{S}^{\mathsf{crs}}_{\Pi_n}$, and $\mathcal{S}^{\mathsf{Sig}}_{\mathcal{HS}}$ which simulates the signatures using $\mathcal{S}^{\mathsf{Prove}}_{\Pi_n}$. The proofs simulated from $\mathcal{S}_{\Pi_n}$ are indistinguishable from the real proofs, so the simulated signatures from $\mathcal{S}_{\mathcal{HS}}$ are indistinguishable from the real signatures.

Theorem 4. *If Π_n is succinct for all $n \in [N]$, then $\mathcal{HS}$ is succinct.*

Proof. The size of a signature produced by $\mathsf{Eval}(g, (\mathcal{P}_k, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}_k}, m_k, \sigma_k)_{k \in [K]})$ is the sum of the proof length of Π_n for some n and the length of the binary representation of n, which is logarithmic in the security parameter. The succinctness of $\mathcal{HS}$ follows directly from the succinctness property of Π.

More formally, denote t_1 as the runtime of the relation $\mathcal{R}_1$ associated with language L_1. t_1 is bounded by the sum of the runtime of $K \in \mathsf{poly}(\lambda)$ verification of $\mathcal{DS}$ signatures, and, computation time of $g(m_1, \ldots, m_K)$ where g is an admissible function with evaluation time bounded by $\mathsf{poly}(\lambda)$. Since they are efficiently computable, there exists a polynomial $q_1(\cdot)$ such that $t_1 < q_1(\lambda)$. Then by the succinctness of Π_1, for all $(1, \sigma') \in \mathsf{Eval}(g, (\mathcal{P}_k, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}_k}, m_k, \sigma_k)_{k \in [K]})$, there exists a polynomial $p_1(\cdot)$ such that $|(1, \sigma')| < p_1(\lambda + \log(t_1))$ and thus exists a polynomial $s_1(\cdot)$ such that $|(1, \sigma')| < s_1(\lambda)$.

For signatures $(n, \sigma') \in \mathsf{Eval}(g, (\mathcal{P}_k, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}_k}, m_k, \sigma_k)_{k \in [K]})$ with $n > 1$, the only difference is that the verification of $\mathcal{DS}$ signatures is replaced by verification of Π_{n-1} proofs. As long as verification of Π_{n-1} proofs are efficiently computable, there exists a polynomial $s_n(\cdot)$ such that $|(n, \sigma')| < s_n(\lambda)$ for $n \in \{[N] \backslash 1\}$ and $N \in \mathsf{poly}(\lambda)$.

6 Relation with Existing Notions

6.1 Functional Signatures from cEUF-CMA-secure M-HS

To understand the relation of M-HS with existing notions, we begin by constructing functional signatures [16] (FS) using a 2-key HS. FS (defined in Section 6.1) allows an authority with a master secret key to derive function-specific signing keys for delegation of signing or computation on signed data. Given a signing key for function f, one can only sign messages in the range of f.

We construct FS using an M-HS which supports 1-hop evaluation of signatures signed under two different keys. The FS signing key consists of a fresh M-HS secret key sk and a signature σ_f of the function f signed under the master secret key. To sign a function output $f(m)$, the signer simply signs the input message m using sk, and evaluates the signatures σ_f and σ_m of the function and the message respectively using the universal circuit U, which is defined as $U(f, m) = f(m)$ for any function f and message m. The unforgeability under corruption of the M-HS scheme is crucial, for otherwise, the signer might be able to produce a signature on any message (possibly outside the range of f) using sk.

```
$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathcal{FS}.\mathsf{Setup}(1^\lambda)$
    $\mathsf{pp} \leftarrow \mathcal{HS}.\mathsf{Setup}(1^\lambda)$
    $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathcal{HS}.\mathsf{KGen}(\mathsf{pp})$
    // Fix any distinct identities $\mathsf{id}_0$ and $\mathsf{id}_1$ in $\mathcal{ID}$.
    return $(\mathsf{mpk}, \mathsf{msk})$

$(f(m), \sigma) \leftarrow \mathcal{FS}.\mathsf{Sig}(f, \mathsf{sk}_f, m)$
    parse $\mathsf{sk}_f$ as $(\mathsf{sk}, \sigma_f)$
    // Pad $f$ and $m$ to length $n$.
    $\tau \leftarrow \{0,1\}^\lambda,\ \sigma_m \leftarrow \mathcal{HS}.\mathsf{Sig}(\mathsf{sk}, (\mathsf{id}_1, \tau), m)$
    $\mathsf{pk}_{\mathsf{id}_0} := \mathsf{mpk}$
    $\mathsf{pk}_{\mathsf{id}_1} := \mathsf{pk}$
    $\eta_f := (\mathcal{I}_{\mathsf{id}_0,\mathsf{pk}}, \mathsf{pk}_{\mathsf{id}_0}, f, \sigma_f)$
    $\eta_m := (\mathcal{I}_{\mathsf{id}_1,\tau}, \mathsf{pk}_{\mathsf{id}_1}, m, \sigma_m)$
    $\mathcal{P} = (U, (\mathsf{id}_0, \mathsf{pk}), (\mathsf{id}_1, \tau))$
    $\sigma' \leftarrow \mathcal{HS}.\mathsf{Eval}(\mathcal{P}, (\eta_f, \eta_m))$
    // Pad $U(f,m)$ to length $n$.
    return $(U(f,m), \sigma := (\mathsf{pk}, \tau, \sigma'))$

$\mathsf{sk}_f \leftarrow \mathcal{FS}.\mathsf{KGen}(\mathsf{msk}, f)$
    $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{HS}.\mathsf{KGen}(1^\lambda)$
    $\sigma_f \leftarrow \mathcal{HS}.\mathsf{Sig}(\mathsf{msk}, \mathsf{pk}, f)$
    return $\mathsf{sk}_f := (\mathsf{sk}, \sigma_f)$

$b \leftarrow \mathcal{FS}.\mathsf{Vf}(\mathsf{mpk}, m, \sigma)$
    parse $\sigma$ as $(\mathsf{pk}, \tau, \sigma')$
    $\mathsf{pk}_{\mathsf{id}_0} := \mathsf{mpk}$
    $\mathsf{pk}_{\mathsf{id}_1} := \mathsf{pk}$
    $\mathcal{P} := (U, (\mathsf{id}_0, \mathsf{pk}), (\mathsf{id}_1, \tau))$
    $b \leftarrow \mathcal{HS}.\mathsf{Vf}(\mathcal{P}, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}}, m, \sigma')$
    return $b$
```

Fig. 3. Construction of FS from M-HS

Formally, let U be the universal circuit which takes as input a circuit f and its input m, and computes $U(f,m) = f(m)$. We assume that the description size of f, the length of the input m, and the length of the output $f(m)$ are all bounded by some integer n. Let $\mathcal{HS}.(\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf}, \mathsf{Eval})$ be a 1-hop 2-HS scheme, with label space $\mathcal{L} = \{0,1\} \times \{0,1\}^*$ and message space $\mathcal{M} = \{0,1\}^n$, for a labeled program family $\mathcal{G}$ where $U \in \mathcal{G}$. We construct a functional signature scheme $\mathcal{FS}.(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ for the function family $\mathcal{F} = \{f : \{0,1\}^\ell \to \{0,1\}^k \text{ s.t. } |f|, \ell, k \le n\}$ as shown in Figure 3. The correctness follows straightforwardly from that of $\mathcal{HS}$.

Theorem 5. *If $\mathcal{HS}$ is cEUF-CMA-secure, $\mathcal{FS}$ is unforgeable.*

Proof. Suppose there exists an adversary $\mathcal{A}_{\mathcal{FS}}$ that produces a forgery in $\mathcal{FS}$ with non-negligible probability. We show how to construct an adversary $\mathcal{A}_{\mathcal{HS}}$ that uses $\mathcal{A}_{\mathcal{FS}}$ to produce a forgery of $\mathcal{HS}$. $\mathcal{A}_{\mathcal{HS}}$ acts as a challenger in the unforgeability game of $\mathcal{FS}$.

$\mathcal{A}_{\mathcal{HS}}$ receives public key $\mathsf{pk}_{\mathcal{HS}}$ from the challenger of the EUF-CMA game of $\mathcal{HS}$. It sets the master public key $\mathsf{mpk} = \mathsf{pk}_{\mathcal{HS}}$ and forwards mpk to $\mathcal{A}_{\mathcal{FS}}$. $\mathcal{A}_{\mathcal{FS}}$ makes two types of queries:

- $\mathcal{O}_{\mathsf{key}}(f, i)$
  - If there exists an entry for (f, i) in the dictionary, output the corresponding value, sk_f^i.
  - Otherwise, generate a public key honestly by $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{HS}.\mathsf{KGen}(1^\lambda)$ and query the signing oracle of $\mathcal{HS}$ to get $\sigma_f^i \leftarrow \mathcal{HS}.\mathsf{Sig}(\mathsf{msk}, \mathsf{pk}, f)$. Then add $\mathsf{sk}_f^i = (\mathsf{sk}, \sigma_f^i)$ to the dictionary entry (f, i) and output sk_f^i.
- $\mathcal{O}_{\mathsf{sign}}(f, i, m)$
  - If there exists an entry for (f, i) in the dictionary, retrieve $\mathsf{sk}_f^i = (\mathsf{sk}, \sigma_f^i)$.
  - Otherwise, generate a public key honestly by $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{HS}.\mathsf{KGen}(1^\lambda)$ and query the signing oracle of $\mathcal{HS}$ to get $\sigma_f^i \leftarrow \mathcal{HS}.\mathsf{Sig}(\mathsf{msk}, \mathsf{pk}, f)$. Then add $\mathsf{sk}_f^i = (\mathsf{sk}, \sigma_f^i)$ to the dictionary entry (f, i).
  - In either case, sample $\tau \leftarrow \{0,1\}^\lambda$, and compute $\sigma_m \leftarrow \mathcal{HS}.\mathsf{Sig}(\mathsf{sk}, \tau, m)$ and $\sigma' \leftarrow \mathcal{HS}.\mathsf{Eval}(\mathcal{P} = (U, (\mathsf{id}_0, \mathsf{pk}), (\mathsf{id}_1, \tau)), ((\mathcal{I}_{\mathsf{id}_0,\mathsf{pk}}, f, \sigma_f^i), (\mathcal{I}_{\mathsf{id}_1,\tau}, m, \sigma_m)))$, where U is the universal circuit. Set $\sigma = (\mathsf{pk}, \tau, \sigma')$ and output $(U(f,m), \sigma)$.

After querying the oracles, $\mathcal{A}_{\mathcal{FS}}$ responds with forgery (m^*, σ^*), where $\sigma^* = (\mathsf{pk}^*, \tau^*, \sigma'^*)$. As the answer to the $\mathsf{EUF\text{-}CMA}$ game, $\mathcal{A}_{\mathcal{HS}}$ returns $(\mathcal{P} = (U, (\mathsf{id}_0, \mathsf{pk}^*), (\mathsf{id}_1, \tau^*)), \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}}, m^*, \sigma'^*)$. It is a valid forgery since, by the definition of the unforgeability game of functional signatures, m^* is not in the range of any f queried to the $\mathcal{O}_{\mathsf{key}}$ oracle, and $m^* \neq f(m)$ for any (f, m) queried to the $\mathcal{O}_{\mathsf{sign}}$ oracle.

Theorem 6. *If $\mathcal{HS}$ is weakly context-hiding, $\mathcal{FS}$ is function-private.*

Proof. Let $\mathcal{A}_{\mathcal{FS}}$ be an adversary playing the function-privacy game. Since $\mathcal{HS}$ is weakly context-hiding, there exists a simulator $\mathcal{S}_{\mathcal{HS}}$ which, on input $(\mathcal{P} = (U, (\mathsf{id}_0, \mathsf{pk}), (\mathsf{id}_1, \tau)), \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}}, f(m))$, outputs a signature of $f(m)$ which is indistinguishable from that produced by $\mathcal{FS}.\mathsf{Sig}(f, \mathsf{sk}_f, m)$. We can thus replace the challenger with the simulator $\mathcal{S}_{\mathcal{HS}}$, which is indistinguishable in the view of $\mathcal{A}_{\mathcal{FS}}$ except with negligible probability. The simulated signatures contain no information about the function f and input message m except for $f(m)$. The probability that the adversary $\mathcal{A}_{\mathcal{FS}}$ guesses correctly in the simulated game is negligible.

Theorem 7. *If $\mathcal{HS}$ is succinct, $\mathcal{FS}$ is succinct.*

Proof. The size of a signature produced by $\mathcal{FS}.\mathsf{Sig}(f, \mathsf{sk}_f, m)$ is the signature length of $\mathcal{HS}$. The succinctness of $\mathcal{FS}$ follows directly from that of $\mathcal{HS}$.

Since the existence of secure functional signatures implies that of SNARGs [16], which cannot be constructed from a black-box reduction to falsifiable assumptions [30], we have the following corollary.

Corollary 1. *If cEUF-CMA-secure and weakly context-hiding M-HS (specifically 1-hop 2-HS) exists, then SNARG for* NP *exists. Moreover, the succinctness of M-HS must rely on either non-falsifiable assumptions or non-black-box techniques.*

6.2 ZK-SNARG from cEUF-CMA-secure M-HS

We have shown that the existence of 2-HS implies that of FS, which in turn implies the existence of SNARGs. This implication is somewhat unsatisfactory since we start from ZK-SNARKs but only end up with non-zero-knowledge SNARGs, with M-HS sitting in between. Moreover, it uses an unnecessarily strong assumption (as will be shown below) of the existence of 2-HS, which might be more difficult to construct that (1-)HS (with unforgeability under corruption). Thus, in this section, we construct ZK-SNARGs directly from HS, making M-HS a notion sitting tightly and nicely in between ZK-SNARKs and ZK-SNARGs.

The direct construction is as follows. Let the public parameters of M-HS be the common reference string. The prover generates a fresh M-HS key and signs both the statement x and the witness w. It then evaluates the signatures using a labeled program $\mathcal{P} = (g, \ell_x, \ell_w)$ which, on input (x, w), g outputs x if and only if w is a valid witness of x. It finally outputs the evaluated signature as the proof. Note that behavior of the program $\mathcal{P}$ with respect to the labels ℓ_x and ℓ_w is rather arbitrarily. We remark that Libert *et al.* [38] also use homomorphic signatures to construct proof systems, while the construction is quite different.

Formally, let $\mathcal{HS} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf}, \mathsf{Eval})$ be a 1-depth (1-)HS scheme. Let g be a function such that $g(x, w) = x$ if $R(x, w) = 1$, $\perp$ otherwise. Figure 4 shows our construction of a SNARG $\mathit{\Pi}$ for NP language L with relation R. The completeness follows straightforwardly from the correctness of $\mathcal{HS}$.

Theorem 8. *If $\mathcal{HS}$ is cEUF-CMA-secure, then $\mathit{\Pi}$ is sound.*

Proof. Suppose there exists an adversary $\mathcal{A}_{\mathit{\Pi}}$ that breaks the soundness of $\mathit{\Pi}$ with non-negligible probability. We show how to construct an adversary $\mathcal{A}_{\mathcal{HS}}$ that uses $\mathcal{A}_{\mathit{\Pi}}$ to produce a forgery of $\mathcal{HS}$. $\mathcal{A}_{\mathcal{HS}}$ acts as a challenger in the soundness game of $\mathit{\Pi}$.

$\mathcal{A}_{\mathcal{HS}}$ receives pp from the challenger of $\mathsf{EUF\text{-}CMA}$ game of $\mathcal{HS}$, and forwards the common reference string $\mathsf{crs} := \mathsf{pp}$ to $\mathcal{A}_{\mathit{\Pi}}$. Eventually, $\mathcal{A}_{\mathit{\Pi}}$ responds with (x^*, π^*) such that $\mathsf{Vf}(\mathsf{crs}, x^*, \pi^*) = 1$ but $x^* \notin L$. $\mathcal{A}_{\mathcal{HS}}$ then parses $\pi^* = (\mathsf{pk}^*, \sigma^*)$, sets $\mathsf{pk}_{\mathsf{id}} := \mathsf{pk}$ and answers $(\mathcal{P}^* = (g, (\mathsf{id}, \tau_x), (\mathsf{id}, \tau_w)), x^*, \sigma^*)$ to its $\mathsf{EUF\text{-}CMA}$ game. Since $x^* \notin L$, we have $x^* \neq g(x, w)$ for all $(x, w) \in \mathcal{M}^2$.

Theorem 9. *If $\mathcal{HS}$ is weakly context-hiding, then $\mathit{\Pi}$ is zero-knowledge.*

```
$\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda)$
    $\mathsf{pp} \leftarrow \mathcal{HS}.\mathsf{Setup}(1^\lambda)$
    // Fix any identity $\mathsf{id}$ in $\mathcal{ID}$.
    return $\mathsf{crs} := \mathsf{pp}$

$b \leftarrow \mathsf{Vf}(\mathsf{crs}, x, \pi)$
    parse $\pi$ as $(\mathsf{pk}, \sigma)$
    $\mathsf{pk}_{\mathsf{id}} := \mathsf{pk}$
    $\mathcal{P} := (g, (\mathsf{id}, \tau_x), (\mathsf{id}, \tau_w))$
    return $b \leftarrow \mathcal{HS}.\mathsf{Vf}(\mathcal{P}, \mathsf{pk}_{\mathsf{id}}, x, \pi)$

$\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)$
    $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{HS}.\mathsf{KGen}(\mathsf{pp})$
    $\mathsf{pk}_{\mathsf{id}} := \mathsf{pk}$
    $\sigma_x \leftarrow \mathcal{HS}.\mathsf{Sig}(\mathsf{sk}, (\mathsf{id}, \tau_x), x))$
    $\sigma_w \leftarrow \mathcal{HS}.\mathsf{Sig}(\mathsf{sk}, (\mathsf{id}, \tau_w), w))$
    $\eta_x := (\mathcal{I}_{\mathsf{id}, \tau_x}, \mathsf{pk}_{\mathsf{id}}, x, \sigma_x)$
    $\eta_w := (\mathcal{I}_{\mathsf{id}, \tau_w}, \mathsf{pk}_{\mathsf{id}}, w, \sigma_w)$
    $\mathcal{P} := (g, (\mathsf{id}, \tau_x), (\mathsf{id}, \tau_w))$
    $\sigma \leftarrow \mathcal{HS}.\mathsf{Eval}(\mathcal{P}, (\eta_x, \eta_w))$
    return $\pi := (\mathsf{pk}, \sigma)$
```

Fig. 4. Construction of SNARG from M-HS

Proof. Since $\mathcal{HS}$ is weakly context-hiding, there exists a simulator $\mathcal{S}_{\mathcal{HS}} = (\mathcal{S}_{\mathcal{HS}}^{\mathsf{Setup}}, \mathcal{S}_{\mathcal{HS}}^{\mathsf{Sig}})$ such that, $\mathcal{S}_{\mathcal{HS}}^{\mathsf{Setup}}$ simulates the public parameter, and $\mathcal{S}_{\mathcal{HS}}^{\mathsf{Sig}}$ simulates on input $(\mathcal{P} = (g, (\mathsf{id}, \tau_x), (\mathsf{id}, \tau_w)), \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}}, x)$ a signature on x which is statistically close to the real signatures. We can thus construct $\mathcal{S}_{\Pi}^{\mathsf{crs}}$ using $\mathcal{S}_{\mathcal{HS}}^{\mathsf{Setup}}$ and $\mathcal{S}_{\Pi}^{\mathsf{Prove}}$ using $\mathcal{S}_{\mathcal{HS}}^{\mathsf{Sig}}$, and conclude that Π is zero-knowledge.

Theorem 10. *If $\mathcal{HS}$ is succinct then Π is succinct.*

Proof. The proof produced by $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)$ consists of a λ-bit string and a signature of $\mathcal{HS}$. The succinctness of Π follows directly from that of $\mathcal{HS}$.

If the underlying M-HS scheme is secure in the standard model (without a common reference string), *i.e.*, $\mathsf{pp} = \lambda$, the above construction would yield a ZK-SNARG in the standard model, which is impossible. Therefore, we can also rule out the possibility of constructing standard model M-HS schemes which are unforgeable under corruption. Interestingly, the only existing M-HS scheme [49] is unforgeable (without corruption) in the standard model.

7 Concluding Remark

We study multi-key homomorphic signatures (M-HS) which are unforgeable under corruption and chosen message attacks (cEUF-CMA). We have constructed cEUF-CMA-secure M-HS from zero-knowledge succinct non-interactive argument of knowledge (ZK-SNARK), and shown that the existence of the former implies the existence of zero-knowledge succinct non-interactive argument (ZK-SNARG). Due to the known impossibility of SNARG from non-falsifiable assumptions, we pose it as an open problem to identify a weaker but still reasonably security model of M-HS, with constructions from standard assumptions.

Acknowledgments

We thank some of the anonymous reviewers for their detailed and helpful comments. We also thank Yvo Desmedt and Daniel Wichs for inspiring discussions.

References

1. Jae Hyun Ahn, Dan Boneh, Jan Camenisch, Susan Hohenberger, abhi shelat, and Brent Waters. Computing on authenticated data. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 1–20, Taormina, Sicily, Italy, March 19–21, 2012. Springer, Heidelberg, Germany.

2. Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 483–501, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.
3. Nuttapong Attrapadung, Benoît Libert, and Thomas Peters. Computing on authenticated data: New privacy definitions and constructions. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 367–385, Beijing, China, December 2–6, 2012. Springer, Heidelberg, Germany.
4. Nuttapong Attrapadung, Benoît Libert, and Thomas Peters. Efficient completely context-hiding quotable and linearly homomorphic signatures. In Kurosawa and Hanaoka [35], pages 386–404.
5. Michael Backes, Özgür Dagdelen, Marc Fischlin, Sebastian Gajek, Sebastian Meiser, and Dominique Schröder. Operational signature schemes. Cryptology ePrint Archive, 2014/820, 2014.
6. Michael Backes, Sebastian Meiser, and Dominique Schröder. Delegatable functional signatures. In Cheng et al. [20], pages 357–386.
7. Mihir Bellare and Georg Fuchsbauer. Policy-based signatures. In Krawczyk [34], pages 520–537.
8. John Bethencourt, Dan Boneh, and Brent Waters. Cryptographic methods for storing ballots on a voting machine. In *NDSS 2007*, San Diego, CA, USA, February 28 – March 2, 2007. The Internet Society.
9. 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*, volume 5443 of *LNCS*, pages 68–87, Irvine, CA, USA, March 18–20, 2009. Springer, Heidelberg, Germany.
10. Dan Boneh and David Mandell Freeman. Homomorphic signatures for polynomial functions. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 149–168, Tallinn, Estonia, May 15–19, 2011. Springer, Heidelberg, Germany.
11. Dan Boneh and David Mandell Freeman. Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In Catalano et al. [17], pages 1–16.
12. Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Nguyen and Oswald [39], pages 533–556.
13. Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic PRFs and their applications. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 410–428, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.
14. Dan Boneh, Gil Segev, and Brent Waters. Targeted malleability: homomorphic encryption for restricted computations. In Shafi Goldwasser, editor, *ITCS 2012*, pages 350–366, Cambridge, MA, USA, January 8–10, 2012. ACM.
15. Xavier Boyen, Xiong Fan, and Elaine Shi. Adaptively secure fully homomorphic signatures based on lattices. Cryptology ePrint Archive, Report 2014/916, 2014. `http://eprint.iacr.org/2014/916`.
16. Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Krawczyk [34], pages 501–519.
17. Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors. *PKC 2011*, volume 6571 of *LNCS*, Taormina, Italy, March 6–9, 2011. Springer, Heidelberg, Germany.
18. Dario Catalano, Dario Fiore, and Bogdan Warinschi. Efficient network coding signatures in the standard model. In Fischlin et al. [26], pages 680–696.
19. Dario Catalano, Dario Fiore, and Bogdan Warinschi. Homomorphic signatures with efficient verification for polynomial functions. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 371–389, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.
20. Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors. *PKC 2016, Part I*, volume 9614 of *LNCS*, Taipei, Taiwan, March 6–9, 2016. Springer, Heidelberg, Germany.
21. Jung Hee Cheon and Tsuyoshi Takagi, editors. *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg, Germany.
22. David Derler, Sebastian Ramacher, and Daniel Slamanig. Homomorphic proxy re-authenticators and applications to verifiable multi-user data aggregation. Cryptology ePrint Archive, 2017/086, 2017. To appear in Financial Cryptography 2017.
23. David Derler and Daniel Slamanig. Key-homomorphic signatures and applications to multiparty signatures. Cryptology ePrint Archive, 2016/792, 2016.
24. Dario Fiore, Aikaterini Mitrokotsa, Luca Nizzardo, and Elena Pagnin. Multi-key homomorphic authenticators. In Cheon and Takagi [46], pages 499–530.
25. 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, Part I*, volume 9985 of *LNCS*, pages 108–138, Beijing, China, October 31 – November 3, 2016. Springer, Heidelberg, Germany.

26. Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors. *PKC 2012*, volume 7293 of *LNCS*, Darmstadt, Germany, May 21–23, 2012. Springer, Heidelberg, Germany.
27. David Mandell Freeman. Improved security for linearly homomorphic signatures: A generic framework. In Fischlin et al. [26], pages 697–714.
28. 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*, volume 6056 of *LNCS*, pages 142–160, Paris, France, May 26–28, 2010. Springer, Heidelberg, Germany.
29. Rosario Gennaro and Daniel Wichs. Fully homomorphic message authenticators. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 301–320, Bengalore, India, December 1–5, 2013. Springer, Heidelberg, Germany.
30. Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108, San Jose, CA, USA, June 6–8, 2011. ACM Press.
31. Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled fully homomorphic signatures from standard lattices. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 469–477, Portland, OR, USA, June 14–17, 2015. ACM Press.
32. Robert Johnson, David Molnar, Dawn Xiaodong Song, and David Wagner. Homomorphic signature schemes. In Bart Preneel, editor, *CT-RSA 2002*, volume 2271 of *LNCS*, pages 244–262, San Jose, CA, USA, February 18–22, 2002. Springer, Heidelberg, Germany.
33. Eike Kiltz, Anton Mityagin, Saurabh Panjwani, and Barath Raghavan. Append-only signatures. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP 2005*, volume 3580 of *LNCS*, pages 434–445, Lisbon, Portugal, July 11–15, 2005. Springer, Heidelberg, Germany.
34. Hugo Krawczyk, editor. *PKC 2014*, volume 8383 of *LNCS*, Buenos Aires, Argentina, March 26–28, 2014. Springer, Heidelberg, Germany.
35. Kaoru Kurosawa and Goichiro Hanaoka, editors. *PKC 2013*, volume 7778 of *LNCS*, Nara, Japan, February 26 – March 1, 2013. Springer, Heidelberg, Germany.
36. Leslie Lamport. Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, October 1979.
37. 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, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 289–307, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.
38. Benoît Libert, Thomas Peters, Marc Joye, and Moti Yung. Non-malleability from malleability: Simulation-sound quasi-adaptive NIZK proofs and CCA2-secure encryption from homomorphic signatures. In Nguyen and Oswald [39], pages 514–532.
39. Phong Q. Nguyen and Elisabeth Oswald, editors. *EUROCRYPT 2014*, volume 8441 of *LNCS*, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany.
40. John Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd ACM STOC*, pages 387–394, Baltimore, MD, USA, May 14–16, 1990. ACM Press.
41. Silvio Micali. Computationally Sound Proofs. *SIAM J. Comput*, 30(4):1253–1298, 2000.
42. R. Gennaro, C. Gentry, B. Parno, and M. Raykova. Quadratic span programs and succinct NIZKs without PCPs. In *EUROCRYPT 2013, Proceedings*, pages 626–645, 2013.
43. H. Lipmaa. Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. In *ASIACRYPT 2013, Proceedings, Part I*, pages 41–60, 2013.
44. G. Danezis, C. Fournet, J. Groth, and M. Kohlweiss. Square span programs with applications to succinct NIZK arguments. In *ASIACRYPT 2014, Proceedings, Part I*, pages 532–550, 2014.
45. J. Groth. On the size of pairing-based non-interactive arguments. In *EUROCRYPT 2016, Proceedings, Part II*, pages 305–326, 2016.
46. Jung Hee Cheon and Tsuyoshi Takagi, editors. *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg, Germany.
47. David Derler and Daniel Slamanig. Key-homomorphic signatures and applications to multiparty signatures. Cryptology ePrint Archive, 2016/792, 2016.
48. Alex Escala, Javier Herranz, and Paz Morillo. Revocable attribute-based signatures with adaptive security in the standard model. In Abderrahmane Nitaj and David Pointcheval, editors, *AFRICACRYPT 11*, volume 6737 of *LNCS*, pages 224–241, Dakar, Senegal, July 5–7, 2011. Springer, Heidelberg, Germany.
49. Dario Fiore, Aikaterini Mitrokotsa, Luca Nizzardo, and Elena Pagnin. Multi-key homomorphic authenticators. In Cheon and Takagi [46], pages 499–530.

50. Essam Ghadafi. Stronger security notions for decentralized traceable attribute-based signatures and more efficient constructions. In Kaisa Nyberg, editor, *CT-RSA 2015*, volume 9048 of *LNCS*, pages 391–409, San Francisco, CA, USA, April 20–24, 2015. Springer, Heidelberg, Germany.
51. Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled fully homomorphic signatures from standard lattices. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 469–477, Portland, OR, USA, June 14–17, 2015. ACM Press.
52. Ali El Kaafarani, Essam Ghadafi, and Dalia Khader. Decentralized traceable attribute-based signatures. In Josh Benaloh, editor, *CT-RSA 2014*, volume 8366 of *LNCS*, pages 327–348, San Francisco, CA, USA, February 25–28, 2014. Springer, Heidelberg, Germany.
53. Benoît Libert, San Ling, Fabrice Mouhartem, Khoa Nguyen, and Huaxiong Wang. Signature schemes with efficient protocols and dynamic group signatures from lattice assumptions. In Cheon and Takagi [46], pages 373–403.
54. Hemanta K. Maji, Manoj Prabhakaran, and Mike Rosulek. Attribute-based signatures. In Aggelos Kiayias, editor, *CT-RSA 2011*, volume 6558 of *LNCS*, pages 376–392, San Francisco, CA, USA, February 14–18, 2011. Springer, Heidelberg, Germany.
55. Tatsuaki Okamoto and Katsuyuki Takashima. Efficient attribute-based signatures for non-monotone predicates in the standard model. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 35–52, Taormina, Italy, March 6–9, 2011. Springer, Heidelberg, Germany.
56. Tatsuaki Okamoto and Katsuyuki Takashima. Decentralized attribute-based signatures. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 125–142, Nara, Japan, February 26 – March 1, 2013. Springer, Heidelberg, Germany.
57. Yusuke Sakai, Nuttapong Attrapadung, and Goichiro Hanaoka. Attribute-based signatures for circuits from bilinear map. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016, Part I*, volume 9614 of *LNCS*, pages 283–300, Taipei, Taiwan, March 6–9, 2016. Springer, Heidelberg, Germany.
58. Siamak Fayyaz Shahandashti and Reihaneh Safavi-Naini. Threshold attribute-based signatures and their application to anonymous credential systems. In Bart Preneel, editor, *AFRICACRYPT 09*, volume 5580 of *LNCS*, pages 198–216, Gammarth, Tunisia, June 21–25, 2009. Springer, Heidelberg, Germany.

A Application: Decentralized Attribute-based Signatures

Apart from the natural application of allowing delegation of computation on data authenticated by multiple parties, M-HS readily implies other variants of signatures. As a case study, we study the implications of M-HS to decentralized attribute-based signatures (D-ABS) [56].

Since the introduction [54], various extended features of ABS have been proposed, such as traceability [48], and decentralized authorities [56]. A line of works focus on extending the expressiveness of the verification policies supported, *e.g.*, threshold policies [58], and expressed as non-monotone span programs [55, 56], monotone span programs [54], and monotone circuits [57].

A D-ABS scheme allows multiple authorities to certify different attributes of a signer in a completely distributed manner. After obtaining the certificates from the authorities, the signer can then issue signatures on messages, while at the same time shows that its certified attributes satisfy a certain access policy. The first D-ABS scheme is constructed based on dual pairing vector spaces [56]. Subsequent works proposed generic constructions of D-ABS [50, 52] from standard building blocks, with traceability as an additional feature. Unfortunately, these schemes only support access policies expressed as monotone span program.

We propose another generic construction of D-ABS from M-HS. Thanks to the corruption resistance of M-HS, it provides unforgeability even in the presence of corrupted authorities, and supports access policies expressed as admissible functions of the M-HS scheme, *i.e.*, arbitrary labeled programs.

Due to the tag-based nature of M-HS, this scheme can only achieve linkable anonymity, a weaker notion of anonymity such that signatures issued by the same signer are linkable. Modulo the linkability, it prevents the verifier from learning any information about the attributes associated with the signatures except those leaked from the access policies. On one hand, linkability is useful for ensuring strong accountability. For example, consider a simple membership system where a user can register by issuing a linkable attribute-based signature, so that the server can use the linkable part of the signature as the identity of the user. Indeed, there is a branch of literature which incorporates various forms of linkability into signatures or credentials. On the other hand, one can generically transform this linkable scheme to an unlinkable one: Simply replace the signature by a non-interactive witness-indistinguishable (NIWI) proof of the knowledge of the tag in the M-HS.

A.1 Definitions

Syntax. An attribute-based signature scheme consists of the PPT algorithms ($\mathsf{Setup}, \mathsf{KGen}_{\mathrm{Auth}}, \mathsf{KGen}_{\mathrm{Sig}}, \mathsf{Auth}, \mathsf{Sig}, \mathsf{Vf}$) defined as follows:

– $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ runs by trusted third party, takes as input the security parameter λ and outputs the public parameter pp, which defines the attribute space $\mathcal{X}$, the message space $\mathcal{M}$, and the class of supported policies $\mathcal{F}$.
– $(\mathsf{apk}_{\mathsf{aid}}, \mathsf{ask}_{\mathsf{aid}}) \leftarrow \mathsf{KGen}_{\mathrm{Auth}}(\mathsf{pp})$ runs by authority aid, takes as input the public parameter and outputs its public-secret key pair $(\mathsf{apk}_{\mathsf{aid}}, \mathsf{ask}_{\mathsf{aid}})$.
– $(\mathsf{pk}_{\mathsf{sid}}, \mathsf{sk}_{\mathsf{sid}}) \leftarrow \mathsf{KGen}_{\mathrm{Sig}}(\mathsf{pp})$ runs by signer sid takes as input the public parameter and outputs its public-secret key pair $(\mathsf{pk}_{\mathsf{sid}}, \mathsf{sk}_{\mathsf{sid}})$.
– $\gamma_{\mathsf{aid},\mathsf{sid},x} \leftarrow \mathsf{Auth}(\mathsf{ask}_{\mathsf{aid}}, \mathsf{pk}_{\mathsf{sid}}, x)$ runs by authority aid, takes as input its secret key $\mathsf{ask}_{\mathsf{aid}}$, a signer public key $\mathsf{pk}_{\mathsf{sid}}$, and an attribute $x \in \mathcal{X}$. It outputs a credential $\gamma_{\mathsf{aid},\mathsf{sid},x}$ corresponding to the signer sid and the attribute x.
– $\sigma \leftarrow \mathsf{Sig}((\mathsf{apk}_{\mathsf{aid}}, x, \gamma_{\mathsf{aid},\mathsf{sid},x})_{(\mathsf{aid},x)\in S}, \mathsf{sk}_{\mathsf{sid}}, m, f)$ runs by signer sid, takes as input an ordered list of authority public keys, credentials and attributes $(\mathsf{apk}_{\mathsf{aid}}, x, \gamma_{\mathsf{aid},\mathsf{sid},x})_{(\mathsf{aid},x)\in S}$ for an ordered list of attributes S it owns, its secret key $\mathsf{sk}_{\mathsf{sid}}$, a message m, and an access policy $f \in \mathcal{F}$ that accepts S. It outputs a signature σ signing m under attributes S and policy f.
– $b \leftarrow \mathsf{Vf}(f, \{\mathsf{apk}_{\mathsf{aid}}\}_{\mathsf{aid}\in f}, \mathsf{pk}_{\mathsf{sid}}, m, \sigma)$ inputs an access policy f, an ordered list of authority public keys $\{\mathsf{apk}_{\mathsf{aid}}\}_{\mathsf{aid}\in f}$ involved in f, a signer public key $\mathsf{pk}_{\mathsf{sid}}$, a message m, and a signature σ. It outputs 1 if the signature is valid, 0 otherwise.

Correctness. We require a signature signed with attributes fulfilling the access policy to be valid. Formally, for any $\mathsf{pp} \in \mathsf{Setup}(1^\lambda)$, any $(\mathsf{apk}_{\mathsf{aid}}, \mathsf{ask}_{\mathsf{aid}}) \in \mathsf{KGen}_{\mathrm{Auth}}(\mathsf{pp})$, any $(\mathsf{pk}_{\mathsf{sid}}, \mathsf{sk}_{\mathsf{sid}}) \in \mathsf{KGen}_{\mathrm{Sig}}(\mathsf{pp})$, any $x \in \mathcal{X}$, any $\gamma_{\mathsf{aid},\mathsf{sid},x} \in \mathsf{Auth}(\mathsf{ask}_{\mathsf{aid}}, \mathsf{pk}_{\mathsf{sid}}, x)$, any policy $f \in \mathcal{F}$, any message $m \in \mathcal{M}$, any ordered list of attributes S, it holds that if f accepts S, then $\mathsf{Vf}(f, \{\mathsf{apk}_{\mathsf{aid}}\}_{\mathsf{aid}\in f}, \mathsf{pk}_{\mathsf{sid}}, m, \sigma) = 1$ for all $\sigma \in \mathsf{Sig}((\mathsf{apk}_{\mathsf{aid}}, x, \gamma_{\mathsf{aid},\mathsf{sid},x})_{(\mathsf{aid},x)\in S}, \mathsf{sk}_{\mathsf{sid}}, m, f)$.

Unforgeability. We allow the adversary to submit a signature involving corrupted authorities or corrupted signer. In the former case, even if all authorities are corrupted, we ensure that they cannot sign on behalf of any uncorrupted signer (*Type-I*). In the latter case, even if a corrupted signer colludes with some corrupted authorities, it is infeasible to sign with respect to non-trivial policy regarding uncorrupted authorities (*Type-II*). Formally, consider the game below between an adversary and a challenger.

– Let $L = \mathsf{poly}(\lambda)$, $K = \mathsf{poly}(\lambda)$ be the maximum number of authorities and signer in the system respectively.
– Initialize an empty dictionary D_{Auth} to store direct credential queries and D_{Sig} to store credential queries made by signing oracle.
– The challenger $\mathcal{C}$ runs $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $\{(\mathsf{apk}_{\mathsf{aid}}, \mathsf{ask}_{\mathsf{aid}}) \leftarrow \mathsf{KGen}_{\mathrm{Auth}}(\mathsf{pp})\}_{\mathsf{aid}\in[L]}$, $\{(\mathsf{pk}_{\mathsf{sid}}, \mathsf{sk}_{\mathsf{sid}}) \leftarrow \mathsf{KGen}_{\mathrm{Sig}}(\mathsf{pp})\}_{\mathsf{sid}\in[K]}$, and gives $\{\mathsf{apk}_{\mathsf{aid}}\}_{\mathsf{aid}\in[L]}$, $\{\mathsf{pk}_{\mathsf{sid}}\}_{\mathsf{sid}\in[K]}$ to $\mathcal{A}$.
– The adversary $\mathcal{A}$ makes four types of queries:
  - $\mathcal{O}_{\mathrm{ACorr}}(\mathsf{aid})$: return the authority secret key $\mathsf{ask}_{\mathsf{aid}}$.
  - $\mathcal{O}_{\mathrm{SCorr}}(\mathsf{sid})$: return the signer secret key $\mathsf{sk}_{\mathsf{sid}}$.
  - $\mathcal{O}_{\mathsf{Auth}}(\mathsf{aid}, \mathsf{sid}, x)$: If there exists a key for the entry $(\mathsf{aid}, \mathsf{sid}, x)$ in the dictionary D_{Auth}, return the corresponding value $(\gamma_{\mathsf{aid},\mathsf{sid},x})$. Otherwise, return $\gamma_{\mathsf{aid},\mathsf{sid},x} \leftarrow \mathsf{Auth}(\mathsf{ask}_{\mathsf{aid}}, \mathsf{pk}_{\mathsf{sid}}, x)$ and add $\gamma_{\mathsf{aid},\mathsf{sid},x}$ to the entry $(\mathsf{aid}, \mathsf{sid}, x)$ in the dictionary D_{Auth}.

    - $\mathcal{O}_{\mathsf{Sig}}(S, \mathsf{sid}, m, f)$: return $\perp$ if f does not accept attributes S. For any $(\mathsf{aid}, x) \in S$ such that $D_{\mathsf{Auth}}[(\mathsf{aid}, \mathsf{sid}, x)] = \perp$ and $D_{\mathsf{Sig}}[(\mathsf{aid}, \mathsf{sid}, x)] = \perp$, sample $\gamma_{\mathsf{aid},\mathsf{sid},x} \leftarrow \mathsf{Auth}(\mathsf{ask}_{\mathsf{aid}}, \mathsf{pk}_{\mathsf{sid}}, x)$ and add $\gamma_{\mathsf{aid},\mathsf{sid},x}$ to the entry $(\mathsf{aid}, \mathsf{sid}, x)$ in the dictionary D_{Sig}. Then, for each $(\mathsf{aid}, x) \in S$, retrieve $\gamma_{\mathsf{aid},\mathsf{sid},x}$ from the entry $(\mathsf{aid}, \mathsf{sid}, x)$ in the dictionary D_{Auth} and D_{Sig}, and return $\sigma \leftarrow \mathsf{Sig}((\mathsf{apk}_{\mathsf{aid}}, x, \gamma_{\mathsf{aid},\mathsf{sid},x})_{(\mathsf{aid},x)\in S}, \mathsf{sk}_{\mathsf{sid}}, m, f)$.
- Eventually, $\mathcal{A}$ outputs a policy $f^* \in \mathcal{F}$, an ordered list of authorities $\{\mathsf{apk}_{\mathsf{aid}^*}\}_{\mathsf{aid}^* \in f^*}$, a signer public key $\mathsf{pk}_{\mathsf{sid}^*}$, a message m^*, and a signature σ^*. It wins the game if the following holds:
    - $\mathsf{Vf}(f^*, \{\mathsf{apk}_{\mathsf{aid}^*}\}_{\mathsf{aid}^* \in f^*}, \mathsf{pk}_{\mathsf{sid}^*}, m^*, \sigma^*) = 1$, and
    - $(S, \mathsf{pk}_{\mathsf{sid}^*}, f^*, m^*)$ was not queried to the signing oracle before, where S is any ordered list of attributes that f^* accepts
    - either of the following holds:
        * *Type-I*: signer sid is uncorrupted.
        * *Type-II*: there exists an uncorrupted authority aid' such that $\mathsf{aid}' \in f^*$, and f^* does not accept any ordered list T such that there exists $x \in T$ and $D_{\mathsf{Auth}}[(\mathsf{aid}', \mathsf{sid}^*, x)] \neq \perp$.

We require that for all PPT adversaries $\mathcal{A}$, $\Pr\{\mathcal{A} \text{ wins}\} \leq \mathsf{negl}(\lambda)$.

Linkable Anonymity. Anonymity guarantees that only $f(S)$ is revealed but not the individual values while signatures are linkable in the sense that the public key pk is revealed. Formally, we require that there exists a simulator $\mathcal{S} = (\mathcal{S}^{\mathsf{Setup}}, \mathcal{S}^{\mathsf{Sig}})$ such that, for all $L = \mathsf{poly}(\lambda)$, $\mathsf{aid} \in [L]$, m, x, f and S, it holds that for any PPT adversaries $\mathcal{A}$, the following value is negligible in λ.

$$\left| \Pr\left[\begin{array}{r l} \mathcal{A}(\mathsf{pk}, \mathsf{sk}, (\mathsf{apk}_{\mathsf{aid}}, \mathsf{ask}_{\mathsf{aid}})_{\mathsf{aid}\in[L]}, (\mathsf{apk}_{\mathsf{aid}}, x, \gamma_{\mathsf{aid},x})_{(\mathsf{aid},x)\in S}, \sigma, f, m) \to 1 : & \\ \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) & \\ (\mathsf{apk}_{\mathsf{aid}}, \mathsf{ask}_{\mathsf{aid}}) \leftarrow \mathsf{KGen}_{\mathrm{Auth}}(\mathsf{pp}) & \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}_{\mathrm{Sig}}(\mathsf{pp}) & \\ \gamma_{\mathsf{apk}_{\mathsf{aid}},x} \leftarrow \mathsf{Auth}(\mathsf{ask}_{\mathsf{aid}}, \mathsf{pk}, x) & \\ \sigma \leftarrow \mathsf{Sig}((\mathsf{apk}_{\mathsf{aid}}, x, \gamma_{\mathsf{aid},x})_{(\mathsf{aid},x)\in S}, \mathsf{sk}, m, f) & \end{array}\right] - \Pr\left[\begin{array}{r l} \mathcal{A}(\mathsf{pk}, \mathsf{sk}, (\mathsf{apk}_{\mathsf{aid}}, \mathsf{ask}_{\mathsf{aid}})_{\mathsf{aid}\in[L]}, (\mathsf{apk}_{\mathsf{aid}}, x, \gamma_{\mathsf{aid},x})_{(\mathsf{aid},x)\in S}, \sigma, f, m) \to 1 : & \\ (\mathsf{pp}, \mathsf{td}) \leftarrow \mathcal{S}^{\mathsf{Setup}}(1^\lambda) & \\ (\mathsf{apk}_{\mathsf{aid}}, \mathsf{ask}_{\mathsf{aid}}) \leftarrow \mathsf{KGen}_{\mathrm{Auth}}(\mathsf{pp}) & \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}_{\mathrm{Sig}}(\mathsf{pp}) & \\ \gamma_{\mathsf{apk}_{\mathsf{aid}},x} \leftarrow \mathsf{Auth}(\mathsf{ask}_{\mathsf{aid}}, \mathsf{pk}, x) & \\ \sigma \leftarrow \mathcal{S}^{\mathsf{Sig}}(\mathsf{td}, \{\mathsf{apk}_{\mathsf{aid}}\}_{(\mathsf{aid},x)\in S}, \mathsf{pk}, m, f) & \end{array}\right] \right|$$

A.2 Construction

From a 1-Hop M-HS scheme $\mathcal{HS}.(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf}, \mathsf{Eval})$, we get a generic construction of decentralized attribute-based signatures (D-ABS) immediately. Let $f = (f_1, f_2) \in \mathcal{F}$ be an access policy and $S = (\mathsf{aid}, x)$ be an ordered list of attributes. We say f accepts S if $f_1(S_1) = 1$ and $f_2(S_2) = 1$, where $S_1 = \{\mathsf{aid}\}_{(\mathsf{aid},x)\in S}$ and $S_2 = \{x\}_{(\mathsf{aid},x)\in S}$. Figure 5 presents this construction.

The correctness of ABS follows from the correctness of $\mathcal{HS}$ directly. Suppose $\mathcal{HS}$ is unforgeable, then ABS is weakly unforgeable. Finally, suppose $\mathcal{HS}$ is context-hiding, then ABS is linkably anonymous.

A.3 Instantiations

We can instantiate the above generic construction from the multi-key generalization of the fully homomorphic signature scheme by Gorbunov *et al.* [51] (GVW). We can obtain a D-ABS scheme for general circuits based on lattice assumptions in the standard model. In our terminology, the generalized GVW is an M-HS scheme

```
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
    $\mathsf{pp} \leftarrow \mathcal{HS}.\mathsf{Setup}(1^\lambda)$
    return $\mathsf{pp}$

$(\mathsf{apk}_{\mathsf{aid}}, \mathsf{ask}_{\mathsf{aid}}) \leftarrow \mathsf{KGen}_{\mathrm{Auth}}(\mathsf{pp})$
    $(\mathsf{apk}_{\mathsf{aid}}, \mathsf{ask}_{\mathsf{aid}}) \leftarrow \mathcal{HS}.\mathsf{KGen}(\mathsf{pp})$
    return $(\mathsf{apk}_{\mathsf{aid}}, \mathsf{ask}_{\mathsf{aid}})$

$\gamma_{\mathsf{aid},\mathsf{sid},x} \leftarrow \mathsf{Auth}(\mathsf{ask}_{\mathsf{aid}}, \mathsf{pk}_{\mathsf{sid}}, x)$
    $\gamma_{\mathsf{aid},\mathsf{sid},x} \leftarrow \mathcal{HS}.\mathsf{Sig}(\mathsf{ask}_{\mathsf{aid}}, \mathsf{pk}_{\mathsf{sid}}, x)$
    return $\gamma_{\mathsf{aid},\mathsf{sid},x}$

$(\mathsf{pk}_{\mathsf{sid}}, \mathsf{sk}_{\mathsf{sid}}) \leftarrow \mathsf{KGen}_{\mathrm{Sig}}(\mathsf{pp})$
    $(\mathsf{pk}_{\mathsf{sid}}, \mathsf{sk}_{\mathsf{sid}}) \leftarrow \mathcal{HS}.\mathsf{KGen}(\mathsf{pp})$
    return $(\mathsf{pk}_{\mathsf{sid}}, \mathsf{sk}_{\mathsf{sid}})$

$\sigma \leftarrow \mathsf{Sig}((\mathsf{apk}_{\mathsf{aid}}, x, \gamma_{\mathsf{aid},\mathsf{sid},x})_{(\mathsf{aid},x)\in S}, \mathsf{sk}_{\mathsf{sid}}, m, f)$
    $\sigma' \leftarrow \mathcal{HS}.\mathsf{Sig}(\mathsf{sk}_{\mathsf{sid}}, (\mathsf{sid}, \mathsf{pk}_{\mathsf{sid}}), m)$
    $\eta_x := (\mathcal{I}_{\mathsf{aid},\mathsf{sid}}, \mathsf{apk}_{\mathsf{aid}}, x, \gamma_{\mathsf{aid},\mathsf{sid},x})_{(\mathsf{aid},x)\in S}$
    $\eta_m := (\mathcal{I}_{\mathsf{sid},\mathsf{sid}}, \mathsf{pk}_{\mathsf{sid}}, m, \sigma')$
    $\sigma \leftarrow \mathcal{HS}.\mathsf{Eval}(f_2, (\eta_x, \eta_m))$
    return $\sigma$

$b \leftarrow \mathsf{Vf}(f, \{\mathsf{apk}_{\mathsf{aid}}\}_{\mathsf{aid}\in f}, \mathsf{pk}_{\mathsf{sid}}, m, \sigma)$
    if $f_1(\{\mathsf{aid}\}_{\mathsf{aid}\in f}) \neq 1$
        return $b = 0$
    else
        $\mathcal{P} := (f_2, (\mathsf{aid}, \mathsf{sid})_{\mathsf{aid}\in f}, (\mathsf{sid}, \mathsf{sid}))$
        $b \leftarrow \mathcal{HS}.\mathsf{Vf}(\mathcal{P}, (\{\mathsf{apk}_{\mathsf{aid}}\}_{\mathsf{aid}\in\mathcal{P}}, \mathsf{pk}_{\mathsf{sid}}), m, \sigma)$
        return $b$
```

Fig. 5. Construction of D-ABS from M-AHS

unforgeable for uncorrupted signers. This implies a weak unforgeability of the resulting D-ABS scheme in the sense that the adversary is not allowed to corrupt either the signer or any subset of the authorities. In (generalized) GVW, normal verification takes as long as computing the function g. To improve efficiency, part of the verification can be pre-computed so that the amortized verification cost with the fixed function g can be made constant (as explained in details [51]). This is suitable for our purpose as the access policies of a verifier in D-ABS typically remain relatively stable. The transformation to a fully anonymous scheme can be instantiated by the recent proof system for linear congruences proposed by Libert *et al.* [53].

B Insecurity of Existing Work against Insider Attack

We briefly explain why the existing construction of M-HS by Fiore *et al.* [49] suffers from insider attacks. Since their construction is a multi-key generation of the (single-key) HS by Gorbunov *et al.* [51], we begin by demonstrating how the attack works in the single-key setting, then generalize it to the multi-key setting.

The HS construction by Gorbunov *et al.* [51] is based on the notion of homomorphic trapdoor functions. To recall, a homomorphic trapdoor function f maps a public key pk, an index x, and a pre-image u to an image v. The function is homomorphic in the following sense: Given a function g and some pre-images v_i for $i \in [N]$, one can efficiently compute an image v_g. If u_i where $v_i = f(\mathsf{pk}, x_i, u_i)$ for $i \in [N]$ are additionally given, then one can compute a pre-image $u_{g(x_1,\ldots,x_N)}$. The tuple $(v_g, u_{g(x_1,\ldots,x_N)})$ "encodes" the computation $g(x_1, \ldots, x_N)$ in the sense that $v_g = f(\mathsf{pk}, g(x_1, \ldots, x_N), u_{g(x_1,\ldots,x_N)})$. Note that these computation can be performed without the knowledge of the secret key. Furthermore, given the secret key sk corresponding to pk, *any* image v, and *any* index x, one can "invert" the function by sampling u such that $v = f(\mathsf{pk}, x, u)$. Given such homomorphic trapdoor functions, the construction of HS is almost apparent. Roughly speaking, the secret key corresponds to the signing key of the HS scheme; the public key and a set of images corresponds to the verification key; the indexes correspond to messages; and the pre-images correspond to the signatures.

Note that the inversion capability of the secret key holder (the signer) of the trapdoor function is more than sufficient for signing. In particular, the signer can choose to invert the function on an image-index tuple (v, x) which is otherwise impossible to obtain through homomorphic evaluation. While in a typical setting the signer is assumed to be honest and not to generate pre-images for "invalid" image-index pairs, a malicious signer can sample a pre-image / signature u^* such that $v_g = f(\mathsf{pk}, x, u^*)$ yet x is not in the range of g.

Generalizing to the multi-key setting, a multi-key homomorphic trapdoor function f (constructed implicitly in [49]) maps a set of public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_M$, an index x, and a pre-image u to an image v. The knowledge of a secret key sk corresponding to any pk in the set of public keys suffices to invert f on the tuple (v, x) with respect to $\mathsf{pk}_1, \ldots, \mathsf{pk}_M$. As a consequence, if any of the M signers is corrupt, an adversary can generate signatures that disrespect the messages signed by the other honest signers.