Multi-Key Homomorphic Signatures Unforgeable under Insider Corruption*

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.lai@cs.fau.de,{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. The unforgeability of existing M-HS notions assumes that all signers are honest. We consider a setting where an arbitrary number of signers can be corrupted, called unforgeability under corruption, which is typical for natural applications (*e.g.*, verifiable multi-party computation) of M-HS. Surprisingly, there is a huge gap between M-HS (for arbitrary circuits) with and without unforgeability under corruption: While the latter can be constructed from standard lattice assumptions (ASIACRYPT'16), we show that the former likely relies on non-falsifiable assumptions. Specifically, we propose a generic construction of M-HS with unforgeability under corruption from zero-knowledge succinct non-interactive argument of knowledge (ZK-SNARK) (and other standard assumptions), and then show that such M-HS implies zero-knowledge succinct non-interactive arguments (ZK-SNARG). Our results leave open the pressing question of what level of authenticity and utility can be achieved in the presence of corrupt signers under standard assumptions.

Keywords: homomorphic signatures, multi-key, insider, ZK-SNARK

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 messages. For example, without the secret key, one cannot derive a signature of the result of a computation over the signed messages.

* A previous version of this paper is known as "A Zoo of Homomorphic Signatures: Multi-Key and Key-Homomorphism." The corresponding author is Sherman Chow.

Homomorphic signature (HS) schemes [39] allow a third-party 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, without knowing the secret signing keys. 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 [23]) 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 no adversary can 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 [15,37] 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 no adversary can 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 [29,28]. 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. M-HS allows multiple data producers, who do not or should not share the same key, to contribute signed data for verifiable computation. Unfortunately, existing work [29,28] 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 the scheme of Fiore *et al.* [29] is able to create a signature on any message-function pairs (m, g) regardless of the honest signer inputs (see Appendix A). This problem seems to be 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 to a multi-party computation are honest. We thus see a need for a stronger notion which provides unforgeability even in the presence of corrupt 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 $(m_1, \ldots, m_K)$ each signed under possibly different public keys to a signature of $(g(m_1, \ldots, m_K), g)$ signed under a combined public key. Existing work [29,28] assumes all signers are honest when defining and analyzing unforgeability. In contrast, we define a strong security notion of M-HS called *existential unforgeability under corruption and chosen message attack (cEUF-CMA)*, where the adversary controls 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, where we require that even the (possibly malicious) 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 zero-knowledge succinct non-interactive arguments of knowledge (ZK-SNARK) together with digital signatures. There are some impossibility results regarding the security of SNARKs in the presence of (signing) oracles (O-SNARK) [30]. In particular, there exists a secure signature scheme Σ such that no candidate construction of O-SNARK satisfies 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" proof of knowledge property [16,30], where the extractor takes as input an additional trapdoor and does not make use of the random tape of the adversary. In other words, the extractor does not need to simulate the signing oracle. The second approach is to use an underlying signature scheme for which there exists a secure O-SNARK [30, Section 5]. Either way, by a recursive witness extraction technique, we show that strong ZK-SNARKs implies a "poly-depth" M-HS, and 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 a bit unsatisfactory as it requires a 2-HS. We thus further show in Section 6.2 that the existence of cEUF-CMA-secure single-key HS is sufficient to imply that of SNARG. This makes cEUF-CMA-secure (M-)HS sits nicely between SNARK and SNARG, which only differ by the existence of the knowledge extractor.

Since it is known that the security of SNARGs cannot be based on falsifiable assumptions via black-box reductions [36], it follows that the cEUF-CMA-security of M-HS must also be based on non-falsifiable assumptions or proven via

non-black-box techniques. This impossibility result puts us into an unfortunate situation where, either we rely on strong assumptions for our authenticity guarantee or we settle for some weaker authenticity guarantee. It would be interesting to construct M-HS schemes which can withstand a lower but still reasonable level of corruption from standard assumptions.

Note that the above implications concern about argument systems and HS schemes for the complexity class NP. Another direction of circumventing the impossibility would be to consider restricted classes of admissible functions.

Applications. Being such a powerful primitive, cEUF-CMA-secure M-HS implies most if not all other notions of signatures [23]. This paper describes two extensions in particular, namely, (multi-key) delegatable homomorphic signatures and (multi-key) attribute-message-homomorphic signatures. As these extensions mainly introduce more complicated syntax/functionalities without too much technicality, we only briefly describe them below but omit the details.

1.3 Extensions

We introduce two extensions, *multi-key delegatable homomorphic signatures* (M-DHS) and *multi-key attribute-message-homomorphic signatures* (M-AMHS), which are immediate applications of cEUF-CMA-secure M-HS but seem not to be realizable from non-corruption-resistant M-HS. M-DHS allows a group of signers to jointly fill in data according to a template. If it is not corruption-resistant, a signer may overwrite the template entries filled out by other signers. M-AMHS allows evaluation not only on data but also on attributes, *e.g.*, the trustworthiness of the data provider. If it is not corruption-resistant, a signer may fake its attributes. Here we consider M-HS schemes which support homomorphic evaluation of labeled-data [35] (to be explained in Section 4.1). In a nutshell, such schemes ensure that data with "incompatible" labels cannot be used for computation.

Delegation. M-DHS can be viewed as an extension to append-only signatures (AOS) [40,8]. It is motivated by the following scenario. Suppose that multiple data producers engage in a verifiable multi-party computation. Instead of contributing independently, these data producers are organized to form groups called delegation chains. Similar to AOS, 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 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. M-AMHS allows "attribute-homomorphism" on top of the message-homomorphism of (M-)HS. 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 [23]. For such a scenario, it is natural to have the authorities issue certificates to the data producers. A certificate is a signature on 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. 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 have undergone great development, notably from supporting only addition or multiplication [9,34,11,19,32,44] to bounded-degree polynomials [10,20], and to (leveled) fully homomorphic operations which allow evaluation of general circuits of apriori bounded depth [15,37]. 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.* [29] and Derler and Slamanig [28], who defined multi-key homomorphic signatures with varying level of security. Independent of their work, we initiate the study of multi-key HS with *unforgeability under corruption.*

Fiore *et al.* [29] proposed the notion of multi-key homomorphic authenticators, which generalizes the multi-key homomorphic version of signatures and message authentication codes (MAC). They extended the HS by Gorbunov *et al.* [37] 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-corrupt 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 [29], 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 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 [28] also defined M-HS, with a stronger security model than that of Fiore *et al.* [29] 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.* [27] introduced homomorphic proxy re-authenticators, in which a proxy can evaluate functions over signed data and derive a corresponding MAC under a key of the receiver. To do so, the proxy needs to use some keys derived from the secrets of the signers and the MAC key. In contrast, homomorphic evaluation and verification of M-HS can be performed publicly without any secret.

2.3 Key-Homomorphism

Key-homomorphism has been studied in the context of threshold fully homomorphic encryption [2] and pseudorandom functions [13]. The main inspiration for considering attribute-homomorphism in M-AMHS comes from the study of key-homomorphic encryption (KHE) by Boneh *et al.* [12], 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.

Although KHE is named with the term "key-homomorphic", 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 M-AMHS, both messages and attributes are public. We thus treat attributes as messages and have the authorities sign them using M-HS.

Derler and Slamanig [28] investigate key-homomorphic signatures in the more literal setting, *i.e.*, the homomorphism is over the randomly sampled keys. Their goal is to use a milder assumption to generalize more basic primitives such as ring signatures [25,17] and universal designated-verifier signatures [50,24].

Key-homomorphism in signatures is also considered in different extents in delegatable functional signatures (DFS) [6] and the 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 $\lambda \in \mathbb{N}$ be the security parameter. Let $\mathsf{negl}(\lambda)$ be functions which are negligible in λ. $[n] = \{1, \ldots, n\}$ denotes the set of positive integers at most n where $n \in \mathbb{N}$. For an algorithm $\mathcal{A}$, $x \in \mathcal{A}(\cdot)$ denotes that x is in the output range of $\mathcal{A}$. $x \leftarrow \mathcal{A}(\cdot)$ denotes assigning the output from the execution of algorithm $\mathcal{A}$ to the variable x. For a set S, $x \leftarrow S$ denotes sampling uniformly at random an element from S and naming it x. We use $:=$ to denote the assignment operation. The empty string and the empty set are denoted by ϵ and $\emptyset$ respectively.

3.1 Succinct Non-Interactive Arguments

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* $|\pi| \leq O(p(\lambda))$*.*

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* $\mathcal{A}$*, we have*

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

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

Definition 3 (Strong SNARK [16,30]). *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* $\mathcal{D}$*,*

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

Furthermore,

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

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

Definition 4 (O-SNARK [30]). *A SNARG* $\Pi = (\mathsf{Gen}, \mathsf{Prove}, \mathsf{Vf})$ *is a* succinct non-interactive argument 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*

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

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 for a message space* $\mathcal{M}$ *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* $m \in \mathcal{M}$*. 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* $m \in \mathcal{M}$*, 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,*

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

for all PPT *adversaries* $\mathcal{A}$*, where the experiment* $\mathsf{EUF\text{-}CMA}_{\mathcal{DS},\mathcal{A}}$ *is 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 the signing oracle was not queried on* m^**, the experiment 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* $\mathcal{FS}.(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$*.*

- $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathcal{FS}.\mathsf{Setup}(1^\lambda)$*: This algorithm takes in the security parameter* λ*. It outputs the master public key* mpk *and the master secret key* msk*.*

- $\mathsf{sk}_f \leftarrow \mathcal{FS}.\mathsf{KGen}(\mathsf{msk}, f)$*: This algorithm takes as input the master secret key* msk *and a function* $f \in \mathcal{F}$*. It outputs a secret key* sk_f *for* f*.*
- $(f(m), \sigma) \leftarrow \mathcal{FS}.\mathsf{Sig}(f, \mathsf{sk}_f, m)$*: This algorithm takes as input a function* $f \in \mathcal{F}$*, the secret key* sk_f *for the function* f*, and a message* $m \in \mathcal{D}_f$*. It outputs* $f(m)$ *and a signature of* $f(m)$*.*
- $b \leftarrow \mathcal{FS}.\mathsf{Vf}(\mathsf{mpk}, m, \sigma)$*: This algorithm takes as input the master public key* mpk*, a message* m*, and a signature* σ*. It outputs* 1 *for a valid signature.*

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* $(m^*, \sigma) \leftarrow \mathcal{FS}.\mathsf{Sig}(f, \mathsf{sk}_f, m)$*, then* $\mathcal{FS}.\mathsf{Vf}(\mathsf{mpk}, m^*, \sigma) = 1$*.*

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 for any* PPT *adversary* $\mathcal{A}$ *the probability that it wins 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 keys to be indistinguishable even if the master signing key and the secret keys are given.

Definition 9 (Function-Privacy). *An FS scheme* $\mathcal{FS}$ *is function-private if for any* PPT *adversary* $\mathcal{A}$ *the probability that it wins 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 values* 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 description size $|f|$ of the function f.

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

4 Insider-Secure Multi-Key Homomorphic Signatures

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 all the signatures are generated by the same secret key.

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 non-corrupt 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 majority and threshold gates. To illustrate the meaning of a forgery, consider the following configuration: Let $g(m_1, \ldots, m_K) = \prod_{k=1}^{K} m_k$ be the product function and $m_k \in R$ for some ring R. As long as $m_k = 0$ for some non-corrupt signer k, the adversary should not be able to produce a signature of (m, g) where $m \neq 0$.

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, meaning that the signature of (m, g) reveals nothing more than the tuple (m, g) itself, then it 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 are (implicitly) used in various homomorphic signature schemes in which each message is signed under a label ℓ. A labeled program $\mathcal{P}$ consists of a function f and the input labels of the input to f. Formally, for a message space $\mathcal{M}$, a labeled program $\mathcal{P} = (f, \ell_1, \ldots, \ell_k)$ consists of a function $f : \mathcal{M}^k \to \mathcal{M}$ for some $k \in \mathbb{N}$, and a set of input labels $\ell_1, \ldots, \ell_k$, where ℓ_i is a label for the i-th input of f. 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 an input label ℓ.

Let $\mathcal{P}_i = (f_i, \ell_{i,1}, \ldots, \ell_{i,k_i})$ be some programs for $i \in [n]$ for some $n \in \mathbb{N}$. A *composed program* $\mathcal{P}^* = g(\mathcal{P}_1, \ldots, \mathcal{P}_n) = (g(f_1, \ldots, f_n), \ell_1, \ldots, \ell_{k^*})$ can be constructed by evaluating a function $g : \mathcal{M}^n \to \mathcal{M}$ on the outputs of a set of labeled programs $\mathcal{P}_1, \ldots, \mathcal{P}_n$. For such a composed program $\mathcal{P}^*$, we consider its labeled inputs $(\ell_1, \ldots, \ell_{k^*})$ only consist of all *distinct labeled inputs* of $\mathcal{P}_1, \ldots, \mathcal{P}_n$, where inputs with the same label are converted to a single input. In particular, a labeled program $\mathcal{P} = (f, \ell_1, \ldots, \ell_k)$ can be expressed as the composition of k identity programs $\mathcal{P} = f(\mathcal{I}_{\ell_1}, \ldots, \mathcal{I}_{\ell_k})$.

Following [29], we assume every user has an identity $\mathsf{id} \in \mathcal{ID}$ for some identity space $\mathcal{ID}$, 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 $\tau \in \mathcal{T}$ is a tag in some tag space $\mathcal{T}$.

For a labeled program $\mathcal{P} = (f, \ell_1, \ldots, \ell_n)$ with labels $\ell_i = (\mathsf{id}_i, \tau_i)$, we use $\mathsf{id} \in \mathcal{P}$ as a *compact notation for* $\mathsf{id} \in \{\mathsf{id}_1, \ldots, \mathsf{id}_n\}$.

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 the public parameter pp which is an implicit input to all other M-HS algorithms. The public parameter defines the *maximum "hop" of evaluations* N, meaning it is not possible to apply Eval on signatures that have been evaluated for N times. It also defines the message space $\mathcal{M}$, the class $\mathcal{G}$ of *admissible functions*, the identity space $\mathcal{ID}$, and the tag space $\mathcal{T}$. The label space $\mathcal{L} := \mathcal{ID} \times \mathcal{T}$ is defined as the Cartesian product of $\mathcal{ID}$ and $\mathcal{T}$.

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(\mathsf{pp})$ inputs the public parameter. It outputs the public key pk and the 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 σ is of the form $\sigma = (0, \sigma')$, where 0 indicates it is a fresh signature.
- $\sigma \leftarrow \mathsf{Eval}(g, (\mathcal{P}_k, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}_k}, m_k, \sigma_k)_{k\in[K]})$ inputs a function $g \in \mathcal{G}$ and, from each contributor, a labeled program[3] $\mathcal{P}_k$, 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 $\mathcal{P} = g(\mathcal{P}_1, \ldots, \mathcal{P}_K)$ 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 that an 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 $g(\mathcal{P}_1, \cdots, \mathcal{P}_k)$.

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 $k \in [K]$, $n_k \leq N - 1$, $\sigma \in \mathsf{Eval}(g, (\mathcal{P}_k, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}_k}, m_k, \sigma_k)_{k\in[K]})$, and $g \in \mathcal{G}$, 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 disrespecting the inputs of honest signers. For example, for a product function $g(m_1, \ldots, m_K) =$

[3] Our definition differs from [29] in that Eval takes previous labeled programs as input. The "recursive-proof"-style construction seems to make this unavoidable, as the evaluator needs to produce a proof for "I know some other proofs which satisfy some other statements". These other statements (containing the previous programs) are part of the new statement to be proven. We are not aware of any SNARK in which the prover does not need to take the statement to be proven as input. Another plausible approach to avoid proving the possession of other proofs is that the evaluator "updates" the input proofs. However, "updatable" SNARK is not known to exist. In practice, an evaluator would naturally verify the input signatures before proceeding with evaluations. Since an evaluator is also a verifier, it would need to know the "history" (the previous labeled programs) of the input messages anyway.

$\prod_{k=1}^{K} m_k$ and $m_k \in R$ for some ring R, as long as $m_k = 0$ for some honest signer k, no adversary can forge a signature of $(1, g)$[4]. Even if all signers are corrupted, they cannot produce a signature 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 $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ and gives pp to $\mathcal{A}$. $\mathcal{C}$ initializes a signing dictionary $D_{\mathsf{Sig}} = \emptyset$ and an honest user dictionary $D_{\mathsf{Honest}} = \emptyset$.
- The adversary $\mathcal{A}$ is given adaptive access to the 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 it is the first query with identity id, $\mathcal{C}$ generates keys $(\mathsf{pk}_{\mathsf{id}}, \mathsf{sk}_{\mathsf{id}}) \leftarrow \mathsf{KGen}(\mathsf{pp})$, updates $D_{\mathsf{Honest}} := D_{\mathsf{Honest}} \cup \{\mathsf{id}\}$, 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.
- The adversary $\mathcal{A}$ outputs a labeled program $\mathcal{P}^* = (g^*, \ell_1^*, \ldots, \ell_K^*)$, a set of public keys $\{\mathsf{pk}_{\mathsf{id}}^*\}_{\mathsf{id} \in \mathcal{P}^*}$, a message m^*, and a signature σ^*.
- To describe the winning conditions, we *establish the following notations*:
  - Let $S = \{i : \mathsf{id}_i^* \in \mathcal{P}^* \cap D_{\mathsf{Honest}}\} \subseteq [K]$ denote the set collecting the indexes of inputs contributed from honest signers involved in $\mathcal{P}^*$.
  - Let $M_i = \{m : (\ell_i^*, m) \in D_{\mathsf{Sig}}\}$ denote the set collecting the messages which were queries to the signing oracle with label ℓ_i^*. Note that $\{\ell_i^*\}_{i \in S}$ are the labels of the inputs from the honest signers in the program $\mathcal{P}^*$.
  - Let $g^*(\{M_i\}_{i \in S})$ denote the set of all possible outputs of g^* when all the inputs of g^* with index $i \in S$ are restricted to the set M_i:
    When $S = \emptyset$, meaning there is no honest signer involved in $\mathcal{P}^*$, we define $g^*(\{M_i\}_{i \in S}) = g^*(\cdot)$.
    When $M_i = \emptyset$ for some $i \in S$, meaning that there exists $i \in S$ such that no query to the signing oracle was of the form $(\ell_i^*, \cdot)$, we define $g^*(\{M_i\}_{i \in S}) = \emptyset$.
- The experiment outputs 1 if all the following conditions are satisfied:
  - $\mathsf{Vf}(\mathcal{P}^*, \{\mathsf{pk}_{\mathsf{id}}^*\}_{\mathsf{id} \in \mathcal{P}^*}, m^*, \sigma^*) = 1$.
  - $\mathsf{pk}_{\mathsf{id}_i^*}^* = \mathsf{pk}_{\mathsf{id}_i^*}$ for all $i \in S$: The public keys for honest signers are consistent with those returned by the oracle.
  - $m^* \notin g^*(\{M_i\}_{i \in S})$: When there are honest signers involved in $\mathcal{P}^*$, it requires that m^* is not the correct output of $\mathcal{P}^*$ when executed over messages previously authenticated. When the signers involved in $\mathcal{P}^*$ are all corrupt, it requires that it is impossible to obtain m^* from $\mathcal{P}^*$.

An 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 include maliciously generated public keys in the forgery, *i.e.*, for all $\mathsf{id} \in \mathcal{P}^*$,

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

it holds that $\mathsf{id} \in D_{\mathsf{Honest}}$. Note that this recovers the definition of previous work [29] in the single dataset setting[5].

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.

$\mathsf{ContextHiding}^0_{\mathcal{HS},\mathcal{S},\mathcal{A}}(1^\lambda)$	$\mathsf{ContextHiding}^1_{\mathcal{HS},\mathcal{S},\mathcal{A}}(1^\lambda)$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$	$(\mathsf{pp}, \mathsf{td}) \leftarrow \mathcal{S}^{\mathsf{Setup}}(1^\lambda)$
$(g, (\mathcal{P}_k, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}_k}, m_k, \sigma_k)_{k=1}^K, \mathsf{st})$ $\leftarrow \mathcal{A}(\mathsf{pp})$	$(g, (\mathcal{P}_k, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}_k}, m_k, \sigma_k)_{k=1}^K, \mathsf{st})$ $\leftarrow \mathcal{A}(\mathsf{pp})$
foreach $k \in [K]$ **do**	**foreach** $k \in [K]$ **do**
$b_k \leftarrow \mathsf{Vf}(\mathcal{P}_k, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}_k}, m_k, \sigma_k)$	$b_k \leftarrow \mathsf{Vf}(\mathcal{P}_k, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}_k}, m_k, \sigma_k)$ $\mathcal{P} \leftarrow g(\mathcal{P}_1, \ldots, \mathcal{P}_K)$ $m \leftarrow g(m_1, \ldots, m_K)$
endfor	**endfor**
$\sigma \leftarrow \mathsf{Eval}(g, (\mathcal{P}_k, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}, m_k, \sigma_k)_{k=1}^K)$	$\sigma \leftarrow \mathcal{S}^{\mathsf{Sig}}(\mathsf{td}, \mathcal{P}, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}, m)$
$b' \leftarrow \mathcal{A}(\mathsf{st}, \sigma)$	$b' \leftarrow \mathcal{A}(\mathsf{st}, \sigma)$
return $\left(\bigwedge_k b_k\right) \wedge b'$	**return** $\left(\bigwedge_k b_k\right) \wedge b'$

Fig. 1. Context hiding experiments of M-HS

[5] To recover their definition in the multiple datasets setting, we need to add dataset identifiers to 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.

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 $\lambda \in \mathbb{N}$, $\mathsf{pp} \in \mathsf{Setup}(1^\lambda)$, positive integer $K \in \mathsf{poly}(\lambda)$, $\{\mathcal{P}_k, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}_k}, m_k, \sigma_k\}_{k\in[K]}$, $g \in \mathcal{G}$, and $\sigma \in \mathsf{Eval}(g, (\mathcal{P}_k, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}, m_k, \sigma_k)_{k=1}^K)$, $|\sigma| \leq O(s(\lambda))$.

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 the following idea. Signatures are produced freshly using an ordinary signature scheme. For 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 argument systems recursively by using the proofs (the evaluated signatures) as witnesses to compute other proofs for further homomorphic evaluation.[6] The family of argument 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 such that, 1) for each $k \in [K]$, $(\mathcal{P}_k, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}_k}, m_k)$ in the language L_{n_k} for some $n_k < n$, 2) $\mathcal{P} = g(\mathcal{P}_1, \ldots, \mathcal{P}_K)$ for some function g, 3) m is the output of g with inputs $m_1, \ldots, m_K$. If each proof is succinct, the recursively generated proofs, and hence the signatures, are also succinct.

Concretely, we define the family of argument systems and languages as follows. Let $\mathcal{DS}$ be a signature scheme for some message space $\mathcal{L} \times \mathcal{M}$, where $\mathcal{L} = \mathcal{ID} \times \mathcal{T}$ is a product of some identity space $\mathcal{ID}$ and tag space $\mathcal{T}$. Let $\mathcal{G} \subseteq \{g : \mathcal{M}^* \to \mathcal{M}\}$ be some set of admissible functions which are computable in polynomial time. For each $n \in [N]$, let Π_n be an argument system[7] for the following NP language L_n with witness relation $\mathcal{R}_n$:

$$L_n = \left\{ \begin{array}{l} (\mathcal{P}, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}, m) : \\ \exists\ (g, (\mathcal{P}_k, m_k, \sigma_k)_{k\in[K]}) \ \text{ s.t.} \\ \mathcal{P} = g(\mathcal{P}_1, \ldots, \mathcal{P}_K) \ \wedge \ m = g(m_1, \ldots, m_K) \ \wedge \\ \forall k \in [K], \begin{array}{l} \sigma_k = (n_k, \sigma'_k) \ \wedge \ n_k \in \{0, \ldots, n-1\} \ \wedge \\ \mathcal{R}_{n_k}((\mathcal{P}_k, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}_k}, m_k), \sigma'_k) = 1 \end{array} \end{array} \right\},$$

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

[7] Defined in this way, our scheme produces N crs's. We see two plausible approaches for just using one crs: 1) Define a single "über language" which captures all N languages, so we only have statements in one language to be proven. 2) If an "updatable" SNARK is available, the evaluator does not need to produce new proofs.

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

$\mathsf{crs}_n \leftarrow \Pi_n.\mathsf{Gen}(1^\lambda)\ \forall n \in [N]$

return $\mathsf{pp} = (1^\lambda, \{\mathsf{crs}_n\}_{n\in[N]})$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(\mathsf{pp})$

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

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 $\sigma = (n, \sigma')$

$b := 0$

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**

  $x := (\mathcal{P}, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}, m)$

  $b \leftarrow \Pi_n.\mathsf{Vf}(\mathsf{crs}_n, x, \sigma')$

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

$n := \max_{k\in[K]}(n_k)$

$\mathcal{P} \leftarrow g(\mathcal{P}_1, \ldots, \mathcal{P}_K)$

$m = g(m_1, \ldots, m_K)$

$x := (\mathcal{P}, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}, m)$

$w := (g, (\mathcal{P}_k, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}_k}, m_k, \sigma_k)_{k\in[K]})$

$\sigma' \leftarrow \Pi_{n+1}.\mathsf{Prove}(\mathsf{crs}_{n+1}, x, w)$

return $\sigma := (n+1, \sigma')$

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

except that L_n is defined by the following instead when $n = 1$:

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

Figure 2 formally shows our generic construction of multi-key homomorphic signature scheme $\mathcal{HS}$ from $\mathcal{DS}$ and $\Pi_1, \ldots, \Pi_N$. Its correctness follows directly from the correctness of $\mathcal{DS}$ and $\Pi_1, \ldots, \Pi_N$.

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 Π_n, or a forgery of $\mathcal{DS}$ verifiable under the public key of the honest signer can be extracted, which breaks the unforgeability of $\mathcal{DS}$.

Theorem 1. *If one-way functions exist, and Π_n is a strong SNARK (Definition 3) for all $n \in [N]$, $\mathcal{HS}$ is unforgeable under corruption.*

Proof. EUF-CMA-secure signatures can be constructed from one-way functions [43,49]. 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 Π_n for some n or produce a forgery of $\mathcal{DS}$. Without loss of generality, assume that $\mathcal{A}_{\mathcal{HS}}$ queries the signing oracle on at most $Q = \mathsf{poly}(\lambda)$ distinct identities.

$\mathcal{A}$ first guesses a number $n' \in \{0, \ldots, N\}$ denoting whether the forgery can be used to produce a forgery of $\mathcal{DS}$ (case $n' = 0$) or break the soundness of Π_n for some n (case $n' \in [N]$).

Case 1: Breaking the Unforgrability of $\mathcal{DS}$. Suppose $\mathcal{A}$ guesses $n' = 0$, *i.e.*, it attempts to use $\mathcal{A}_{\mathcal{HS}}$ to produce a forgery of $\mathcal{DS}$, 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 $n \in [N]$, $(\mathsf{crs}_n, \mathsf{td}_n) \leftarrow \Pi_n.\mathsf{E}^1(1^\lambda)$, a simulated crs_n for Π_n, together with a trapdoor td_n, and forwards $\mathsf{pp} = (1^\lambda, \mathsf{crs}_1, \ldots, \mathsf{crs}_N)$ to $\mathcal{A}_{\mathcal{HS}}$. Then $\mathcal{A}_{\mathcal{DS}}$ initializes an empty signing dictionary $D_{\mathsf{Sig}} = \emptyset$ and an empty honest user dictionary $D_{\mathsf{Honest}} = \emptyset$. $\mathcal{A}_{\mathcal{DS}}$ also randomly picks a value $q \in [Q]$.

Let $\hat{\mathsf{id}}$ be the q-th distinct identity on which $\mathcal{A}_{\mathcal{HS}}$ queries the signing oracle. $\mathcal{A}_{\mathcal{DS}}$ answers signing oracle queries as follows:

- $\mathcal{A}_{\mathcal{HS}}$ queries on (ℓ, m) where $\ell = (\mathsf{id}, \tau) \in \mathcal{L}$ and $m \in \mathcal{M}$.
  If this is the first query with identity id, $\mathcal{A}_{\mathcal{DS}}$ configures $\mathsf{pk}_{\mathsf{id}}$ as followings.
  If $\hat{\mathsf{id}} = \mathsf{id}$, $\mathcal{A}_{\mathcal{DS}}$ sets $\mathsf{pk}_{\hat{\mathsf{id}}} := \mathsf{pk}_{\mathcal{DS}}$ and gives it to $\mathcal{A}_{\mathcal{HS}}$, else $\mathcal{A}_{\mathcal{DS}}$ generates keys $(\mathsf{pk}_{\mathsf{id}}, \mathsf{sk}_{\mathsf{id}}) \leftarrow \mathsf{KGen}(\mathsf{pp})$ and gives $\mathsf{pk}_{\mathsf{id}}$ to $\mathcal{A}$.
  When $(\ell, m) \notin D_{\mathsf{Sig}}$, 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.

$\mathcal{A}_{\mathcal{HS}}$ will output, as an alleged forgery of $\mathcal{HS}$, a labeled program $\mathcal{P}^* = (g^*, \ell_1^*, \ldots, \ell_K^*)$, a set of public keys $\{\mathsf{pk}_{\mathsf{id}}^*\}_{\mathsf{id} \in \mathcal{P}^*}$, a message m^*, and a signature $\sigma^* = (n^*, \sigma')$ such that $\mathsf{Vf}(\mathcal{P}^*, \{\mathsf{pk}_{\mathsf{id}}^*\}_{\mathsf{id} \in \mathcal{P}^*}, m^*, \sigma^*) = 1$, $\mathsf{pk}_{\mathsf{id}_i^*}^* = \mathsf{pk}_{\mathsf{id}_i^*}$ for all $i \in S$, and $m^* \notin g^*(\{M_i\}_{i \in S})$, where S is the set of indexes of inputs contributed by honest signers.

If $S = \emptyset$, meaning that all signers involved in $\mathcal{P}^*$ are corrupt, then $\mathcal{A}_{\mathcal{DS}}$ aborts (since the guess $n' = 0$ is wrong). Otherwise, there exists $i \in S$ and with probability at least $1/Q$ we have $\mathsf{id}_i^* = \hat{\mathsf{id}}$.

$\mathcal{A}_{\mathcal{DS}}$ greedily runs $\Pi_n.\mathsf{E}^2$, the extractor of ZK-SNARK for L_n, recursively from $n = n^*$ to $n = 1$, attempting to recover a set of label-message-signature tuples $\{((\ell_k^*, m_k^*), \sigma_k^*)\}$ such that all of which pass the verification of $\mathcal{DS}$. The only case when $\mathcal{A}_{\mathcal{DS}}$ is unable to do so is when there exists $n \in [N]$ such that a statement for which $\mathcal{A}_{\mathcal{DS}}$ possesses a valid proof is actually false. In this case, the guess $n' = 0$ is wrong, and $\mathcal{A}_{\mathcal{DS}}$ aborts.

Suppose $\mathcal{A}_{\mathcal{DS}}$ indeed successfully extracts such label-message-signature tuples. Since all statements for which proofs are extracted are true, *i.e.*, all evaluations are done faithfully, and $m^* \notin g^*(\{M_i\}_{i\in S})$, there must exists a tuple $((\ell' = (\hat{\mathsf{id}}, \tau'), m'), \sigma') \in \{((\ell^*_k, m^*_k), \sigma^*_k)\}_k$ such that $(\ell', m') \notin D_{\mathsf{Sig}}$. Since $\mathsf{pk}^*_{\mathsf{id}^*_i} = \mathsf{pk}_{\mathsf{id}^*_i}$ for all $i \in S$, and in particular $\mathsf{pk}^*_{\hat{\mathsf{id}}} = \mathsf{pk}_{\mathcal{DS}}$, $((\ell', m'), \sigma')$ 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. So, the extraction of each layer contributes an additive, instead of multiplicative, overhead to the runtime of the overall extraction. We can, therefore, afford the number of hops N to be polynomially large.

Case 2: Breaking the Soundness of Π_n. Suppose $\mathcal{A}$ guesses $n' \in [N]$, meaning that it attempts to use the forgery to break the soundness of $\Pi_{n'}$. We write $\mathcal{A}$ as $\mathcal{A}_{\Pi_{n'}}$, who acts as a challenger in the $\mathsf{cEUF\text{-}CMA}$ game of $\mathcal{HS}$.

$\mathcal{A}_{\Pi_{n'}}$ obtains from its challenger the common reference string crs. It sets $\mathsf{crs}_{n'} = \mathsf{crs}$. It generates for each $n \in [N]\backslash\{n'\}$, $(\mathsf{crs}_n, \mathsf{td}_n) \leftarrow \Pi_n.\mathsf{E}^1(1^\lambda)$, *i.e.*, a simulated crs_n for Π_n, together with a trapdoor td_n. It forwards the public parameters $\mathsf{pp} = (1^\lambda, \mathsf{crs}_1, \ldots, \mathsf{crs}_N)$ to $\mathcal{A}_{\mathcal{HS}}$. Then $\mathcal{A}_{\mathcal{DS}}$ initializes an empty signing dictionary $D_{\mathsf{Sig}} = \emptyset$ and an empty honest user dictionary $D_{\mathsf{Honest}} = \emptyset$.

$\mathcal{A}_{\Pi_{n'}}$ answers signing oracle queries as follows:

- $\mathcal{A}_{\mathcal{HS}}$ 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 the query is ignored.

$\mathcal{A}_{\mathcal{HS}}$ will output, as an alleged forgery of $\mathcal{HS}$, a labeled program $\mathcal{P}^* = (g^*, \ell^*_1, \ldots, \ell^*_K)$, a set of public keys $\{\mathsf{pk}^*_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}^*}$, 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 $m^* \notin g^*(\{M_i\}_{i\in S})$.

$\mathcal{A}_{\Pi_{n'}}$ greedily runs $\Pi_n.\mathsf{E}^2$, the extractor of ZK-SNARK for L_n, recursively from $n = n^*$ to n', attempting to recover all tuples $\{(\mathcal{P}^*_k, \{\mathsf{pk}^*_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}^*_k}, m^*_k, \sigma^*_k)\}$ such that all of which passes the verification of $\Pi_{n'}$. The only case when $\mathcal{A}_{\Pi_{n'}}$ is unable to do so is when there exists $n \in \{n^*, \ldots, n'+1\}$ such that a statement in L_n induced by the forgery is false. In this case, the guess n' is wrong, and $\mathcal{A}_{\Pi_{n'}}$ aborts.

Suppose the above greedy extraction is successful, $\mathcal{A}_{\Pi_{n'}}$ checks if there exists an extracted tuple which does not satisfy the relation for $L_{n'}$. If so, then $\mathcal{A}_{\Pi_{n'}}$ successfully obtains a $\Pi_{n'}$ proof for a false statement and hence breaks the soundness of $\Pi_{n'}$. If not, then the guess n' is wrong and $\mathcal{A}_{\Pi_{n'}}$ aborts.

Summary. Overall, since the abort conditions of $\mathcal{A}$ for different choices of n' are disjoint, and n' is chosen randomly from $\{0, \ldots, N\}$, the probability that $\mathcal{A}$ does not abort is non-negligible. Therefore, we conclude that $\mathcal{A}$ can either break the unforgeability of $\mathcal{DS}$, or the soundness of Π_n for some $n \in [N]$.

Theorem 2. *Assume one-way function exists. If Π_n is an 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 is exactly the same as the proof of unforgeability from strong SNARK (Theorem 1), except that extractors with dependence on 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 need 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.* [30], there are a few candidates of O-SNARK. Computationally-sound proofs of Micali [47] can be used as O-SNARK without putting any restrictions on the underlying signature scheme in our construction. If we require the underlying signatures to be hash-and-sign signatures and model the hash as a random oracle, then 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.

Yet, 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 [33,46,26,38] 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 zero knowledge for $n \in [N]$, $\mathcal{HS}$ is weakly context hiding.*

Proof. Π_n is 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. *Let $N = \mathsf{poly}(\lambda)$ be a positive integer. 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 proof length of Π_n for some n plus the length of the binary representation of n. By the succinctness of Π_n, the proof length of Π_n is bounded by $O(p(\lambda))$ for some fixed polynomial p. Since $N \in \mathsf{poly}(\lambda)$ and $n \in [N]$, the binary representation of n is of size $O(\log \lambda)$. Therefore, $\mathcal{HS}$ is succinct.

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 (Definition 7) allows an authority with a master secret key to derive function-specific signing keys. Given a signing key for function f, one can only sign messages in the range of f.

We construct FS using an M-HS supporting 1-hop evaluation of signatures signed under two different keys. For the setup, we generate two sets of M-HS keys, include both public keys and one secret key sk_1 in the master public key, and keep the other secret key sk_0 as the master secret key. The FS signing key consists of 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_1, 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 (under the combined key $(\mathsf{pk}_0, \mathsf{pk}_1)$) on any message (possibly outside the range of f) using sk_1.

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 $n = \mathsf{poly}(\lambda)$. Let $\mathcal{F} = \{f : \{0,1\}^\ell \to \{0,1\}^k \text{ s.t. } |f|, \ell, k \le n\}$ denote the function family. 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 $\mathcal{G}$ such that $U \in \mathcal{G}$. We construct a functional signature scheme $\mathcal{FS}.(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ for the function family $\mathcal{F}$ 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. With an adversary $\mathcal{A}_{\mathcal{FS}}$ that produces a forgery of $\mathcal{FS}$ with non-negligible probability, we 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 pp and $\mathsf{pk}_{\mathcal{HS}}$ from the $\mathsf{EUF\text{-}CMA}$ game of $\mathcal{HS}$. It sets $\mathsf{pk}_0 := \mathsf{pk}_{\mathcal{HS}}$ and generates $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathcal{HS}.\mathsf{KGen}(\mathsf{pp})$. It sets the master public key $\mathsf{mpk} = (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{sk}_1)$ and forwards mpk to $\mathcal{A}_{\mathcal{FS}}$. $\mathcal{A}_{\mathcal{HS}}$ simulates the two types of queries made by $\mathcal{A}_{\mathcal{FS}}$, namely, key generation oracle queries and signing oracle queries, as follows:

$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathcal{FS}.\mathsf{Setup}(1^\lambda)$

$\mathsf{pp} \leftarrow \mathcal{HS}.\mathsf{Setup}(1^\lambda)$
$(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow \mathcal{HS}.\mathsf{KGen}(\mathsf{pp})$
$(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathcal{HS}.\mathsf{KGen}(\mathsf{pp})$
// Without loss of generality, assume pk_b is
// assigned to the identity b for $b \in \{0,1\}$.
$\mathsf{mpk} := (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{sk}_1)$
$\mathsf{msk} := \mathsf{sk}_0$
return $(\mathsf{mpk}, \mathsf{msk})$

$\mathsf{sk}_f \leftarrow \mathcal{FS}.\mathsf{KGen}(\mathsf{msk}, f)$

$\sigma_f \leftarrow \mathcal{HS}.\mathsf{Sig}(\mathsf{sk}_0, (0, \mathsf{pk}_1), f)$
return $\mathsf{sk}_f := \sigma_f$

$(f(m), \sigma) \leftarrow \mathcal{FS}.\mathsf{Sig}(f, \mathsf{sk}_f, m)$

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

$b \leftarrow \mathcal{FS}.\mathsf{Vf}(\mathsf{mpk}, m, \sigma)$

parse σ **as** (τ, σ')
$\mathcal{P} := (U, (0, \mathsf{pk}_1), (1, \tau))$
$b \leftarrow \mathcal{HS}.\mathsf{Vf}(\mathcal{P}, \{\mathsf{pk}_0, \mathsf{pk}_1\}, m, \sigma')$
return b

Fig. 3. Construction of FS from M-HS

- $\mathcal{O}_{\mathsf{key}}(f, i)$
  - If there exists an entry for (f, i) in the dictionary, output the corresponding value sk_f^i.
  - Otherwise, query the signing oracle of $\mathcal{HS}$ to get
$$\sigma_f^i \leftarrow \mathcal{HS}.\mathsf{Sig}(\mathsf{sk}_{\mathcal{HS}}, (0, \mathsf{pk}_1), f).$$
    Then add $\mathsf{sk}_f^i = \sigma_f^i$ 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 $\mathsf{sk}_f^i = \sigma_f^i$.
  - Otherwise, query the signing oracle of $\mathcal{HS}$ to get σ_f^i as above. Then add $\mathsf{sk}_f^i = \sigma_f^i$ to the dictionary entry (f, i).
  - Finally, sample $\tau \leftarrow \{0,1\}^\lambda$ and compute $\sigma_m \leftarrow \mathcal{HS}.\mathsf{Sig}(\mathsf{sk}_1, (1, \tau), m)$. Let $\mathcal{P} := (U, (0, \mathsf{pk}_1), (1, \tau))$, $\eta_f := (\mathcal{I}_{0,\mathsf{pk}_1}, \mathsf{pk}_0, f, \sigma_f^i)$ and $\eta_m := (\mathcal{I}_{1,\tau}, \mathsf{pk}_1, m, \sigma_m)$. Compute $\sigma' \leftarrow \mathcal{HS}.\mathsf{Eval}(\mathcal{P}, (\eta_f, \eta_m))$ and output $(U(f,m), (\tau, \sigma'))$.

After querying the oracles, $\mathcal{A}_{\mathcal{FS}}$ responds with forgery (m^*, σ^*), where $\sigma^* = (\tau^*, \sigma'^*)$. $\mathcal{A}_{\mathcal{HS}}$ returns $(\mathcal{P} = (U, (0, \mathsf{pk}_1), (1, \tau^*)), \{\mathsf{pk}_0, \mathsf{pk}_1\}, m^*, \sigma'^*)$. It is a valid forgery of $\mathcal{HS}$, 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 of the function-privacy game. As $\mathcal{HS}$ is weakly context hiding, there exists a simulator $\mathcal{S}_{\mathcal{HS}}$ which, on input $(\mathcal{P} = (U, (0, \mathsf{pk}_1), (1, \tau)), \{\mathsf{pk}_0, \mathsf{pk}_1\}, f(m))$ for a random tag τ, 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 $\mathcal{A}_{\mathcal{FS}}$ guesses correctly in the simulated game is $\frac{1}{2}$.

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], for which security cannot be proven via a black-box reduction from falsifiable assumptions [36], we have the following corollary.

Corollary 1. *If cEUF-CMA-secure, weakly context hiding, and succinct* 1*-hop* 2*-HS for* NP *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 it relies on the existence of 2-HS, which might be more difficult to construct than (1-)HS (with unforgeability under corruption). Thus, in this section, we construct SNARGs directly from HS, making (M-)HS with unforgeability under corruption a notion sitting tightly and nicely in between SNARKs and SNARGs. This transformation also gives us zero-knowledge for free[8].

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. Let $\ell_x = (\mathsf{id}, \tau_x)$ and $\ell_w = (\mathsf{id}, \tau_w)$ be labels for arbitrary identity id and tags τ_x and τ_w. It then evaluates the signatures using a labeled program $\mathcal{P} = (g, \ell_x, \ell_w)$ which, on input (x, w), 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.* [45] 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 for any label space $\mathcal{L} = \mathcal{ID} \times \mathcal{T}$ where $\log |\mathcal{ID}| = \mathsf{poly}(\lambda)$ and $\log |\mathcal{T}| = \mathsf{poly}(\lambda)$. Let g be a function such that $g(x, w) = x$ if $R(x, w) = 1$, $\perp$ otherwise. Figure 4 shows our SNARG construction Π for NP language L with relation R. The completeness follows straightforwardly from the correctness of $\mathcal{HS}$.

[8] Function privacy of FS is very similar to zero-knowledge, except that the former is defined in "indistinguishability-style" while the latter is defined in "simulation-style".

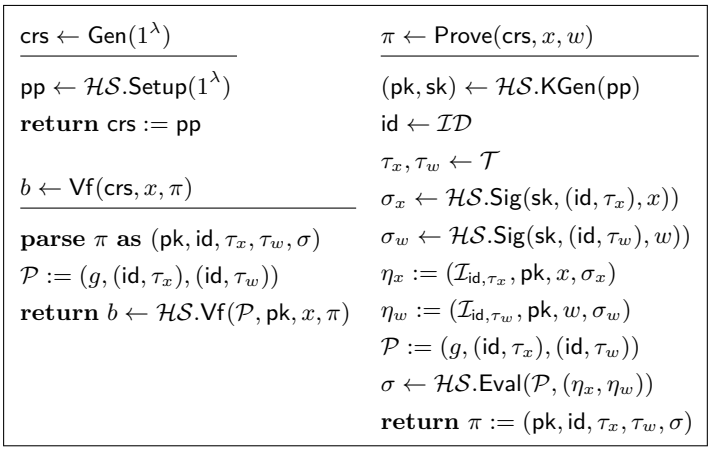

Fig. 4. Construction of SNARG from M-HS

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

Proof. If there exists an adversary $\mathcal{A}_\Pi$ that breaks the soundness of Π with non-negligible probability, we can construct an adversary $\mathcal{A}_{\mathcal{HS}}$ that uses $\mathcal{A}_\Pi$ to produce a forgery of $\mathcal{HS}$. $\mathcal{A}_{\mathcal{HS}}$ acts as a challenger in the soundness game of Π.

$\mathcal{A}_{\mathcal{HS}}$ receives pp from the challenger of the $\mathsf{cEUF\text{-}CMA}$ game of $\mathcal{HS}$, and forwards the common reference string $\mathsf{crs} := \mathsf{pp}$ to $\mathcal{A}_\Pi$. Eventually, $\mathcal{A}_\Pi$ responds with (x^*, π^*) such that $\mathsf{Vf}(\mathsf{crs}, x^*, \pi^*) = 1$ but $x^* \notin L$. $\mathcal{A}_{\mathcal{HS}}$ then parses $\pi^* = (\mathsf{pk}^*, \mathsf{id}^*, \tau_x^*, \tau_w^*, \sigma^*)$, and answers $(\mathcal{P}^* = (g, (\mathsf{id}^*, \tau_x^*), (\mathsf{id}^*, \tau_w^*)), \mathsf{pk}^*, x^*, \sigma^*)$ to its $\mathsf{cEUF\text{-}CMA}$ 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 Π is zero-knowledge.*

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 $(\mathcal{P} = (g, (\mathsf{id}, \tau_x), (\mathsf{id}, \tau_w)), \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}}, x)$, for some arbitrary id, τ_x, and τ_w, 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 an HS public key, an identity, two tags, all of which has polynomial length, and a signature of $\mathcal{HS}$. By the succinctness of $\mathcal{HS}$, the signature size is also bounded by a polynomial.

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 M-HS schemes which are unforgeable under corruption in the standard model. Interestingly, the only existing M-HS scheme [29] is unforgeable but without corruption in the standard model.

7 Conclusion and Open Problem

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 reasonable) security model of M-HS, with constructions from standard assumptions.

Acknowledgments

Sherman S. M. Chow is supported by the General Research Fund (CUHK 14210217) of the Research Grants Council, University Grant Committee of Hong Kong.

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

References

1. Ahn, J.H., Boneh, D., Camenisch, J., Hohenberger, S., shelat, a., Waters, B.: Computing on authenticated data. In: Cramer, R. (ed.) TCC 2012: 9th Theory of Cryptography Conference. Lecture Notes in Computer Science, vol. 7194, pp. 1–20. Springer, Heidelberg (Mar 2012)
2. Asharov, G., Jain, A., López-Alt, A., Tromer, E., Vaikuntanathan, V., Wichs, D.: Multiparty computation with low communication, computation and interaction via threshold FHE. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology – EUROCRYPT 2012. Lecture Notes in Computer Science, vol. 7237, pp. 483–501. Springer, Heidelberg (Apr 2012)
3. Attrapadung, N., Libert, B., Peters, T.: Computing on authenticated data: New privacy definitions and constructions. In: Wang, X., Sako, K. (eds.) Advances in Cryptology – ASIACRYPT 2012. Lecture Notes in Computer Science, vol. 7658, pp. 367–385. Springer, Heidelberg (Dec 2012)
4. Attrapadung, N., Libert, B., Peters, T.: Efficient completely context-hiding quotable and linearly homomorphic signatures. In: Kurosawa and Hanaoka [42], pp. 386–404
5. Backes, M., Dagdelen, Ö., Fischlin, M., Gajek, S., Meiser, S., Schröder, D.: Operational signature schemes. Cryptology ePrint Archive, Report 2014/820 (2014)
6. Backes, M., Meiser, S., Schröder, D.: Delegatable functional signatures. In: Cheng et al. [21], pp. 357–386
7. Bellare, M., Fuchsbauer, G.: Policy-based signatures. In: Krawczyk [41], pp. 520–537
8. Bethencourt, J., Boneh, D., Waters, B.: Cryptographic methods for storing ballots on a voting machine. In: ISOC Network and Distributed System Security Symposium – NDSS 2007. The Internet Society (Feb / Mar 2007)

9. Boneh, D., Freeman, D., Katz, J., Waters, B.: Signing a linear subspace: Signature schemes for network coding. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009: 12th International Conference on Theory and Practice of Public Key Cryptography. Lecture Notes in Computer Science, vol. 5443, pp. 68–87. Springer, Heidelberg (Mar 2009)
10. Boneh, D., Freeman, D.M.: Homomorphic signatures for polynomial functions. In: Paterson, K.G. (ed.) Advances in Cryptology – EUROCRYPT 2011. Lecture Notes in Computer Science, vol. 6632, pp. 149–168. Springer, Heidelberg (May 2011)
11. Boneh, D., Freeman, D.M.: Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In: Catalano et al. [18], pp. 1–16
12. Boneh, D., Gentry, C., Gorbunov, S., Halevi, S., Nikolaenko, V., Segev, G., Vaikuntanathan, V., Vinayagamurthy, D.: Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In: Nguyen and Oswald [48], pp. 533–556
13. Boneh, D., Lewi, K., Montgomery, H.W., Raghunathan, A.: Key homomorphic PRFs and their applications. In: Canetti, R., Garay, J.A. (eds.) Advances in Cryptology – CRYPTO 2013, Part I. Lecture Notes in Computer Science, vol. 8042, pp. 410–428. Springer, Heidelberg (Aug 2013)
14. Boneh, D., Segev, G., Waters, B.: Targeted malleability: homomorphic encryption for restricted computations. In: Goldwasser, S. (ed.) ITCS 2012: 3rd Innovations in Theoretical Computer Science. pp. 350–366. Association for Computing Machinery (Jan 2012)
15. Boyen, X., Fan, X., Shi, E.: Adaptively secure fully homomorphic signatures based on lattices. Cryptology ePrint Archive, Report 2014/916 (2014)
16. Boyle, E., Goldwasser, S., Ivan, I.: Functional signatures and pseudorandom functions. In: Krawczyk [41], pp. 501–519
17. Brakerski, Z., Kalai, Y.T.: A framework for efficient signatures, ring signatures and identity based encryption in the standard model. Cryptology ePrint Archive, Report 2010/086 (2010)
18. Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.): PKC 2011: 14th International Conference on Theory and Practice of Public Key Cryptography, Lecture Notes in Computer Science, vol. 6571. Springer, Heidelberg (Mar 2011)
19. Catalano, D., Fiore, D., Warinschi, B.: Efficient network coding signatures in the standard model. In: Fischlin et al. [31], pp. 680–696
20. Catalano, D., Fiore, D., Warinschi, B.: Homomorphic signatures with efficient verification for polynomial functions. In: Garay, J.A., Gennaro, R. (eds.) Advances in Cryptology – CRYPTO 2014, Part I. Lecture Notes in Computer Science, vol. 8616, pp. 371–389. Springer, Heidelberg (Aug 2014)
21. Cheng, C.M., Chung, K.M., Persiano, G., Yang, B.Y. (eds.): PKC 2016: 19th International Conference on Theory and Practice of Public Key Cryptography, Part I, Lecture Notes in Computer Science, vol. 9614. Springer, Heidelberg (Mar 2016)
22. Cheon, J.H., Takagi, T. (eds.): Advances in Cryptology – ASIACRYPT 2016, Part II, Lecture Notes in Computer Science, vol. 10032. Springer, Heidelberg (Dec 2016)
23. Chow, S.S.M.: Functional credentials for Internet of things. In: Chow, R., Saldamli, G. (eds.) Proceedings of the 2nd ACM International Workshop on IoT Privacy, Trust, and Security, IoTPTS@AsiaCCS, Xi'an, China, May 30, 2016. p. 1. ACM (2016)
24. Chow, S.S.M., Haralambiev, K.: Non-interactive confirmer signatures. In: Kiayias, A. (ed.) Topics in Cryptology – CT-RSA 2011. Lecture Notes in Computer Science, vol. 6558, pp. 49–64. Springer, Heidelberg (Feb 2011)

25. Chow, S.S.M., Wei, V.K.W., Liu, J.K., Yuen, T.H.: Ring signatures without random oracles. In: Lin, F.C., Lee, D.T., Lin, B.S., Shieh, S., Jajodia, S. (eds.) ASIACCS 06: 1st ACM Symposium on Information, Computer and Communications Security. pp. 297–302. ACM Press (Mar 2006)
26. Danezis, G., Fournet, C., Groth, J., Kohlweiss, M.: Square span programs with applications to succinct NIZK arguments. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology – ASIACRYPT 2014, Part I. Lecture Notes in Computer Science, vol. 8873, pp. 532–550. Springer, Heidelberg (Dec 2014)
27. Derler, D., Ramacher, S., Slamanig, D.: Homomorphic proxy re-authenticators and applications to verifiable multi-user data aggregation. In: Kiayias, A. (ed.) Financial Cryptography and Data Security - 21st International Conference, FC 2017, Sliema, Malta, April 3-7, 2017, Revised Selected Papers. Lecture Notes in Computer Science, vol. 10322, pp. 124–142. Springer (2017)
28. Derler, D., Slamanig, D.: Key-homomorphic signatures and applications to multiparty signatures and non-interactive zero-knowledge. Cryptology ePrint Archive, Report 2016/792 (2016)
29. Fiore, D., Mitrokotsa, A., Nizzardo, L., Pagnin, E.: Multi-key homomorphic authenticators. In: Cheon and Takagi [22], pp. 499–530
30. Fiore, D., Nitulescu, A.: On the (in)security of SNARKs in the presence of oracles. In: Hirt, M., Smith, A.D. (eds.) TCC 2016-B: 14th Theory of Cryptography Conference, Part I. Lecture Notes in Computer Science, vol. 9985, pp. 108–138. Springer, Heidelberg (Oct / Nov 2016)
31. Fischlin, M., Buchmann, J., Manulis, M. (eds.): PKC 2012: 15th International Conference on Theory and Practice of Public Key Cryptography, Lecture Notes in Computer Science, vol. 7293. Springer, Heidelberg (May 2012)
32. Freeman, D.M.: Improved security for linearly homomorphic signatures: A generic framework. In: Fischlin et al. [31], pp. 697–714
33. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct NIZKs without PCPs. In: Johansson, T., Nguyen, P.Q. (eds.) Advances in Cryptology – EUROCRYPT 2013. Lecture Notes in Computer Science, vol. 7881, pp. 626–645. Springer, Heidelberg (May 2013)
34. Gennaro, R., Katz, J., Krawczyk, H., Rabin, T.: Secure network coding over the integers. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010: 13th International Conference on Theory and Practice of Public Key Cryptography. Lecture Notes in Computer Science, vol. 6056, pp. 142–160. Springer, Heidelberg (May 2010)
35. Gennaro, R., Wichs, D.: Fully homomorphic message authenticators. In: Sako, K., Sarkar, P. (eds.) Advances in Cryptology – ASIACRYPT 2013, Part II. Lecture Notes in Computer Science, vol. 8270, pp. 301–320. Springer, Heidelberg (Dec 2013)
36. Gentry, C., Wichs, D.: Separating succinct non-interactive arguments from all falsifiable assumptions. In: Fortnow, L., Vadhan, S.P. (eds.) 43rd Annual ACM Symposium on Theory of Computing. pp. 99–108. ACM Press (Jun 2011)
37. Gorbunov, S., Vaikuntanathan, V., Wichs, D.: Leveled fully homomorphic signatures from standard lattices. In: Servedio, R.A., Rubinfeld, R. (eds.) 47th Annual ACM Symposium on Theory of Computing. pp. 469–477. ACM Press (Jun 2015)
38. Groth, J.: On the size of pairing-based non-interactive arguments. In: Fischlin, M., Coron, J.S. (eds.) Advances in Cryptology – EUROCRYPT 2016, Part II. Lecture Notes in Computer Science, vol. 9666, pp. 305–326. Springer, Heidelberg (May 2016)

39. Johnson, R., Molnar, D., Song, D.X., Wagner, D.: Homomorphic signature schemes. In: Preneel, B. (ed.) Topics in Cryptology – CT-RSA 2002. Lecture Notes in Computer Science, vol. 2271, pp. 244–262. Springer, Heidelberg (Feb 2002)
40. Kiltz, E., Mityagin, A., Panjwani, S., Raghavan, B.: Append-only signatures. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005: 32nd International Colloquium on Automata, Languages and Programming. Lecture Notes in Computer Science, vol. 3580, pp. 434–445. Springer, Heidelberg (Jul 2005)
41. Krawczyk, H. (ed.): PKC 2014: 17th International Conference on Theory and Practice of Public Key Cryptography, Lecture Notes in Computer Science, vol. 8383. Springer, Heidelberg (Mar 2014)
42. Kurosawa, K., Hanaoka, G. (eds.): PKC 2013: 16th International Conference on Theory and Practice of Public Key Cryptography, Lecture Notes in Computer Science, vol. 7778. Springer, Heidelberg (Feb / Mar 2013)
43. Lamport, L.: Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory (Oct 1979)
44. Libert, B., Peters, T., Joye, M., Yung, M.: Linearly homomorphic structure-preserving signatures and their applications. In: Canetti, R., Garay, J.A. (eds.) Advances in Cryptology – CRYPTO 2013, Part II. Lecture Notes in Computer Science, vol. 8043, pp. 289–307. Springer, Heidelberg (Aug 2013)
45. Libert, B., Peters, T., Joye, M., Yung, M.: Non-malleability from malleability: Simulation-sound quasi-adaptive NIZK proofs and CCA2-secure encryption from homomorphic signatures. In: Nguyen and Oswald [48], pp. 514–532
46. Lipmaa, H.: Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. In: Sako, K., Sarkar, P. (eds.) Advances in Cryptology – ASIACRYPT 2013, Part I. Lecture Notes in Computer Science, vol. 8269, pp. 41–60. Springer, Heidelberg (Dec 2013)
47. Micali, S.: Computationally sound proofs. SIAM J. Comput. 30(4), 1253–1298 (2000)
48. Nguyen, P.Q., Oswald, E. (eds.): Advances in Cryptology – EUROCRYPT 2014, Lecture Notes in Computer Science, vol. 8441. Springer, Heidelberg (May 2014)
49. Rompel, J.: One-way functions are necessary and sufficient for secure signatures. In: 22nd Annual ACM Symposium on Theory of Computing. pp. 387–394. ACM Press (May 1990)
50. Steinfeld, R., Bull, L., Wang, H., Pieprzyk, J.: Universal designated-verifier signatures. In: Laih, C.S. (ed.) Advances in Cryptology – ASIACRYPT 2003. Lecture Notes in Computer Science, vol. 2894, pp. 523–542. Springer, Heidelberg (Nov / Dec 2003)

A Insecurity of Existing Work against Insider Attack

We briefly explain why the existing construction of M-HS by Fiore *et al.* [29] suffers from insider attacks. Since their construction is a multi-key generalization of the (single-key) HS by Gorbunov *et al.* [37], we first demonstrate how the attack works in the single-key setting, then generalize it to the multi-key setting.

The HS construction by Gorbunov *et al.* [37] 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 preimage u to an image v. The function is homomorphic in the following sense: Given a function g and some preimages 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 preimage $u_{g(x_1,\ldots,x_N)}$. 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 computations 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 preimages correspond to the signatures.

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

Generalizing, a multi-key homomorphic trapdoor function f (constructed implicitly in [29]) maps a set of public keys $\{\mathsf{pk}_i\}_{i\in[M]}$, an index x, and a preimage u to an image v. The knowledge of a secret key sk corresponding to any pk in $\{\mathsf{pk}_i\}_{i\in[M]}$ suffices to invert f on the tuple (v, x) with respect to $\{\mathsf{pk}_i\}_{i\in[M]}$. As a result, if any of the M signers is corrupt, an adversary can generate signatures that disrespect the messages signed by the other honest signers.