A Zoo of Homomorphic Signatures
Multi-Key and Key-Homomorphism

Russell W. F. Lai, Raymond K. H. Tai, Harry W. H. Wong, Sherman S. M. Chow

Department of Information Engineering
The Chinese University of Hong Kong
Sha Tin, N.T., Hong Kong
{russell,raymondtai,whwong,sherman}@ie.cuhk.edu.hk

Abstract. Homomorphic signatures (HS) allow evaluation of signed messages by producing a signature on a function of messages signed by the same key. Motivated by the vast potential of applications, we initiate the study of multi-key HS (M-HS) which allows evaluation of signatures under different keys. We also study other multi-key extensions, namely, hierarchical HS (M-HiHS) for delegation of signing power over message sub-spaces, and key-message-HS (M-KMHS) for evaluation of signatures under different keys with respect to both keys and messages. We thus also introduce the concept of key-homomorphism in signatures, which leads to the notion of multi-key key-HS (M-KHS) for evaluation of signatures with respect to keys only.

Notion-wise, our result shows that M-HS can act as a central notion since all its seemingly different extensions are all equivalent. In particular, this suggests that key-homomorphism and message-homomorphism in signatures are identical in nature. As a sample application, we show that M-KHS implies decentralized attribute-based signatures (D-ABS). Our work also provides the first (leveled) fully KHS and the first (D-)ABS for circuits from standard assumptions.

Surprisingly, there is a huge gap between homomorphism in a single space and in two spaces. Indeed all existing (leveled) fully homomorphic signature schemes support only a single signer. In the multi-space setting, we construct M-HS from any adaptive zero-knowledge succinct non-interactive argument of knowledge (ZK-SNARK) (and other standard assumptions). We also show that two-key HS implies functional signatures. Our study equips the literature with a suite of signature schemes allowing different kinds of flexible evaluations.

1 Introduction

Homomorphic signatures (HS) allow evaluation of signed messages by producing a signature on a function of messages signed by the same key. HS has undergone great development, notably from supporting either only addition or multiplication [BFKW09, GKKR10, BF11b, CFW12, Fre12, LPJY13], to bounded-degree polynomials [BF11a, CFW14], and even to (leveled) fully homomorphic operations which allow evaluation of general circuits of a-priori bounded depth [GVW15, BFS14]. Beyond unforgeability, some works consider stronger privacy notions such as context-hiding [ABC$^+$12, ALP12, ALP13]. This supports applications which require computation on authenticated data, for example, verifiable computation. Yet, the state-of-the-art HS is still restricted to the single-key setting. Also, as an indirect consequence, the homomorphism is restricted to only the message space. In this paper, we initiate the study of HS in the multi-key setting, as well as key-homomorphism in (message-)homomorphic signatures in HS.

To illustrate the usefulness of multi-key HS which is homomorphic in both key space and message space, consider a scenario where two managers Alice and Bob in a company are jointly making a decision. They sign their decisions m_1 and m_2 with their signing keys associated with attributes x_1 and x_2 respectively. Is it possible for a secretary Charlie to derive a signature on their joint decision, and prove that it is from combining those of the two managers? Technically, we ask whether a public evaluator can derive a signature of $g((x_1, m_1), (x_2, m_2))$ under the attribute $f(x_1, x_2)$, for some functions f and g. In other words, such a signature scheme supports operations on both the key space and message space. In particular, if $x_1 = x_2$, $f(x_1, x_2) = x_1$ and $g((x_1, m_1), (x_2, m_2)) = g'(m_1, m_2)$, the above syntax captures (regular) homomorphic

signatures. On the other hand, if $m_1 = m_2$ and $g((x_1, m_1), (x_2, m_2)) = m_1$, it becomes a signature scheme with homomorphism in the key space, or what we call key-homomorphic signatures (KHS), which itself is a new notion never considered explicitly in the literature.

To unlock the full potential of key-homomorphism, it will be useful to first consider multi-key homomorphic signatures (M-HS), where the operation on the public keys is fixed to the union function. The difference between M-HS and HS is similar to that between multi-signatures and vanilla signatures. In M-HS, users with different secret and public key pairs can sign on different messages, so that a public evaluator can perform homomorphic evaluations of a function g on all message-signature pairs to produce a signature which verifies under the union of all input public keys. A distinctive property of M-HS, which is absent in the single-key setting, is that we require M-HS to be secure against *insider attack*. That is, suppose signer A signs a message m_A while the evaluator colludes with another signer B, we require this coalition can only produce signatures on $(g, g(m_A, \cdot))$. In other words, signer B cannot falsely claim that m is computed from some value m'_A not contributed by signer A.

As an immediate application, M-HS can be used along with multi-key homomorphic encryption [LTV12, CM15, MW16, PS16] to achieve verifiable secure multi-party computation. Apart from being a powerful primitive on its own, M-HS implies several of its seeming different extensions and variants, including multi-key hierarchical homomorphic signatures (M-HiHS), multi-key key-message-homomorphic signatures (M-KMHS), and multi-key key-homomorphic signatures (M-KHS), whose functionality will be explained below.

1.1 Our Results

Multi-key homomorphic signatures. We present multi-key homomorphic signatures (M-HS), a generalization of homomorphic signatures which allows a public evaluator to transform signatures of different set of messages M signed under different public keys to a signature of $(g, g(\ldots, M, \ldots))$ signed under a combined public key. We define a strong security notion of M-HS called unforgeability under K-bounded corruption, which requires that a coalition of the evaluator and any K signers cannot forge a signature of (g, m), where the resulting message m is not in the range of g restricted by the inputs of the honest signers. Interestingly, such a definition also makes sense in the single-key setting, where we require that even the signer cannot produce a signature on (g, m) where m is not in the range of g.

Implications to and from existing notions. Treated as a central hub of our work, we study how M-HS is related to other notions. First, we show that M-HS can be constructed from any adaptive zero-knowledge succinct non-interactive argument of knowledge (ZK-SNARK). Then, we show that functional signatures (FS) [BGI14] can be constructed from a two-key M-HS (2-HS). Since the existence of succinct functional signatures implies the existence of succinct non-interactive argument (SNARG), we obtain as a corollary that the existence of 2-HS implies the existence of SNARG. Even stronger, we show that the existence of single-key HS which is unforgeable under corruption implies the existence of ZK-SNARG.

Equivalence of seemingly different variants. We then proceed to the relations between M-HS and its extensions and variants. Concretely, we propose several seemingly different notions listed as follows:

- Multi-Key Hierarchical Homomorphic Signatures (M-HiHS): extends M-HS such that a signer can delegate its signing power over any supersets of a specified set M.
- Multi-Key Key-Message-Homomorphic Signatures (M-KMHS): extends M-HS such a public evaluator can evaluate signatures of a set of messages M signed by a key associated to a set of attributes X to a signature of $g(X, M)$ signed by a key associated to the attribute $f(X)$.
- Multi-Key Key-Homomorphic Signatures (M-KHS): allows a public evaluator to evaluate signatures of M signed by a key associated to attributes X to one signed by a key associated to the attribute $f(X)$.

Although these notions are seemingly stronger than or at least different from M-HS, we show that their existence are all equivalent to that of M-HS (up to the existence of collision resistant hash for M-HiHS). In particular, their corresponding special cases, HS and KHS, are also equivalent. Conceptually, this result

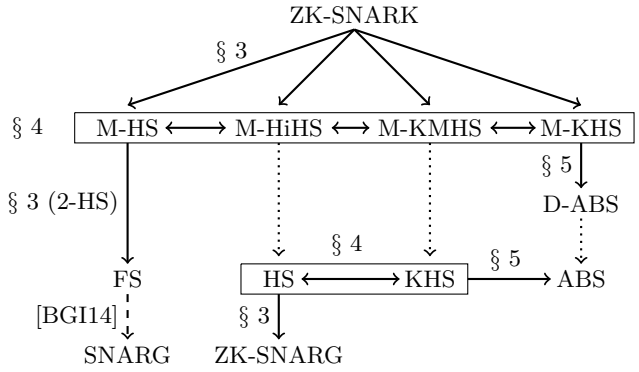

Fig. 1. Relations between all notions considered in this work: Solid arrows represent implications shown in this work. Dashed and dotted arrows represent existing and trivial implications respectively.

bridges the gap between message- and key-homomorphism in the signatures setting. When instantiated by state-of-the-art (leveled) fully homomorphic signatures [GVW15], we obtain (leveled) fully key-homomorphic signatures (FKHS) which inherit all the nice properties, such as security based on (standard) lattices in the standard model.

Applications. Apart from grand applications such as multi-party verifiable computation, we are also interested in the implications of M-HS to more basic primitives. As an example, we show that (linkable) decentralized attribute-based signatures (D-ABS) can be constructed from M-KHS generically. When instantiated by the FKHS above, we obtain ABS for general circuits, for which the only known efficient constructions are based on multilinear maps. Our scheme satisfies linkable anonymity. That is, while a signature does not leak any information about the attributes under which it is signed beyond whether they satisfy the policy of the verifier, different signatures from the same signer are publicly linkable. By a generic transformation using non-interactive witness-indistinguishable (NIWI) proofs, the resulting scheme achieves full anonymity.

To summarize, Figure 1 illustrates the relations between all notions studied in this work.

1.2 Related Work on Key-Homomorphism

Key-homomorphism has been studied in the context of key-homomorphic encryption (KHE) by Boneh *et al.* [BGG+14], who formulated KHE and constructed it based on the learning with errors problem. Furthermore, they used KHE to construct attribute-based encryption for general circuits with short secret keys. Inspired by their work, we study the corresponding notions in the signatures setting, namely KHS, and attribute-based signatures (ABS) for general circuits.

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 public keys. As the plaintexts are private while the public keys are public, KHE and HE are inherently different. However, for signature schemes, both the messages and the public keys are public. It is natural to ask whether there is any connection between HS and KHS. Indeed, we show that the two notions are equivalent.

Key-homomorphism in signatures is considered in different extents in delegatable functional signatures (DFS) [BMS16] and operational signature scheme (OSS) [BDF+14]. 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 [BF14], in which a policy-dependent signing key can only sign messages satisfying the specified policy, and functional signatures [BGI14], in which a functional signing key can only sign messages in the range of the specified function.

2 Preliminaries

Let λ be the security parameter. We use $\mathsf{negl}(\lambda)$ to denote functions which are negligible in λ. Barred variables are vectors, *e.g.*, $\bar{x}$. If A is a probabilistic algorithm, $x \leftarrow \mathsf{A}(\cdot)$ denotes assigning the output from the execution of A to the variable x. For a set S, $x \leftarrow S$ denotes the sampling of a uniformly random $x \in S$.

2.1 Digital Signatures

Definition 1 (Signatures). *A signature scheme is a tuple of PPT algorithms $\mathcal{DS}.(\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ defined as follows:*

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

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

Definition 2 (Existential Unforgeability). *A signature scheme $\mathcal{DS}.(\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ is* existentially unforgeable under chosen message attacks (EUF-CMA-secure) *if, for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that $\Pr[\mathbf{expEUF}^{\mathcal{A}}_{\mathcal{DS}}(1^\lambda) = 1] \leq \mathsf{negl}(\lambda)$ where $\mathbf{expEUF}^{\mathcal{A}}_{\mathcal{DS}}$ is an experiment defined as follows:*

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

2.2 Adaptive Zero-Knowledge Non-Interactive Argument (of Knowledge)

Definition 3 (ZK-SNARG). *$\Pi = (\mathsf{Gen}, \mathsf{Prove}, \mathsf{Vf})$ is an* adaptive zero-knowledge succinct non-interactive argument *(ZK-SNARG) for a language $L \in \mathsf{NP}$ with the witness relation $\mathcal{R}$ if it satisfies the following properties:*

- ***Completeness**: For all x, w such that $\mathcal{R}(x, w) = 1$, and for all strings $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda)$, we have $\mathsf{Vf}(\mathsf{crs}, x, \mathsf{Prove}(x, w, \mathsf{crs})) = 1$.*
- ***Adaptive Soundness**: If $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda)$ is sampled uniformly at random, then for all PPT adversaries $\mathcal{A}$, the probability that $\mathcal{A}(\mathsf{crs})$ will output a pair (x, π) such that $x \notin L$ but $\mathsf{Vf}(\mathsf{crs}, x, \pi) = 1$, is at most $\mathsf{negl}(\lambda)$.*
- ***Succinctness**: For all x, w such that $\mathcal{R}(x, w) = 1$, if $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda)$ and $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)$, then there exists an universal polynomial $p(\cdot)$ that does not depend on the relation $\mathcal{R}$, such that $|\pi| \leq p(k + \log t)$, where t denotes the runtime of the relation R associated with language L.*

– ***Adaptive Zero-Knowledge****: There exists a PPT algorithm* $\mathcal{S} = (\mathcal{S}^{\mathsf{crs}}, \mathcal{S}^{\mathsf{Prove}})$ *such that, for all PPT adversaries* $\mathcal{A}$*,*

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

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

Definition 4 ((Strong) ZK-SNARK [BGI14, FN16]). *A ZK-SNARK* $\Pi = (\mathsf{Gen},\mathsf{Prove},\mathsf{Vf})$ *is a (strong)* adaptive zero-knowledge succinct non-interactive argument of knowledge *(ZK-SNARK) for a language* L *in* NP *with witness relation* $\mathcal{R}$ *if there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that, for all PPT provers* P^**, there exists a PPT algorithm* $\mathsf{E}_{P^*} = (\mathsf{E}^1_{P^*}, \mathsf{E}^2_{P^*})$ *such that for* every *adversaries* $\mathcal{A}$*,*

$$|\Pr[\mathcal{A}(\mathsf{crs}) \to 1 : \mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda)] - \Pr[\mathcal{A}(\mathsf{crs}) \to 1 : (\mathsf{crs},\mathsf{td}) \leftarrow \mathsf{E}^1_{P^*}(1^\lambda)]| = \mathsf{negl}(\lambda)\,, \text{and}$$

$$|\Pr[(x,\pi) \leftarrow P^*(\mathsf{crs}) \ \wedge \ w^* \leftarrow \mathsf{E}^2_{P^*}(\mathsf{crs},\mathsf{xk},x,\pi) \ s.t. \ \mathsf{Vf}(\mathsf{crs},x,\pi) = 1 \wedge (x,w^*) \notin \mathcal{R}]| = \mathsf{negl}(\lambda)\,,$$

where the probabilities are taken over $(\mathsf{crs},\mathsf{xk}) \leftarrow \mathsf{E}^1_{P^*}(1^\lambda)$*, and the random coin of the extractor* $\mathsf{E}^2_{P^*}$*.*

3 Multi-Key Homomorphic Signatures (M-HS)

We define a new primitive called multi-key homomorphic signatures (M-HS). M-HS allows an arbitrary number of signers to generate keys and sign messages independently in a fully distributed manner. Suppose that each signer k signs a set of messages M_k with its secret key sk_k to produce a set of signatures Σ_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 K secret keys are owned by the same party.

In the multi-signer setting, we must carefully analyze unforgeability when the adversary can corrupt some signers. Such an insider attack is absent in HS since there is only one signer and hence one signing key involved with a signature. We formulate the unforgeability against insider corruption, which requires that such group of corrupt signers cannot produce signatures of (m, g), where the message m is outside the range of the function g restricted by the inputs of the uncorrupted signers.

To illustrate the meaning of a successful forgery, consider the following specific configuration: Let g be the product function and $M_k \in \{0, 1\}$. As long as $M_k = 0$ for some uncorrupted signer k, the adversary is unable to produce a signature of $(1, g)$.

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

3.1 Settings and Notations

Before formally defining the notion, we first establish the settings and notations to be used throughout the paper. Let $N = \mathsf{poly}(\lambda)$ be an a-priori bounded maximum number of hops of homomorphic evaluation. That is, a freshly signed message m can at most pass through N functions. There is however no further restriction on the number of inputs and circuit depth of each function as long as it is efficiently computable. We assume that functions obtained by concatenation, such as $g \circ \bar{h}$, keep the concatenated representation (without simplification). Thus, given a function g, one can efficiently write down the evaluation process of a function g as a tree, called the evaluation tree of g. As a consequence, there exists a polynomial time algorithm Hop which, on input a function g, outputs the depth of the evaluation tree of g, which represents the number of hops $\mathsf{Hop}(g)$ passed through in the evaluation process of g. We abuse the notation $\mathsf{pk} = g(\mathsf{pk}_1, \ldots, \mathsf{pk}_K)$ to represent a tree similar to the evaluation tree of g, except that the tree nodes are replaced by the public keys pk_k of the corresponding inputs. We also abuse the notion $\tau = g(\tau_1, \ldots, \tau_K)$ in a similar way.

3.2 Definitions

Syntax. A multi-key homomorphic signature scheme (M-HS) for a class $\mathcal{G}$ of admissible functions consists of the PPT algorithms $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf}, \mathsf{Eval})$ defined as follows:

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ inputs the security parameter λ. It outputs a public parameter pp which is input to all algorithms implicitly. The public parameter also defines the message space $\mathcal{M}$ and the function family $\mathcal{G}$ which contains the identity function $\mathsf{id} : \mathcal{M} \to \mathcal{M}$.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(\mathsf{pp})$ inputs the public parameter. It outputs the public key pk and secret key sk. When an algorithm takes as input a secret key sk, we assume sk contains its corresponding public pk implicitly.
- $\Sigma \leftarrow \mathsf{Sig}(\mathsf{sk}, \tau, M)$ inputs the secret key sk, a tag $\tau \in \{0,1\}^*$, and a set of messages $M \in \mathcal{M}^*$. It outputs a set of signatures Σ.
- $b \leftarrow \mathsf{Vf}(\mathsf{pk}, \tau, \sigma, m, g)$ inputs a (possibly combined) public key pk, a (possibly combined) tag τ, a signature σ, a message $m \in \mathcal{M}$, and a function $g \in \mathcal{G}$. It outputs a bit $b = 0$ or $b = 1$, indicating whether m is the output of the function g over some signed data under tag τ.
- $\sigma \leftarrow \mathsf{Eval}(g, (\mathsf{pk}_k, \tau_k, M_k, \Sigma_k)_{k=1}^K)$ inputs a function $g \in \mathcal{G}$, and, from each contributor, a public key, a tag, a set of messages, and a set of signatures $(\mathsf{pk}_k, \tau_k, M_k, \Sigma_k)$, where $k \in [K]$. It outputs a signature σ signing the evaluated data $m = g(M_1, \ldots, M_K)$ under the combined public key $\mathsf{pk} = g(\mathsf{pk}_1, \ldots, \mathsf{pk}_K)$ and combined tags $\tau = g(\tau_1, \ldots, \tau_K)$. We assume that the evaluator has knowledge of the history of the evaluated functions which produce M_k.

Correctness. For any $\mathsf{pp} \in \mathsf{Setup}(1^\lambda)$, any $K, j = \mathsf{poly}(\lambda)$ and $k \in [K]$, any $(\mathsf{pk}_k, \mathsf{sk}_k) \in \mathsf{KGen}(\mathsf{pp})$, any $M_k \in \mathcal{M}^*$, any $\tau_k \in \{0,1\}^*$, and any $g, h_{k,j} \in \mathcal{G}$ with appropriate dimensions, it holds that

- (Signing Correctness.) if $\Sigma_k \leftarrow \mathsf{Sig}(\mathsf{sk}_k, \tau_k, M_k)$, then $\mathsf{Vf}(\mathsf{pk}_k, \tau_k, \sigma_{k,j}, m_{k,j}, \mathsf{id}) = 1$ for each $\sigma_{k,j} \in \Sigma_k$ and $m_{k,j} \in M_k$; furthermore,
- (Evaluation Correctness.) for any Σ_k, if $\mathsf{Vf}(\mathsf{pk}_k, \tau_k, \sigma_{k,j}, m_{k,j}, h_{k,j}) = 1$ for all $\sigma_{k,j} \in \Sigma_k$ and $m_{k,j} \in M_k$, $\mathsf{Hop}(h_{k,j}) \leq N - 1$, and $\sigma \leftarrow \mathsf{Eval}(g, (\mathsf{pk}_k, \tau_k, M_k, \Sigma_k)_{k=1}^K)$, then $\mathsf{Vf}(\mathsf{pk}, \tau, \sigma, g(M_1, \ldots, M_k), g \circ h) = 1$, where $\mathsf{pk} = g(\mathsf{pk}_1, \ldots, \mathsf{pk}_K)$ and $\tau = g(\tau_1, \ldots, \tau_K)$.

Unforgeability. Consider the following security game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

- Let $\mathfrak{K} = \mathsf{poly}(\lambda)$ be the number of signers in the system.
- The challenger $\mathcal{C}$ runs $\mathsf{pp} \in \mathsf{Setup}(1^\lambda)$ and $\{(\mathsf{pk}_k, \mathsf{sk}_k) \leftarrow \mathsf{KGen}(\mathsf{pp})\}_{k=1}^{\mathfrak{K}}$, and gives $\mathsf{pp}, \mathsf{pk}_1, \ldots, \mathsf{pk}_{\mathfrak{K}}$ to $\mathcal{A}$.
- The adversary $\mathcal{A}$ adaptively issues a polynomial number of signing queries and corruption queries. In each signing query $q \in [Q]$ where $Q = \mathsf{poly}(\lambda)$, $\mathcal{A}$ chooses a signer k, a tag $\tau_{q,k} \in \{0,1\}^*$, and a set of data $M_{q,k} \in \mathcal{M}^*$. The challenger $\mathcal{C}$ responds with $\Sigma_{q,k} \leftarrow \mathsf{Sig}(\mathsf{sk}_k, \tau_{q,k}, M_{q,k})$. In each corruption query, $\mathcal{A}$ chooses a signer k. The challenger $\mathcal{C}$ responds with sk_k.
- Without loss of generality, we assume that the adversary $\mathcal{A}$ corrupts signers $1, \ldots, K$.
- The adversary $\mathcal{A}$ outputs a public key pk^*, a tag $\tau^* \in \{0,1\}^*$, a function $g^* \in \mathcal{G}$, a message $m^* \in \mathcal{M}$, and a signature σ^*. It wins the game if $\mathsf{Vf}(\mathsf{pk}^*, \tau^*, \sigma^*, m^*, g^*) = 1$, pk^* is produced by evaluating g on some subset of $\{\mathsf{pk}_k\}$, and $(\mathsf{pk}^*, \tau^*, \sigma^*, m^*, g^*)$ is either a forgery such that
  - *Type-I*: τ^* is not a root produced by evaluating g on any subset of $\{\tau_{q,k}\}$ and at least one signer k is not corrupted, or
  - *Type-II*: τ^* is a root produced by evaluating g on some subset of $\{\tau_{q,k}\}$, but the message $m^* \notin g^*(\cdots, M_{q,k}, \cdots)$ for the corresponding subset of messages.

We say that the scheme is unforgeable under K-bounded corruption if, for all PPT adversaries $\mathcal{A}$, we have $\Pr\{\mathcal{A} \text{ wins}\} \leq \mathsf{negl}(\lambda)$ in the above game. We simply say that the scheme is unforgeable if the adversary $\mathcal{A}$ can corrupt all $\mathfrak{K}$ signers in the game.

Context Hiding. There exist a simulator $\mathcal{S} = (\mathcal{S}^{\mathsf{Setup}}, \mathcal{S}^{\mathsf{Sig}})$ such that, for all $K = \mathsf{poly}(\lambda)$, $k \in [K]$, τ_k, M_k, and g, it holds that for any PPT adversaries $\mathcal{A}$,

$$\left| \Pr\left[\begin{array}{c} \mathcal{A}((\mathsf{pk}_k, \mathsf{sk}_k, \tau_k, \Sigma_k, M_k)_{k=1}^K, \sigma, g) \to 1 : \\ \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{pk}_k, \mathsf{sk}_k) \leftarrow \mathsf{KGen}(\mathsf{pp}) \\ \Sigma_k \leftarrow \mathsf{Sig}(\mathsf{sk}_k, \tau_k, M_k) \\ \sigma \leftarrow \mathsf{Eval}(g, (\mathsf{pk}_k, \tau_k, M_k, \Sigma_k)_{k=1}^K) \end{array}\right] - \right.$$
$$\left. \Pr\left[\begin{array}{c} \mathcal{A}((\mathsf{pk}_k, \mathsf{sk}_k, \tau_k, \Sigma_k, M_k)_{k=1}^K, \sigma, g) \to 1 : \\ (\mathsf{pp}, \mathsf{td}) \leftarrow \mathcal{S}^{\mathsf{Setup}}(1^\lambda) \\ (\mathsf{pk}_k, \mathsf{sk}_k) \leftarrow \mathsf{KGen}(\mathsf{pp}) \\ \Sigma_k \leftarrow \mathsf{Sig}(\mathsf{sk}_k, \tau_k, M_k); \\ \sigma \leftarrow \mathcal{S}^{\mathsf{Sig}}(\mathsf{td}, g, (\mathsf{pk}_k, \tau_k)_{k=1}^K, g(M_1, \ldots, M_K)) \end{array}\right] \right| = \mathsf{negl}(\lambda)\,.$$

The scheme is weakly context hiding if the above holds.

Succinctness. There exist a polynomial $s(\cdot)$ such that for every $\lambda \in \mathbb{N}$, $K = \mathsf{poly}(\lambda)$, $k \in [K]$, $g \in \mathcal{G}$, $\tau_k \in \{0,1\}^*$, $M_k \in \mathcal{M}^*$, it holds with probability 1 over $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$; $(\mathsf{pk}_k, \mathsf{sk}_k) \leftarrow \mathsf{KGen}(\mathsf{pp})$; $\Sigma_k \leftarrow \mathsf{Sig}(\mathsf{sk}_k, \tau_k, M_k)$; that the resulting signature $\sigma \leftarrow \mathsf{Eval}(g, (\mathsf{pk}_k, \tau_k, M_k, \Sigma_k)_{k=1}^K)$ on $m = g(M_1, \ldots, M_k)$ has size $|\sigma| \leq s(\lambda, |m|)$. In particular, the signature size is independent of the sizes $|M_k|$ of the inputs to the function, and of the size $|g|$ of a description of the function g.

3.3 Construction from ZK-SNARK

It is well known that digital signatures, denoted by $\mathcal{DS}.(\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$, can be constructed from one-way functions [Lam79, Rom90]. For each $n \in [N]$, let $\Pi_n.(\mathsf{Gen}, \mathsf{Prove}, \mathsf{Vf})$ be a ZK-SNARK for the following recursively defined NP language L_n:

$$L_n = \begin{cases} \{(\phi, m, g, \mathsf{pk}, \tau) : m = g(M_1, \ldots, M_K) \ \wedge \ \exists (M_k, \Sigma_k)_{k=1}^K \ s.t. \\ \forall\ m_{k,j} \in M_k\ \forall\ \sigma_{k,j} = (n, \sigma'_{k,j}) \in \Sigma_k, \mathcal{DS}.\mathsf{Vf}(\mathsf{pk}_k, (\tau_k, m_{k,j}), \sigma'_{k,j}) = 1\} & n = 1 \\ \{(\mathsf{crs}_{n-1}, m, g \circ \bar{h}, \mathsf{pk}, \tau) : m = g(M_1, \ldots, M_K) \ \wedge \ \exists (M_k, \Sigma_k)_{k=1}^K \ s.t. \ \forall\ m_{k,j} \in M_k \\ \forall\ \sigma_{k,j} = (n, \sigma'_{k,j}) \in \Sigma_k, \Pi_{n-1}.\mathsf{Vf}(\mathsf{crs}_{n-1}, (\mathsf{crs}_{n-2}, m_{k,j}, h_{k,j}, \mathsf{pk}_{k,j}, \tau_{k,j}), \sigma'_{k,j}) = 1\} & n > 1 \end{cases}$$

where $h_{k,j}$, $\mathsf{pk}_{k,j}$ (resp. pk_k) and $\tau_{k,j}$ (resp. τ_k) are layer-2 (the layer just below the root) nodes of the trees $g \circ \bar{h}$, pk and τ respectively, corresponding to the messages $m_{k,j}$. Assuming the existence of one-way functions and adaptive zero-knowledge succinct non-interactive argument of knowledge (ZK-SNARK), we construct a multi-key homomorphic signature scheme $\mathcal{HS}$ as follows.

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
  - Compute $\mathsf{crs}_n \leftarrow \Pi_n.\mathsf{Gen}(1^\lambda)$ for $n \in [N]$.
  - Define $\mathsf{crs}_0 := \phi$.
  - Output $\mathsf{pp} = (1^\lambda, \mathsf{crs}_0, \ldots, \mathsf{crs}_N)$.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(\mathsf{pp})$
  - Compute $(\mathsf{pk}_{\mathcal{DS}}, \mathsf{sk}_{\mathcal{DS}}) \leftarrow \mathcal{DS}.\mathsf{KGen}(1^\lambda)$.
  - Output $\mathsf{pk} = \mathsf{pk}_{\mathcal{DS}}$, $\mathsf{sk} = \mathsf{sk}_{\mathcal{DS}}$.
- $\Sigma \leftarrow \mathsf{Sig}(\mathsf{sk}, \tau, M)$
  - Set $n = 0$.
  - Compute $\sigma'_j \leftarrow \mathcal{DS}.\mathsf{Sig}(\mathsf{sk}_{\mathcal{DS}}, (\tau, m_j))$ for each $m_j \in M$.
  - Set $\sigma_j := (n, \sigma'_j)$.
  - Output $\Sigma := \{\sigma_j\}_j$.
- $b \leftarrow \mathsf{Vf}(\mathsf{pk}, \tau, \sigma, m, g)$
  - If $n > N$, output 0.

  - If $n = 0$, output $\mathcal{DS}.\mathsf{Vf}(\mathsf{pk}, (\tau, m), \sigma')$.
  - Otherwise output $\Pi_n.\mathsf{Vf}(\mathsf{crs}_n, (\mathsf{crs}_{n-1}, m, g, \mathsf{pk}, \tau), \sigma')$.
- $\sigma \leftarrow \mathsf{Eval}(g, (\mathsf{pk}_k, \tau_k, M_k, \Sigma_k)_{k=1}^K)$
  - For each k and $\sigma_{k,j} \in \Sigma_k$, parse $\sigma_{k,j} = (n_{k,j}, \sigma'_{k,j})$.
  - Let $n = \max_{k,j}(n_{k,j})$.
  - Run $\sigma_{k,j} \leftarrow \mathsf{Eval}(\mathsf{id}, (\mathsf{pk}_k, \tau_k, m_{k,j}, \sigma_{k,j}))$ until $n_{k,j}$ reaches n.
  - Compute $m \leftarrow g(M_1, \ldots, M_K)$, $\mathsf{pk} \leftarrow g(\mathsf{pk}_1, \ldots, \mathsf{pk}_K)$, and $\tau \leftarrow g(\tau_1, \ldots, \tau_K)$.
  - Compute $\sigma' \leftarrow \Pi_{n+1}.\mathsf{Prove}(\mathsf{crs}_{n+1}, (\mathsf{crs}_n, m, g \circ \bar{h}, \mathsf{pk}, \tau), (M_k, \Sigma_k)_{k=1}^K)$.
  - Output $\sigma = (n+1, \sigma')$.

The correctness of the above construction follows straightforwardly from the correctness of $\mathcal{DS}$ and Π, and the context-hiding property follows from the zero-knowledge property of Π.

Next, we argue that the above construction is unforgeable against insider corruption. The intuition is that, if the adversary outputs a signature (a proof) of a message outside the range of g restricted by the inputs of the honest signers, then either the proof is valid for a statement outside L, which breaks the soundness of Π, or it breaks the unforgeability of $\mathcal{DS}$. Specifically, the extractor of one of the Π_n can extract signatures under some public key pk_k of some message m not signed by the honest signer k.

Theorem 1. *Suppose $\mathcal{DS}$ is EUF-CMA-secure and Π_n is sound for all $n \in [N]$, then $\mathcal{HS}$ is unforgeable.*

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

Consider a PPT simulator $\mathcal{S}$ who plays the role of the challenger. At the start of the game, $\mathcal{S}$ makes a random guess of whether $\mathcal{A}_{HS}$ will output a type-I or type-II forgery.

Case 1: Type-I Forgery. $\mathcal{S}$ acts as an adversary $\mathcal{A}_{\mathcal{DS}}$ in the unforgeability game of $\mathcal{DS}$, obtains from its challenger the public key $\mathsf{pk}_{\mathcal{DS},\mathfrak{K}}$, and generates other $\mathcal{DS}$ public keys honestly by $\{(\mathsf{pk}_{\mathcal{DS},k}, \mathsf{sk}_{\mathcal{DS},k}) \leftarrow \mathcal{DS}.\mathsf{KGen}(1^\lambda)\}_{k=1}^{\mathfrak{K}-1}$. It generates for each n $(\mathsf{crs}_n, \mathsf{xk}_n) \leftarrow \Pi_n.\mathsf{E}^1(1^\lambda)$, a simulated crs_n for Π_n, together with an extraction key xk_n, and forwards the public parameters $\mathsf{pp} = (1^\lambda, \mathsf{crs}_0, \ldots, \mathsf{crs}_N)$ and the public keys $\{\mathsf{pk}_k\}_{k=1}^{\mathfrak{K}} = \{\mathsf{pk}_{\mathcal{DS},k}\}_{k=1}^{\mathfrak{K}}$ (in random order) to $\mathcal{A}_{HS}$, where $\mathsf{crs}_0 := \phi$.

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

- Signing queries q with signer k, tag $\tau_{q,k} \in \{0,1\}^*$ and a set of messages $M_{q,k} \in \mathcal{M}^*$: If $k \neq \mathfrak{K}$, $\mathcal{S}$ signs the messages honestly by $\sigma_{q,k,j} \leftarrow \mathcal{DS}.\mathsf{Sig}(\mathsf{sk}_{\mathcal{DS},k}, (\tau_{q,k}, m_{q,k,j}))$ for each $m_{q,k,j} \in M_{q,k}$. Else, if $k = \mathfrak{K}$, $\mathcal{S}$ forwards all $(\tau_{q,\mathfrak{K}}, m_{q,\mathfrak{K},j})$ to its signing oracle, and receives $\sigma_{q,\mathfrak{K},j}$. In either case, it outputs $\{\sigma_{q,k,j}\}$.
- Corruption queries with index k: If $\mathcal{S}$ guesses type-I forgery correctly, by assumption, at least one signer is not corrupted and $\mathcal{A}$ corrupts signer $1, \ldots, K$ for some $K < \mathfrak{K}$, which means that signer $\mathfrak{K}$ is not corrupted. Thus, $\mathcal{S}$ returns $\mathsf{sk}_{\mathcal{DS},k}$. Otherwise, if $\mathcal{S}$ guesses wrongly and $\mathcal{A}$ corrupts signer $\mathfrak{K}$, then $\mathcal{S}$ aborts. This happens with probability at most $\frac{1}{2}$.

After querying the oracles, $\mathcal{A}_{HS}$ will output an alleged forgery of $\mathcal{HS}$, a public key pk^*, a tag $\tau^* \in (\{0,1\}^*)^{\mathfrak{K}}$, a function $g^* \in \mathcal{G}$, some data $m^* \in \mathcal{M}$, and a signature $\sigma^* = (n^*, \sigma')$ such that τ^* is not a root produced by evaluating g on any subset of $\{\tau_{q,k}\}$ and signer $\mathfrak{K}$ is not corrupted. $\mathcal{S}$ runs the extractors $\Pi_n.\mathsf{E}^2$ recursively from $n = n^*$ to $n = 1$, so that it recovers a set of key-message-signature tuples $\{(\mathsf{pk}^*_{q,k}, (\tau^*_{q,k}, m^*_{q,k}), \sigma^*_{q,k})\}$, all of which passes the verification of $\mathcal{DS}$. Since pk^* is produced by evaluating g on some subset of $\{\mathsf{pk}_k\}$, and τ^* is not a root produced by evaluating g on any subset of $\{\tau_{q,k}\}$, there is at least $\frac{1}{\mathfrak{K}}$ probability that τ^* contains some $\tau^*_{q,\mathfrak{K}} \notin \{\tau_{q,k}\}$ corresponding to $\mathsf{pk}_{\mathfrak{K}}$ and some $m^*_{q,\mathfrak{K}}$. Suppose that is the case, then $((\tau^*_{q,\mathfrak{K}}, m^*_{q,\mathfrak{K}}), \sigma^*_{q,\mathfrak{K}})$ is a valid forgery to $\mathcal{DS}$.

Case 2: Type-II Forgery. $\mathcal{S}$ makes a guess $\hat{n}$ of whether $\mathcal{A}$ will produce a forgery on $\mathcal{DS}$ ($\hat{n} = 0$) or a false proof in Π_n for some $\hat{n}$ ($\hat{n} \in [N]$). For $\hat{n} = 0$, $\mathcal{S}$ acts as an adversary $\mathcal{A}_{\mathcal{DS}}$ in the unforgeability game of $\mathcal{DS}$, from which it receives a public key $\mathsf{pk}_{\mathcal{DS},\mathfrak{K}}$. Otherwise, it acts as an adversary in the soundness game of $\Pi_{\hat{n}}$, from which it receives the common reference string $\mathsf{crs}_{\hat{n}}$. It generates all other public keys of $\mathcal{DS}$ honestly by $(\mathsf{pk}_{\mathcal{DS},k}, \mathsf{sk}_{\mathcal{DS},k}) \leftarrow \mathcal{DS}.\mathsf{KGen}(1^\lambda)$. For $n > \hat{n}$, it simulates the common reference string by $(\mathsf{crs}_n, \mathsf{xk}_n) \leftarrow \Pi_n.\mathsf{E}^1(1^\lambda)$. For $n < \hat{n}$, it generates the common reference string honestly by $\mathsf{crs}_n \leftarrow \Pi_n.\mathsf{Gen}(1^\lambda)$. It then forwards the public parameters $\mathsf{pp} = (1^\lambda, \mathsf{crs}_0, \ldots, \mathsf{crs}_N)$ and the public keys $\{\mathsf{pk}_k\}_{k=1}^{\mathfrak{K}} = \{\mathsf{pk}_{\mathcal{DS},k}\}_{k=1}^{\mathfrak{K}}$ (in random order) to $\mathcal{A}_{HS}$.

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

- Signing queries: If $\mathcal{S}$ guessed $\hat{n} = 0$, it answers exactly as in the type-I forgery case. Otherwise, it signs all messages (including those for signer $\mathfrak{K}$) honestly.
- Corruption queries with index k: If the guess $\hat{n} = 0$ of $\mathcal{S}$ is correct, which happens with probability at least $\frac{1}{N+1}$, then $\mathcal{A}$ will never corrupt all signers. With probability at least $\frac{1}{\mathfrak{K}}$, signer $\mathfrak{K}$ is never corrupted. Otherwise, assume the guess $\hat{n} > 0$ is correct. In both cases, $\mathcal{S}$ is always able to answer with $\mathsf{sk}_{\mathcal{DS},k}$.

Suppose $\mathcal{A}_{HS}$ corrupts signer $1, \ldots, K$ where $K \leq \mathfrak{K}$. After querying the oracles, $\mathcal{A}_{HS}$ will output an alleged forgery of $\mathcal{HS}$, a public key pk^* a tag τ^*, a function $g^* \in \mathcal{G}$, a message $m^* \in \mathcal{M}$, and a signature σ^* such that τ^* is produced by evaluating g^* on some subset of $\{\tau_{q,k}\}$ but $m^* \notin g^*(\ldots, M_{q,k}, \ldots)$ for the corresponding subset of messages.

As discussed above, if the guess $\hat{n} = 0$ of $\mathcal{S}$ is correct, which happens with probability at least $\frac{1}{N+1}$, then there is at least one uncorrupted signer. Using the same recursive extraction procedure as in the type-I forgery case, $\mathcal{S}$ recovers a set of key-message-signature tuples $\{(\mathsf{pk}^*_{q,k}, (\tau^*_{q,k}, m^*_{q,k}), \sigma^*_{q,k})\}$, all of which passes the verification of $\mathcal{DS}$. However, since $m^* \notin g^*(\ldots, M_{q,k}, \ldots)$, there must exists at least one tuple $(\tau^*_{q,k}, m^*_{q,k})$ which is not queried by $\mathcal{A}$. With probability at least $\frac{1}{\mathfrak{K}}$, such tuple corresponds to signer $\mathfrak{K}$. In such case, $((\tau^*_{q,\mathfrak{K}}, m^*_{q,\mathfrak{K}}), \sigma^*_{q,\mathfrak{K}})$ is a valid forgery to $\mathcal{DS}$.

Theorem 2. *Suppose Π_n is adaptive zero knowledge for all $n \in [N]$, then $\mathcal{HS}$ is weakly context hiding.*

Proof. Since Π_n is adaptive zero-knowledge, 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 $\mathcal{S}^{\mathsf{Prove}}_{\Pi_n}$. Since the proofs simulated from $\mathcal{S}_{\Pi_n}$ are indistinguishable from the real proofs, we also have that the simulated signatures from $\mathcal{S}_{\mathcal{HS}}$ are indistinguishable from the real signatures.

Theorem 3. *Suppose Π_n is succinct for all $n \in [N]$, then $\mathcal{HS}$ is succinct.*

Proof. The size of a signature produced by $\mathsf{Eval}(g, (\mathsf{pk}_k, \tau_k, M_k, \Sigma_k)_{k=1}^K)$ is the sum of proof length of Π_n for some n and the length of the binary representation of N, which is logarithmic in the security parameter. The succinctness of $\mathcal{HS}$ follows directly from the succinctness property of Π.

3.4 Construction from Lattices

If we settle for unforgeability under 0-bounded corruption, then a generalization of the (leveled) fully homomorphic signatures by Gorbunov *et al.* [GVW15] (GVW) gives an M-HS scheme. The generalization is straightforward, thus we only explain the core idea. [1]

The verification equation of the GVW scheme is of the form $\mathbf{V} = \mathbf{AU} + x\mathbf{G}$, where $x \in \{0, 1\}$ is the message signed, $\mathbf{A}$ and $\mathbf{V}$ are random matrices given in the public key, $\mathbf{U}$ is a random matrix of small norm acting as the signature of x, and $\mathbf{G}$ is a fixed, nicely-structured "gadget" matrix such that there exists an efficient algorithm (with abused notion as a matrix) $\mathbf{G}^{-1}$ which samples matrices of small norm from the kernel of $\mathbf{G}$. The secret key of the scheme is a "trapdoor" for the matrix $\mathbf{A}$ which allows efficient sampling of matrices of small norm from the kernel of $\mathbf{A}$.

[1] A concurrent work by Fiore *et al.* [FMNP16] has formalized the idea.

Suppose $\mathbf{U}_1$ and $\mathbf{U}_2$ are the signatures given by two different signers on the messages x_1 and x_2 respectively which satisfy the equations $\mathbf{V}_1 = \mathbf{A}_1\mathbf{U}_1 + x_1\mathbf{G}$ and $\mathbf{V}_2 = \mathbf{A}_2\mathbf{U}_2 + x_2\mathbf{G}$. Given the signatures, a public evaluator can compute the signatures $\mathbf{U}^+ = (\mathbf{U}_1, \mathbf{U}_2)^T$ and $\mathbf{U}^\times = (\mathbf{U}_1\mathbf{G}^{-1}\mathbf{V}_2, x_1\mathbf{U}_2)^T$ for the addition and multiplication function respectively. We can verify that these signatures satisfy $\mathbf{V}_1 + \mathbf{V}_2 = (\mathbf{A}_1, \mathbf{A}_2)\mathbf{U}^+ + (x_1 + x_2)\mathbf{G}$ and $\mathbf{V}_1\mathbf{G}^{-1}(\mathbf{V}_2) = (\mathbf{A}_1, \mathbf{A}_2)\mathbf{U}^\times + (x_1x_2)\mathbf{G}$. The treatments for adaptive security, multi-data support, and context-hiding property follow directly from the work of Gorbunov *et al.* [GVW15].

Unfortunately, such simple generalization only achieves unforgeability under 0-bounded corruption, which means the adversary is not allowed to corrupt any of the signers. This is because, if the adversary has knowledge of the trapdoor of $\mathbf{A}_k$ for some signer k, then it can sample matrices of small norm from the kernel of $(\mathbf{A}_1, \ldots, \mathbf{A}_K)$ using standard trapdoor delegation technique. More disappointingly, even the original (single-key) scheme of GVW does not satisfy unforgeability under 1-bounded corruption. We believe that crossing the 0- to 1-bounded corruption gap based on standard assumption requires substantially different techniques.

3.5 Functional Signatures from M-HS

We begin to show the power of M-HS by constructing functional signatures [BGI14] using a 2-key HS. As mentioned in the introduction, FS allows an authority with a master secret key to derive function-specific signing keys. Given a signing key for a function f, one can only sign messages in the range of f.

Definition. We recall the formal definition of functional signatures [BGI14].

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

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

Correctness. We require that 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 $\mathcal{FS}.\mathsf{Vf}(\mathsf{mpk}, m^*, \sigma) = 1$.

Unforgeability. The scheme is unforgeable if the advantage of any PPT adversary $\mathcal{A}$ in the following game is negligible:

- The challenger generates $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathcal{FS}.\mathsf{Setup}(1^\lambda)$, and gives mpk to $\mathcal{A}$.
- $\mathcal{A}$ is allowed to query a key generation oracle $\mathcal{O}_{\mathsf{key}}$ and a signing oracle $\mathcal{O}_{\mathsf{sign}}$. 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 during the unforgeability game. The oracles are defined as follows:
  - $\mathcal{O}_{\mathsf{key}}(f, i)$
    - If there exists an entry for the key (f, i) in the dictionary, then 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 $(f, i) \to \mathsf{sk}_f^i$ 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 an entry $(f, i) \rightarrow \mathsf{sk}_f^i$ to 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 (f, m) pair such that (f, m) was a query to the $\mathcal{O}_{\mathsf{sign}}$ oracle and $m^* = f(m)$.

Function Privacy. The scheme is function private if the advantage of any PPT adversary $\mathcal{A}$ in the following game is negligible:

- The challenger honestly generates a key pair $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathcal{FS}.\mathsf{Setup}(1^\lambda)$ and gives both values to $\mathcal{A}$. (Note that w.l.o.g. this includes the randomness used in generation).
- $\mathcal{A}$ chooses a function f_0 and receives an (honestly generated) secret key $\mathsf{sk}_{f_0} \leftarrow \mathcal{FS}.\mathsf{KGen}(\mathsf{msk}, f_0)$.
- $\mathcal{A}$ chooses a second function f_1 for which $|f_0| = |f_1|$ (where padding can be useful if there is a known upper bound) and receives an (honestly generated) secret key $\mathsf{sk}_{f_1} \leftarrow \mathcal{FS}.\mathsf{KGen}(\mathsf{msk}, f_1)$.
- $\mathcal{A}$ chooses a pair of value m_0, m_1 for which $|m_0| = |m_1|$ and $f_0(m_0) = f_1(m_1)$.
- The challenger select 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 $b' = b$.

Succinctness. There exist a polynomial $s(\cdot)$ such that for every $k \in \mathbb{N}$, $f \in \mathcal{F}$, $m \in \mathcal{D}_f$, it holds with probability 1 over $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathcal{FS}.\mathsf{Setup}(1^\lambda)$; $\mathsf{sk}_f \leftarrow \mathcal{FS}.\mathsf{KGen}(\mathsf{msk}, f)$; $(f(m), \sigma) \leftarrow \mathcal{FS}.\mathsf{Sig}(f, \mathsf{sk}_f, m)$ that the resulting signature on $f(m)$ has size $|\sigma| \leq s(k, |f(m)|)$. In particular, the signature size is independent of the size $|m|$ of the input to the function, and of the size $|f|$ of a description of the function f.

Construction. Let $\mathcal{HS}.(\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf}, \mathsf{Eval})$ be a 1-hop 2-HS scheme for a function family $\mathcal{G} = \{U : \{0,1\}^s \times \{0,1\}^n \rightarrow \{0,1\}^k\}$, where U is the universal circuit taking as input a circuit f with description size s and its n-bit input m, and computes $U(f, m) = f(m)$ of length k. Let $\mathcal{M} = \{0, 1\}$. We construct a functional signature scheme $\mathcal{FS}.(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ for the function family $\mathcal{F} = \{f : \{0,1\}^n \rightarrow \{0,1\}^k \ s.t. \ |f| = s\}$ as follows:

- $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathcal{FS}.\mathsf{Setup}(1^\lambda)$
    - Compute $(\mathsf{pk}_k, \mathsf{sk}_k) \leftarrow \mathcal{HS}.\mathsf{KGen}(1^\lambda)$ for $k = 1, 2$.
    - Output $\mathsf{mpk} = (\mathsf{pk}_1, \mathsf{pk}_2)$ and $\mathsf{msk} = (\mathsf{sk}_1, \mathsf{sk}_2)$.
- $\mathsf{sk}_f \leftarrow \mathcal{FS}.\mathsf{KGen}(\mathsf{msk}, f)$
    - Compute $\sigma_f \leftarrow \mathcal{HS}.\mathsf{Sig}(\mathsf{sk}_1, \mathsf{pk}_2, f)$.
    - Output $\mathsf{sk}_f = (\mathsf{sk}_2, \sigma_f)$.
- $(f(m), \sigma) \leftarrow \mathcal{FS}.\mathsf{Sig}(f, \mathsf{sk}_f, m)$
    - Parse $\mathsf{sk}_f = (\mathsf{sk}_2, \sigma_f)$.
    - Compute $\sigma_m \leftarrow \mathcal{HS}.\mathsf{Sig}(\mathsf{sk}_2, 0, m)$.
    - Compute $\sigma \leftarrow \mathcal{HS}.\mathsf{Eval}(U, ((\mathsf{pk}_1, \mathsf{pk}_2, f, \sigma_f), (\mathsf{pk}_2, 0, m, \sigma_m)))$, where U is the universal circuit.
    - Output $(U(f, m), \sigma)$.
- $b \leftarrow \mathcal{FS}.\mathsf{Vf}(\mathsf{mpk}, m, \sigma)$
    - Output $b \leftarrow \mathcal{HS}.\mathsf{Vf}((\mathsf{pk}_1, \mathsf{pk}_2), (\mathsf{pk}_2, 0), \sigma, m, U)$.

Theorem 4. *Suppose $\mathcal{HS}$ is unforgeable under 1-bounded corruption, then $\mathcal{FS}$ is unforgeable.*

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

$\mathcal{A}_{\mathcal{HS}}$ received $(\mathsf{pk}_1, \mathsf{pk}_2)$ from the challenger of the unforgeability game of $\mathcal{HS}$. $\mathcal{A}_{\mathcal{HS}}$ also queries the corruption oracle of $\mathcal{HS}$ game to get sk_2. It forwards public key $\mathsf{mpk} = (\mathsf{pk}_1, \mathsf{pk}_2)$ to $\mathcal{A}_{\mathcal{FS}}$.

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

– $\mathcal{O}_{\mathsf{key}}(f, i)$
  - If there exists an entry for the key (f, i) in the dictionary, than output the corresponding value, sk_f^i.
  - Otherwise, query the signing oracle of $\mathcal{HS}$ game to get $\sigma_f^i \leftarrow \mathcal{HS}.\mathsf{Sig}(\mathsf{sk}_1, \mathsf{pk}_2, f)$. Then add an entry $(f, i) \rightarrow \mathsf{sk}_f^i = (\mathsf{sk}_2, \sigma_f^i)$ 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, retrieve $\mathsf{sk}_f^i = (\mathsf{sk}_2, \sigma_f^i)$.
  - Otherwise, query the signing oracle of $\mathcal{HS}$ game to get $\sigma_f^i \leftarrow \mathcal{HS}.\mathsf{Sig}(\mathsf{sk}_1, \mathsf{pk}_2, f)$. Then add an entry $(f, i) \rightarrow \mathsf{sk}_f^i = (\mathsf{sk}_2, \sigma_f^i)$.
  - Compute $\sigma_m \leftarrow \mathcal{HS}.\mathsf{Sig}(\mathsf{sk}_2, 0, m)$ and $\sigma \leftarrow \mathcal{HS}.\mathsf{Eval}(U, ((\mathsf{pk}_1, \mathsf{pk}_2, f, \sigma_f^i), (\mathsf{pk}_2, 0, m, \sigma_m)))$ in either case, where U is the universal circuit. Output $(U(f, m), \sigma)$.

After querying the oracles, $\mathcal{A}_{\mathcal{FS}}$ responds with forgery (m^*, σ^*). $\mathcal{A}_{\mathcal{HS}}$ then answer $((\mathsf{pk}_2, 0), \sigma^*, m^*, U)$ to its unforgeability game. Notice that $\mathcal{A}_{\mathcal{HS}}$ only queries the corruption oracle once. Moreover, the answer of $\mathcal{A}_{\mathcal{HS}}$ is a valid forgery since, by the definition of the unforgeability game of functional signatures, m^* is not in the range of any f queried to the $\mathcal{O}_{\mathsf{key}}$ oracle, and $m^* \neq f(m)$ for any (f, m) queried to the $\mathcal{O}_{\mathsf{sign}}$ oracle.

Theorem 5. *Suppose $\mathcal{HS}$ is weakly context-hiding, then $\mathcal{FS}$ is function private.*

Proof. Let $\mathcal{A}_{\mathcal{FS}}$ be an adversary playing the function privacy with a challenger. Since $\mathcal{HS}$ is weakly context-hiding, there exists a simulator $\mathcal{S}_{\mathcal{HS}}$ which, on input $(U, ((\mathsf{pk}_1, \mathsf{pk}_2), (\mathsf{pk}_2, 0)), f(m))$, outputs a signature of $f(m)$ which is indistinguishable from that produced by $\mathcal{FS}.\mathsf{Sig}(f, \mathsf{sk}_f, m)$. We can thus replace the challenger with the simulator $\mathcal{S}_{\mathcal{HS}}$, which is indistinguishable in the view of $\mathcal{A}_{\mathcal{FS}}$ except with negligible probability. Notice that the simulated signatures contain no information about the function f and input message m except for $f(m)$. Thus, the probability that the adversary $\mathcal{A}_{\mathcal{FS}}$ guesses correctly in the simulated game is 0.

Theorem 6. *Suppose $\mathcal{HS}$ is succinct then $\mathcal{FS}$ is succinct.*

Proof. The size of a functional 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 the succinctness property of $\mathcal{HS}$.

Since the existence of secure functional signatures implies that of SNARG, we have the following corollary.

Corollary 1. *Suppose an unforgeable (under 1-bounded corruption) and context-hiding 1-hop 2-HS exists, then SNARG for* NP *exists.*

3.6 SNARG from M-HS

In the previous subsection, we show that the existence of 2-HS implies that of FS, which in turn implies the existence of SNARG. However, there are two limitations. First, the existence of FS only implies the existence of non-zero-knowledge SNARG. Second, as constructing 2-HS might be significantly more difficult than constructing (1-)HS (both with unforgeability under 1-bounded corruption), it is desirable to construct (ZK-)SNARG directly from HS.

Let $\mathcal{HS} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf}, \mathsf{Eval})$ be a 1-hop (1-)HS scheme. We construct a SNARG system $\Pi = (\mathsf{Gen}, \mathsf{Prove}, \mathsf{Vf})$ for NP language L with relation R as follows:

– $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda)$
  - Compute $\mathsf{pp} \leftarrow \mathcal{HS}.\mathsf{Setup}(1^\lambda)$.
  - Compute $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{HS}.\mathsf{KGen}(\mathsf{pp})$.
  - Output $\mathsf{crs} = (\mathsf{pk}, \mathsf{sk})$.
– $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)$
  - Compute $(\sigma_x, \sigma_w) \leftarrow \mathcal{HS}.\mathsf{Sig}(\mathsf{sk}, 0, (x, w))$.

- Compute $\sigma \leftarrow \mathcal{HS}.\mathsf{Eval}(g, ((\mathsf{pk}, 0, x, \sigma_x), (\mathsf{pk}, 0, w, \sigma_w)))$, where g is the following function:

$$g(x, w) = \begin{cases} x & \text{if } R(x, w) = 1 \\ \perp & \text{otherwise} \end{cases}$$

- Output $\pi = \sigma$.

– $b \leftarrow \mathsf{Vf}(\mathsf{crs}, x, \pi)$
- Output $b \leftarrow \mathcal{HS}.\mathsf{Vf}(\mathsf{pk}, 0, \pi, x, g)$.

Theorem 7. *Suppose $\mathcal{HS}$ is correct, then Π is complete.*

Theorem 8. *Suppose $\mathcal{HS}$ is unforgeable under 1-bounded corruption, then Π is sound.*

Proof. Suppose there exists an adversary $\mathcal{A}_\Pi$ that breaks the soundness of Π with non-negligible probability. We show how to 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 unforgeability game of $\mathcal{FS}$.

$\mathcal{A}_{\mathcal{HS}}$ received pk from the challenger of the unforgeability game of $\mathcal{HS}$. $\mathcal{A}_{\mathcal{HS}}$ also queries the corruption oracle of the unforgeability game to get sk. It forwards the common reference string $\mathsf{crs} = (\mathsf{pk}, \mathsf{sk})$ 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 answer $(\mathsf{pk}, 0, \pi^*, x^*, g)$ to its unforgeability game. Since $x^* \notin L$, we have $x^* \neq g(x, w)$ for all $(x, w) \in \mathcal{M}^2$.

Theorem 9. *Suppose $\mathcal{HS}$ 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 $(R, \mathsf{pk}, 0, x)$ a signature on x which is statistically close to the real signatures. We can thus construct $\mathcal{S}_\Pi^{\mathsf{crs}}$ using $\mathcal{S}_{\mathcal{HS}}^{\mathsf{Setup}}$ and $\mathcal{S}_\Pi^{\mathsf{Prove}}$ using $\mathcal{S}_{\mathcal{HS}}^{\mathsf{Sig}}$, and conclude that Π is zero-knowledge.

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

Proof. The size of a proof produced by $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)$ is the signature length of $\mathcal{HS}$. The succinctness of Π follows directly from the succinctness of $\mathcal{HS}$.

4 Extensions and Special Cases of Multi-Key HS

4.1 Multi-Key Hierarchical Homomorphic Signatures (M-HiHS)

Imagine the following scenario: Alice fills in the first two entries of a form, and passes the form to Bob. The latter fills the next two entries, and passes to Charlie, and so on. At the end, an evaluator collects a number of forms filled by multiple groups of people and processes them in batch. To support such functionality using some variant of M-HS, we require additionally that the scheme is delegatable. Roughly, a delegator is able to derive a signing key which is able to sign any set of messages that contains a subset specified by the delegator.

Formally, we define an extension of M-HS, namely, multi-key hierarchical homomorphic signatures (M-HiHS). M-HiHS allows a signer to specify a set of messages M, and delegate the signing power of all supersets of M to another signer. More concretely, let pk and sk be the public and secret key of the delegator. To delegate the signing power over a set Y to another signer with public key pk', the delegator derives a secret key $\mathsf{sk}_{(\mathsf{pk},\mathsf{pk}'),M}$ A delegatee can use this secret key to sign any set $M' \supseteq M$ to create a complete signature, so that any public evaluator can derive a signature of $g(M')$ signed under the delegation chain $(\mathsf{pk}, \mathsf{pk}')$. This generalizes proxy signatures in the literature where M serves as a "smart warrant" with respect to function g.

More generally, a delegatee can further delegate its signing power to form a longer delegation chain. Lastly, similar to M-HS, the public evaluator can combine signatures signed under different chains.

Definitions

Syntax. An M-HIHS scheme consists of the PPT algorithms $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Del}, \mathsf{Sig}, \mathsf{Vf}, \mathsf{Eval})$ defined as follows:

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ is the same as that of M-HS.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(\mathsf{pp})$ is the same as that of M-HS.
- $\mathsf{sk}_{(\bar{\mathsf{pk}}, \mathsf{pk}'), M'} \leftarrow \mathsf{Del}(\mathsf{sk}, \mathsf{sk}_{\bar{\mathsf{pk}}, M}, \mathsf{pk}', \tau, M')$ is a new delegation algorithm which inputs the secret key sk of the delegator, an *optional* delegated key $\mathsf{sk}_{\bar{\mathsf{pk}}, M}$ for the delegation chain $\bar{\mathsf{pk}} = (\ldots, \mathsf{pk})$ and messages M, the public key pk' of the delegatee, a tag $\tau \in \{0,1\}^*$, and a set of messages $M' \in \mathcal{M}^*$ such that $M' \supseteq M$. It outputs a new delegated key $\mathsf{sk}_{(\bar{\mathsf{pk}}, \mathsf{pk}'), M'}$.
- $\Sigma \leftarrow \mathsf{Sig}(\mathsf{sk}, \mathsf{sk}_{\bar{\mathsf{pk}}, M}, \tau, M')$ is the same as that of M-HS, except that it takes as input an *optional* delegated key $\mathsf{sk}_{\bar{\mathsf{pk}}, M}$ for the delegation chain $\bar{\mathsf{pk}} = (\ldots, \mathsf{pk})$ and messages M. It outputs the finalized signatures Σ on $M' \in \mathcal{M}^*$ such that $M' \supseteq M$.
- $b \leftarrow \mathsf{Vf}(\mathsf{pk}, \tau, \sigma, m, g)$ is the same as that of M-HS.
- $\sigma \leftarrow \mathsf{Eval}(g, (\mathsf{pk}_k, \tau_k, M_k, \Sigma_k)_{k=1}^K)$ is the same as that of M-HS.

If the delegation chain length is limited to 1, all delegation chains $\bar{\mathsf{pk}}$ becomes a singleton containing only the public key pk of the root delegator. The M-HiHS scheme thus becomes an M-HS scheme. The correctness, unforgeability, and context-hiding property are defined similarly to those of M-HS. A major difference is that, in the unforgeability game, the adversary is in addition allowed to query a delegation oracle, and the condition of valid forgery is adjusted accordingly to avoid trivial attacks.

Correctness. For any $\mathsf{pp} \in \mathsf{Setup}(1^\lambda)$, any $K, D_k = \mathsf{poly}(\lambda)$, $k \in [K]$, and $d_k \in [D_k]$, any $(\mathsf{pk}_{k,d_k}, \mathsf{sk}_{k,d_k}) \in \mathsf{KGen}(\mathsf{pp})$, any $M_{k,d_k} \in \mathcal{M}^*$, any $\tau_k \in \{0,1\}^*$, and any $g, h_{k,j} \in \mathcal{G}$ with appropriate dimensions, the following is true:

- (Delegation and Signing Correctness.) For any K delegation chains of lengths D_k, *i.e.*,

$$\mathsf{sk}_{\bar{\mathsf{pk}}_{k,d_k+1}, M_{k,d_k}} \leftarrow \mathsf{Del}(\mathsf{sk}_{k,d_k}, \mathsf{sk}_{\bar{\mathsf{pk}}_{k,d_k}, M_{k,d_k-1}}, \mathsf{pk}_{k,d_k+1}, M_{k,d_k}),$$

  and any K finalizing signatures, *i.e.*,

$$\Sigma_k \leftarrow \mathsf{Sig}(\mathsf{sk}_{k,D_k}, \mathsf{sk}_{\bar{\mathsf{pk}}_{k,D_k}, M_{k,D_k-1}}, \tau_{k,D_k}, M_{k,D_k}),$$

  the verification of any individual signature passes, *i.e.*,

$$\mathsf{Vf}(p_j(\bar{\mathsf{pk}}_{k,D_k}), p_j(\bar{\tau}_{k,D_k}), \sigma_{k,d_k}, m_{k,d_k}, p_j) = 1$$

  for each $\sigma_{k,j} \in \Sigma_k$ and $m_{k,j} \in M_k$, where p_j is the j-th projection function;
- (Evaluation Correctness.) For any Σ_{k,d_k}, if $\mathsf{Vf}(\mathsf{pk}_{k,d_k}, \tau_{k,d_k}, \sigma_{k,d_k,j}, m_{k,d_k,j}, h_{k,d_k,j}) = 1$ for all $\sigma_{k,d_k,j} \in \Sigma_{k,d_k}$ and $m_{k,d_k,j} \in M_{k,d_k}$, $\mathsf{Hop}(h_{k,j,d_k}) \leq N-1$, and $\sigma \leftarrow \mathsf{Eval}(g, (\mathsf{pk}_{k,d_k}, \tau_{k,d_k}, M_{k,d_k}, \Sigma_{k,d_k})_{k=1,d_k=1}^{K,D_k})$, then $\mathsf{Vf}(\mathsf{pk}, \tau, \sigma, g(M_{1,1}, \ldots, M_{K,D_k}), g \circ \bar{h}) = 1$, where $\mathsf{pk} = g(\ldots, \mathsf{pk}_{k,d_k}, \ldots)$ and $\tau = g(\ldots, \tau_{k,d_k}, \ldots)$.

Unforgeability. Consider the following security game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

- Let $\mathfrak{K} = \mathsf{poly}(\lambda)$ be the number of signers in the system.
- The challenger $\mathcal{C}$ initializes a dictionary storing all existing delegation chains with the delegated messaged in the system.
- The challenger $\mathcal{C}$ runs $\mathsf{pp} \in \mathsf{Setup}(1^\lambda)$ and $\{(\mathsf{pk}_k, \mathsf{sk}_k) \leftarrow \mathsf{KGen}(\mathsf{pp})\}_{k=1}^{\mathfrak{K}}$, and gives $\mathsf{pp}, \mathsf{pk}_1, \ldots, \mathsf{pk}_{\mathfrak{K}}$ to $\mathcal{A}$. It also adds each $(\mathsf{pk}_k, M = \phi)$ to the dictionary.
- The adversary $\mathcal{A}$ adaptively issues a polynomial number of delegation queries, signing queries, and corruption queries.

- Delegation queries: $\mathcal{A}$ chooses an existing delegation chain $\bar{pk}$ (we assume each individual public key forms a chain of length 1 without specified messages), a public key pk' to extend the chain with, and a set of messages M'. The challenger $\mathcal{C}$ checks if $(\bar{\mathsf{pk}}, M)$ exists on the dictionary for some $M \subseteq M'$. If that is not the case, it responds with $\perp$. Otherwise, it responds with $\mathsf{sk}_{(\bar{\mathsf{pk}},\mathsf{pk}'),M'} \leftarrow \mathsf{Del}(\mathsf{sk}, \mathsf{sk}_{\bar{\mathsf{pk}},M}, \mathsf{pk}', M')$.
- Signing queries: In each signing query, $\mathcal{A}$ chooses a delegation chain $\bar{\mathsf{pk}}$, a tag $\tau \in \{0,1\}^*$, and a set of data $M' \in \mathcal{M}^*$. The challenger $\mathcal{C}$ checks if $(\bar{\mathsf{pk}}, M)$ exists on the dictionary for some $M \subseteq M'$. If that is not the case, it responds with $\perp$. Otherwise, it responds with $\Sigma \leftarrow \mathsf{Sig}(\mathsf{sk}, \mathsf{sk}_{\bar{\mathsf{pk}},M}, \tau, M)$.
- Corruption queries: $\mathcal{A}$ chooses a signer k. The challenger $\mathcal{C}$ responds with sk_k.

– Without loss of generality, we assume that the adversary $\mathcal{A}$ corrupts signers $1, \ldots, K$.
– The adversary $\mathcal{A}$ outputs a public key pk^*, a tag $\tau^* \in \{0,1\}^*$, a function $g^* \in \mathcal{G}$, a message $m^* \in \mathcal{M}$, and a signature σ^*. It wins the game if $\mathsf{Vf}(\mathsf{pk}^*, \tau^*, \sigma^*, m^*, g^*) = 1$, pk^* is produced by evaluating g on some subset of $\{\mathsf{pk}_k\}$, and $(\mathsf{pk}^*, \tau^*, \sigma^*, m^*, g^*)$ is either a forgery such that
  - *Type-I*: τ^* is not a root produced by evaluating g on any subset of tags submitted in the delegation and signing queries, and at least one signer k is not corrupted, or
  - *Type-II*: τ^* is a root produced by evaluating g on some subset of tags submitted in the delegation and signing queries, but the message m^* is not in the range of g^* restricted to the corresponding subset of messages submitted in the queries.

We say that the scheme is unforgeable under K-bounded corruption if, for all PPT adversaries $\mathcal{A}$, we have $\Pr\{\mathcal{A} \text{ wins}\} \leq \mathsf{negl}(\lambda)$ in the above game. We simply say that the scheme is unforgeable if the adversary $\mathcal{A}$ can corrupt all $\mathfrak{K}$ signers in the game.

Context-Hiding. There exist a simulator $\mathcal{S} = (\mathcal{S}^{\mathsf{Setup}}, \mathcal{S}^{\mathsf{Sig}})$ such that, for all $K, D_k = \mathsf{poly}(\lambda)$, $k \in [K]$, $d_k \in [D_k]$, τ_{k,d_k}, M_{k,d_k}, and $g \in \mathcal{G}$, it holds that for any PPT adversaries $\mathcal{A}$,

$$
\left|\Pr\left[\begin{array}{c}
\mathcal{A}((\mathsf{pk}_{k,d_k}, \mathsf{sk}_{k,d_k}, \tau_{k,d_k}, \Sigma_{k,d_k}, M_{k,d_k})_{k=1,d_k=1}^{K,D_k}, \sigma, g) \to 1 : \\
\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\
(\mathsf{pk}_{k,d_k}, \mathsf{sk}_{k,d_k}) \leftarrow \mathsf{KGen}(\mathsf{pp}) \\
\mathsf{sk}_{\bar{\mathsf{pk}}_{k,d_k+1},M'} \leftarrow \mathsf{Del}(\mathsf{sk}_{k,d_k}, \mathsf{sk}_{\bar{\mathsf{pk}}_{k,d_k},M_{k,d_k-1}}, \mathsf{pk}_{k,d_k+1}, \tau, M_{k,d_k}) \\
\Sigma_k \leftarrow \mathsf{Sig}(\mathsf{sk}_{k,D_k}, \mathsf{sk}_{\bar{\mathsf{pk}}_{k,D_k},M_{k,D_k-1}}, \tau_k, M_{k,D_k}) \\
\sigma \leftarrow \mathsf{Eval}(g, (\bar{\mathsf{pk}}_{k,D_k}, \tau_{k,D_k}, M_{k,D_k}, \Sigma_k)_{k=1}^K)
\end{array}\right] - \right.
$$

$$
\left.\Pr\left[\begin{array}{c}
\mathcal{A}((\mathsf{pk}_{k,d_k}, \mathsf{sk}_{k,d_k}, \tau_{k,d_k}, \Sigma_{k,d_k}, M_{k,d_k})_{k=1,d_k=1}^{K,D_k}, \sigma, g) \to 1 : \\
(\mathsf{pp}, \mathsf{td}) \leftarrow \mathcal{S}^{\mathsf{Setup}}(1^\lambda) \\
(\mathsf{pk}_{k,d_k}, \mathsf{sk}_{k,d_k}) \leftarrow \mathsf{KGen}(\mathsf{pp}) \\
\mathsf{sk}_{\bar{\mathsf{pk}}_{k,d_k+1},M'} \leftarrow \mathsf{Del}(\mathsf{sk}_{k,d_k}, \mathsf{sk}_{\bar{\mathsf{pk}}_{k,d_k},M_{k,d_k-1}}, \mathsf{pk}_{k,d_k+1}, \tau, M_{k,d_k}) \\
\Sigma_k \leftarrow \mathsf{Sig}(\mathsf{sk}_{k,D_k}, \mathsf{sk}_{\bar{\mathsf{pk}}_{k,D_k},M_{k,D_k-1}}, \tau_k, M_{k,D_k}) \\
\sigma \leftarrow \mathcal{S}^{\mathsf{Sig}}(\mathsf{td}, g, (\bar{\mathsf{pk}}_{k,D_k}, \tau_{k,D_k})_{k=1}^K, g(M_{1,D_1}, \ldots, M_{K,D_K}))
\end{array}\right]\right| = \mathsf{negl}(\lambda)\,.
$$

The scheme is weakly context hiding if the above holds.

Succinctness. There exist polynomials $s_1(\cdot)$ and $s_2(\cdot)$ such that for every $\lambda \in \mathbb{N}$, $K, D_k = \mathsf{poly}(\lambda)$, $k \in [K]$, $d_k \in [D_k]$, $g \in \mathcal{G}$, $\tau_{k,d_k} \in \{0,1\}^*$, $M_{k,d_k} \in \mathcal{M}^*$, it holds with probability 1 over $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$; $(\mathsf{pk}_{k,d_k}, \mathsf{sk}_{k,d_k}) \leftarrow \mathsf{KGen}(\mathsf{pp})$; $\mathsf{sk}_{\bar{\mathsf{pk}}_{k,d_k+1},M'} \leftarrow \mathsf{Del}(\mathsf{sk}_{k,d_k}, \mathsf{sk}_{\bar{\mathsf{pk}}_{k,d_k},M_{k,d_k-1}}, \mathsf{pk}_{k,d_k+1}, \tau, M_{k,d_k})$; $\Sigma_k \leftarrow \mathsf{Sig}(\mathsf{sk}_{k,D_k}, \mathsf{sk}_{\bar{\mathsf{pk}}_{k,D_k},M_{k,D_k-1}}, \tau_k, M_{k,D_k})$; that the resulting signature $\sigma \leftarrow \mathsf{Eval}(g, (\bar{\mathsf{pk}}_{k,D_k}, \tau_{k,D_k}, M_{k,D_k}, \Sigma_k)_{k=1}^K)$ on $m = g(M_{1,1}, \ldots, M_{K,D_k})$ has size $|\sigma| \leq s_1(\lambda, |m|) + s_2(\lambda, \sum_{k=1}^K D_k)$. In particular, the signature size is independent of the sizes $|M_{k,d_k}|$ of the inputs to the function, and of the size $|g|$ of a description of the function g. We note that one could also require that the signature size to be independent of the number of delegation chains and their lengths.

Construction We next show that M-HiHS can be constructed from M-HS, digital signatures, and collision-resistant hash functions, which implies the equivalence of their existence up to the existence of the latter building blocks. The idea is to use the delegation chain as part of the tag of a signature. To extend the delegation chain $\bar{\mathsf{pk}}$ to pk' with the set of messages $M' \supseteq M$, the delegator simply signs M under a random tag using M-HS, and signs the hash value of all previous tags together with pk' by ordinary signatures. Likewise, to finalize the delegation chain $\bar{\mathsf{pk}}$ with a set of messages M and tag τ, the delegatee signs M under the tag τ using M-HS, and signs all previous tags with ordinary signatures.

Formally, let $H : \{0,1\}^* \to \{0,1\}^\lambda$ be a collision-resistant hash function, $\mathcal{HS}.(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf}, \mathsf{Eval})$ be an N-hop M-HS scheme, and $\mathcal{DS}.(\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ be a digital signature scheme. We construct an N-hop M-HiHS scheme $\mathcal{HS}'$ as follows:

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
  - Output $\mathsf{pp} \leftarrow \mathcal{HS}.\mathsf{Setup}(1^\lambda)$.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$
  - Compute $(\mathsf{pk}_{\mathcal{HS}}, \mathsf{sk}_{\mathcal{HS}}) \leftarrow \mathcal{HS}.\mathsf{KGen}(\mathsf{pp})$.
  - Compute $(\mathsf{pk}_{\mathcal{DS}}, \mathsf{sk}_{\mathcal{DS}}) \leftarrow \mathcal{DS}.\mathsf{KGen}(1^\lambda)$.
  - Output $\mathsf{pk} := (\mathsf{pk}_{\mathcal{HS}}, \mathsf{pk}_{\mathcal{DS}})$ and $\mathsf{sk} := (\mathsf{sk}_{\mathcal{HS}}, \mathsf{sk}_{\mathcal{DS}})$.
- $\mathsf{sk}_{(\bar{\mathsf{pk}}, \mathsf{pk}'), M'} \leftarrow \mathsf{Del}(\mathsf{sk}, \mathsf{sk}_{\bar{\mathsf{pk}}, M}, \mathsf{pk}', \tau, M')$
  - If $\mathsf{sk}_{\bar{\mathsf{pk}}, M} = \phi$ (fresh delegation chain)
    - Compute $\gamma \leftarrow \mathcal{DS}.\mathsf{Sig}(\mathsf{sk}_{\mathcal{DS}}, (\mathsf{pk}', H(\tau)))$.
    - Compute $\Sigma' \leftarrow \mathcal{HS}.\mathsf{Sig}(\mathsf{sk}_{\mathcal{HS}}, \tau, M')$.
    - Output (τ, γ, Σ').
  - Else, parse $\mathsf{sk}_{\bar{\mathsf{pk}}, M}$ as $(\tau_d, \gamma_d, \Sigma'_d)_{d=1}^{D}$.
  - Define $\tau_{D+1} := \tau$.
  - Compute $\gamma_{D+1} \leftarrow \mathcal{DS}.\mathsf{Sig}(\mathsf{sk}_{\mathcal{DS}}, (\mathsf{pk}', H(\tau_1, \ldots, \tau_{D+1})))$.
  - Compute $\Sigma'_{D+1} \leftarrow \mathcal{HS}.\mathsf{Sig}(\mathsf{sk}_{\mathcal{HS}}, \tau_{D+1}, M' \setminus M)$.
  - Output $(\tau_d, \gamma_d, \Sigma'_d)_{d=1}^{D+1}$.
- $\Sigma \leftarrow \mathsf{Sig}(\mathsf{sk}, \mathsf{sk}_{\bar{\mathsf{pk}}, M}, \tau, M')$
  - If $\mathsf{sk}_{\bar{\mathsf{pk}}, M} = \phi$, output $\Sigma \leftarrow \mathcal{HS}.\mathsf{Sig}(\mathsf{sk}_{\mathcal{HS}}, \tau, M')$.
  - Else, parse $\mathsf{sk}_{\bar{\mathsf{pk}}, M}$ as $(\tau_d, \gamma_d, \Sigma'_d)_{d=1}^{D}$.
  - Partition M' as $(M_1, \ldots, M_{D+1})$ where $M_{D+1} = M' \setminus M$.
  - Define $\mathsf{pk}_{D+1} := \mathsf{pk}$ and $\tau_{D+1} := \tau$.
  - Compute $\gamma_{D+1} \leftarrow \mathcal{DS}.\mathsf{Sig}(\mathsf{sk}_{\mathcal{DS}}, H(\tau_1, \ldots, \tau_{D+1}))$.
  - Compute $\Sigma'_{D+1} \leftarrow \mathcal{HS}.\mathsf{Sig}(\mathsf{sk}_{\mathcal{HS}}, \tau_{D+1}, M' \setminus M)$.
  - Output $(\gamma_d, \Sigma'_d)_{d=1}^{D+1}$. (Partitioned as $\sigma_j := ((\gamma_d)_{d=1}^{D+1}, \sigma'_j)$.)
- $b \leftarrow \mathsf{Vf}(\mathsf{pk}, \tau, \sigma, m, g)$
  - Extract from pk the delegation chains $(\mathsf{pk}_{k,d_k})_{k=1,d_k=1}^{K,D_k}$ and parse pk_{k,d_k} as $(\mathsf{pk}_{\mathcal{HS},k,d_k}, \mathsf{pk}_{\mathcal{DS},k,d_k})$.
  - If $K = 1$ and $D_K = 1$, output $b \leftarrow \mathcal{HS}.\mathsf{Vf}(\mathsf{pk}_{1,1}, \tau, \sigma, m, g)$.
  - Else, parse σ as $((\gamma_{k,d_k})_{k=1,d_k=1}^{K,D_k}, \sigma')$.
  - Compute $b_0 \leftarrow \mathcal{HS}.\mathsf{Vf}(\mathsf{pk}_{\mathcal{HS}}, \tau, \sigma', m, g)$.
  - Compute $b_{k,d_k} \leftarrow \mathcal{DS}.\mathsf{Vf}(\mathsf{pk}_{\mathcal{DS},k,d_k}, (\mathsf{pk}_{d_k+1}, H(\tau_{k,1}, \ldots, \tau_{k,d_k})), \gamma_{k,d_k})$ for $k \in [K]$ and $d_k \in [D_k - 1]$.
  - Compute $b_{k,D_k} \leftarrow \mathcal{DS}.\mathsf{Vf}(\mathsf{pk}_{\mathcal{DS},k,D_k}, H(\tau_{k,1}, \ldots, \tau_{k,D_k}), \gamma_{k,D_k})$ for $k \in [K]$.
  - Output $\cap_{k=1,d_k=1}^{K,D_k} b_{k,d_k} \cap b_0$.
- $\sigma \leftarrow \mathsf{Eval}(g, (\mathsf{pk}_k, \tau_k, M_k, \Sigma_k)_{k=1}^{K})$
  - Extract from pk_k the delegation chains $(\mathsf{pk}_{k,d_k})_{d_k=1}^{D_k}$ and parse pk_{k,d_k} as $(\mathsf{pk}_{\mathcal{HS},k,d_k}, \mathsf{pk}_{\mathcal{DS},k,d_k})$.
  - Partition M_k as $(M_{k,1}, \ldots, M_{k,D_k})$.
  - Parse Σ_k as $(\gamma_{k,d_k}, \Sigma'_{k,d_k})_{d_k=1}^{D_k}$.
  - Compute $\sigma' \leftarrow \mathcal{HS}.\mathsf{Eval}(g, (\mathsf{pk}_{\mathcal{HS},k,d_k}, \tau_{k,d_k}, M_{k,d_k}, \Sigma'_{k,d_k})_{k=1,d_k=1}^{K,D_k})$.
  - Output $((\gamma_{k,d_k})_{k=1,d_k=1}^{K,D_k}, \sigma')$. (Partitioned as $\sigma_j := ((\gamma_{k,d_k})_{k=1,d_k=1}^{K,D_k}, \sigma'_j)$.)

The correctness and context-hiding property follow straightforwardly from those of $\mathcal{HS}$. In the following, we give the proof sketches for unforgeability and succinctness.

Theorem 11. *Suppose $\mathcal{HS}$ is unforgeable under K-bounded corruption, $\mathcal{DS}$ is EUF-CMA-secure, and H is collision-resistant, then $\mathcal{HS}'$ is unforgeable under K-bounded corruption.*

Proof. 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 $\mathcal{A}_{\mathcal{HS}}$ that uses $\mathcal{A}_{\mathcal{HS}'}$ to produce a forgery of $\mathcal{HS}$. $\mathcal{A}_{\mathcal{HS}}$ acts as a challenger in the unforgeability game of $\mathcal{HS}'$.

$\mathcal{A}_{\mathcal{HS}}$ received pp and $\{\mathsf{pk}_{\mathcal{HS},k}\}_{k=1}^{\mathfrak{K}}$ from the challenger of the unforgeability game of $\mathcal{HS}$. It generates the keys for $\mathcal{DS}$ honestly and forwards pp and the public keys $\{\mathsf{pk}_k\}_{k=1}^{\mathfrak{K}}$ to $\mathcal{A}_{\mathcal{HS}'}$.

$\mathcal{A}_{\mathcal{HS}'}$ makes three types of queries. For delegation and signing queries, $\mathcal{A}_{\mathcal{HS}}$ signs the γ parts honestly using $\mathcal{DS}$ and queries the challenger of $\mathcal{HS}$ for the signatures σ'. For corruption queries on k, it forwards the query to corruption oracle of $\mathcal{HS}$. It then forwards the response from the challenger of $\mathcal{HS}$, as well as the $\mathcal{DS}$ secret key $\mathsf{sk}_{\mathcal{DS},k}$ to $\mathcal{A}_{\mathcal{HS}'}$. After querying the oracles, $\mathcal{A}_{\mathcal{HS}'}$ responds with forgery, a public key pk^*, a tag $\tau^* \in \{0,1\}^*$, a function $g^* \in \mathcal{G}$, some data $m^* \in \mathcal{M}$, and a signature σ^*. The only ways for the forgery to be valid is to either forge the underlying $\mathcal{HS}$ signature, the $\mathcal{DS}$ signature, or find a collision in H. Since the latter two are infeasible, for otherwise we can construct adversaries for breaking $\mathcal{DS}$ or H, $\mathcal{A}_{\mathcal{HS}}$ simply answers the σ' part of σ^* to its unforgeability game.

Theorem 12. *Suppose $\mathcal{HS}$ is succinct, then $\mathcal{HS}'$ is succinct.*

Proof. The succinctness of the σ' part of a signature is trivial. For the γ part, succinctness follows since there is one $\mathcal{DS}$ signature on a λ-bit message for each member of each delegation chain.

4.2 Multi-Key Key-Message Homomorphic Signatures (M-KMHS)

Another extension of M-HS is the multi-key key-message homomorphic signatures (M-KMHS). In M-KMHS, homomorphic operations are not only with respect to the message space, but also the key space. Computation over randomly generated public keys is however not always meaningful. To make computation over keys more meaningful, we adopt the terminology of Boneh *et al.* [BGG+14] where the public keys are actually attributes, whose corresponding secret keys are generated from their respective authorities. More concretely, suppose an authority authorizes the set of attributes X and X' of two signers respectively. The signer with attributes X then issues a signature on a set of messages M, and the signer attributed X' also signs M'. Given the signatures, a public evaluator can then derive a signature on the message $g((X,M),(X',M'))$ attributed to $f(X,X')$ for any functions f and g.

Definitions

Syntax. A key-message homomorphic signature scheme consists of the following six PPT algorithms (Setup, $\mathsf{KGen}_{\mathrm{Auth}}, \mathsf{KGen}_{\mathrm{Sig}}, \mathsf{Auth}, \mathsf{Sig}, \mathsf{Vf}, \mathsf{Eval}$) defined as follows:

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ is the same as that of M-HS. The public parameter additionally defines the attribute space $\mathcal{X}$ and the function family $\mathcal{F}$.
- $(\mathsf{apk}, \mathsf{ask}) \leftarrow \mathsf{KGen}_{\mathrm{Auth}}(\mathsf{pp})$ is a new key generation algorithm for generating an authority public key apk and an authority secret key ask.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}_{\mathrm{Sig}}(\mathsf{pp})$ is the same as that of M-HS.
- $\Gamma \leftarrow \mathsf{Auth}(\mathsf{ask}, \mathsf{pk}, X)$ is a new authentication algorithm which inputs an authority secret key ask, the public key of the signer pk, and a set of attributes X. It outputs a set of credentials Γ.
- $\Sigma \leftarrow \mathsf{Sig}(\mathsf{sk}, \tau, M)$ is the same as that of M-HS.
- $b \leftarrow \mathsf{Vf}(\mathsf{apk}, \mathsf{pk}, \tau, \gamma, \sigma, x, m, f, g)$ is the same as that of M-HS, except that it also verifies the certificate of the attribute. Concretely, it takes as input an additional authority public key apk, credential γ, attribute $x \in \mathcal{X}$, and a function $f \in \mathcal{F}$.

– $(\gamma, \sigma) \leftarrow \mathsf{Eval}(f, g, (\mathsf{apk}_k, \mathsf{pk}_k, \tau_k, X_k, \Gamma_k, M_k, \Sigma_k)_{k=1}^K)$ is the same as that of M-HS, except that it also evaluates the attributes. Concretely, for each input tuple corresponding to the K signers, it inputs an additional authority public key apk_k, a set of attributes X_k and credentials Γ_k. It also takes an additional function $f \in \mathcal{F}$. It outputs a new credential γ along with the new signature σ.

If the evaluation and verification of the attribute part are replaced by dummy operations, then M-KMHS scheme essentially becomes an M-HS scheme. The correctness, unforgeability, and context-hiding property are defined similarly to those of M-HS. A major difference is that, in the unforgeability game, the adversary is in addition allowed to corrupt the authorities and query an authentication oracle, and the condition of valid forgery is adjusted accordingly to avoid trivial attacks. Furthermore, the context-hiding property now also hides the attributes input to the evaluation algorithm.

Correctness. A simple way to define the correctness of M-KMHS is to view $\mathsf{KGen}_{\mathrm{Auth}}$ and Auth as the key generation and signing algorithms of a separate M-HS scheme respectively. The evaluation correctness of M-KMHS requires that the evaluation correctness of the two separate M-HS holds simultaneously. Concretely, for any $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, any $K, L = \mathsf{poly}(\lambda)$, $k \in [K]$, $\ell \in [L]$, any $(\mathsf{apk}_\ell, \mathsf{ask}_\ell) \in \mathsf{KGen}_{\mathrm{Auth}}(\mathsf{pp})$ any $(\mathsf{pk}_k, \mathsf{sk}_k) \in \mathsf{KGen}_{\mathrm{Sig}}(\mathsf{pp})$, any $X_{k,\ell} \in \mathcal{X}^*$, any $M_k \in \mathcal{M}^*$, any $\tau_k \in \{0,1\}^*$, any $r_{k,i}, f \in \mathcal{F}$ and $s_{k,j}, g \in \mathcal{G}$ with appropriate dimensions, the following correctness requirements hold.

– (Authentication and Signing Correctness.) If $\Gamma_{k,\ell} \in \mathsf{Auth}(\mathsf{ask}_\ell, \mathsf{pk}_k, X_{k,\ell})$ and $\Sigma_k \to \mathsf{Sig}(\mathsf{sk}_k, \tau_k, M_k)$, then for all $(x_{k,\ell,i}, m_{k,j}) \in X_{k,\ell} \times M_k$ and $(\gamma_{k,\ell,i}, \sigma_{k,j}) \in \Gamma_{k,\ell} \times \Sigma_k$,

$$\mathsf{Vf}(\mathsf{apk}_\ell, \mathsf{pk}_k, \tau_k, (\gamma_{k,\ell,i}, \sigma_{k,j}), (x_{k,\ell,i}, m_{k,j}), \mathsf{id}, \mathsf{id}) = 1.$$

– (Evaluation Correctness.) For any Γ_k and Σ_k, if for all $(x_{k,i}, m_{k,j}) \in X_k \times M_k$ and $(\gamma_{k,i}, \sigma_{k,j}) \in \Gamma_k \times \Sigma_k$

$$\mathsf{Vf}(\mathsf{apk}_k, \mathsf{pk}_k, \tau_k, (\gamma_{k,i}, \sigma_{k,j}), (x_{k,i}, m_{k,j}), r_{k,i}, s_{k,j}) = 1;$$

and $(\gamma, \sigma) \leftarrow \mathsf{Eval}(f, g, (\mathsf{apk}_k, \mathsf{pk}_k, \tau_k, X_k, \Gamma_k, M_k, \Sigma_k)_{k=1}^K)$ it holds that

$$\mathsf{Vf}(\mathsf{apk}, \mathsf{pk}, \tau, \gamma, \sigma, f(X_1, \ldots, X_K), g((X_1, M_1), \ldots, (X_K, M_k), f \circ \bar{r}, g \circ \bar{s}) = 1,$$

where $\mathsf{apk} = f(\mathsf{apk}_1, \ldots, \mathsf{apk}_K)$, $\mathsf{pk} = g(\mathsf{pk}_1, \ldots, \mathsf{pk}_K)$, and $\tau = g(\tau_1, \ldots, \tau_K)$.

Unforgeability. Consider the following security game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

– Let $\mathfrak{L}, \mathfrak{K} = \mathsf{poly}(\lambda)$ be the number of authorities and the number of signers in the system respectively.
– Initialize an empty dictionary D.
– The challenger $\mathcal{C}$ runs $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $\{(\mathsf{apk}_\ell, \mathsf{ask}_\ell) \leftarrow \mathsf{KGen}_{\mathrm{Auth}}(\mathsf{pp})\}_{\ell=1}^{\mathfrak{L}}$, $\{(\mathsf{pk}_k, \mathsf{sk}_k) \leftarrow \mathsf{KGen}_{\mathrm{Sig}}(\mathsf{pp})\}_{k=1}^{\mathfrak{K}}$, and gives $(\mathsf{apk}_1, \ldots, \mathsf{apk}_{\mathfrak{L}})$ and $(\mathsf{pk}_1, \ldots, \mathsf{pk}_{\mathfrak{K}})$ to $\mathcal{A}$.
– The adversary $\mathcal{A}$ makes four types of queries:
  • $\mathcal{O}_{\mathrm{CorrAuth}}(\ell)$ The challenger responds with the authority secret key ask_ℓ. Without loss of generality, suppose authority $1, \ldots, L$ are corrupt.
  • $\mathcal{O}_{\mathrm{CorrSig}}(k)$ The challenger responds with the signer secret key sk_k. Without loss of generality, suppose signer $1, \ldots, K$ are corrupt.
  • $\mathcal{O}_{\mathsf{Auth}}(\ell, k, X_{k,\ell})$ If there exists a key for the entry (k, ℓ) in the dictionary D, return $\perp$. Otherwise, the challenger returns $\Gamma_{k,\ell} \leftarrow \mathsf{Auth}(\mathsf{ask}_\ell, \mathsf{pk}_k, X_{k,\ell})$ and add $\Gamma_{k,\ell}$ to the entry (k, ℓ) in the dictionary D.
  • $\mathcal{O}_{\mathsf{Sig}}(q, k, \tau_{q,k}, M_{q,k})$ It returns $\Sigma_{q,k} \leftarrow \mathsf{Sig}(\mathsf{sk}, \tau_{q,k}, M_{q,k})$.
– Eventually the adversary $\mathcal{A}$ outputs an authority public key apk^*, a signer public key pk^*, a tag τ^*, a signature σ^*, two functions $f^* \in \mathcal{F}$, $g^* \in \mathcal{G}$, an attribute x^*, and a message m^*. It wins the game if $\mathsf{Vf}(\mathsf{apk}^*, \mathsf{pk}^*, \tau^*, \gamma^*, \sigma^*, x^*, m^*, f^*, g^*) = 1$ and the forgery is either of:
  • *Type-I*: τ^* is not composed of the previously queried $\tau_{q,k}$, and at least one authority ℓ or one signer k is not corrupted, or
  • *Type-II*: τ^* is composed of the previously queried $\tau_{q,k}$, but either x^* is not in the range of f^* restricted by the inputs of the corresponding uncorrupted authorities, or m^* is not in the range of g^* restricted by the inputs of the corresponding uncorrupted authorities and signers..

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

Construction In the previous subsection, we show that M-HiHS can be constructed from M-HS, hence showing the equivalence of their existences. We now construct M-KMHS using M-HS, which in turn shows the equivalence of all three notions.

Let $\mathcal{HS}.(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf}, \mathsf{Eval})$ be an M-HS scheme. We construct an M-KMHS scheme $\mathcal{HS}'$ as follows:

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
  - Output $\mathsf{pp} \leftarrow \mathcal{HS}.\mathsf{Setup}(1^\lambda)$.
- $(\mathsf{apk}, \mathsf{ask}) \leftarrow \mathsf{KGen}_{\mathrm{Auth}}(\mathsf{pp})$
  - Output $(\mathsf{apk}, \mathsf{ask}) \leftarrow \mathcal{HS}.\mathsf{KGen}(\mathsf{pp})$.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}_{\mathrm{Sig}}(\mathsf{pp})$
  - Output $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{HS}.\mathsf{KGen}(\mathsf{pp})$.
- $\Gamma \leftarrow \mathsf{Auth}(\mathsf{ask}, \mathsf{pk}, X)$
  - Output $\Gamma \leftarrow \mathcal{HS}.\mathsf{Sig}(\mathsf{ask}, \mathsf{pk}, X)$.
- $\Sigma \leftarrow \mathsf{Sig}(\mathsf{sk}, \tau, M)$
  - Output $\Sigma \leftarrow \mathcal{HS}.\mathsf{Sig}(\mathsf{sk}, \tau, M)$.
- $b \leftarrow \mathsf{Vf}(\mathsf{apk}, \mathsf{pk}, \tau, \gamma, \sigma, x, m, f, g)$
  - Compute $b_0 \leftarrow \mathcal{HS}.\mathsf{Vf}(\mathsf{apk}, \mathsf{pk}, \gamma, x, f)$.
  - Compute $b_1 \leftarrow \mathcal{HS}.\mathsf{Vf}(\mathsf{pk}, \tau, \sigma, m, g)$.
  - Output $b = b_0 \cap b_1$.
- $(\gamma, \sigma) \leftarrow \mathsf{Eval}(f, g, (\mathsf{apk}_k, \mathsf{pk}_k, \tau_k, X_k, \Gamma_k, M_k, \Sigma_k)_{k=1}^K)$
  - Compute $\gamma \leftarrow \mathcal{HS}.\mathsf{Eval}(f, (\mathsf{apk}_k, \mathsf{pk}_k, X_k, \Gamma_k)_{k=1}^K)$.
  - Compute $\sigma \leftarrow \mathcal{HS}.\mathsf{Eval}(g, (\mathsf{pk}_k, \tau_k, (X_k, M_k), (\Gamma_k, \Sigma_k))_{k=1}^K)$.
  - Output (γ, σ).

The correctness, unforgeability, and context-hiding property follow straightforwardly from those of $\mathcal{HS}$.

4.3 Multi-Key Key-Homomorphic Signatures (M-KHS)

Recall from the definition of M-KMHS in Section 4.2, if the verification algorithm always take the identity function $g = \mathsf{id}$ as input, we obtain a multi-key signature scheme which is only homomorphic in the attribute space, which we refer to as multi-key key-homomorphic signature scheme (M-KHS). In such configuration, the construction in Section 4.2 is actually a transformation from M-HS to M-KHS. Furthermore, the message can actually be signed by an ordinary signature scheme instead of a M-HS scheme.

In this section, we show that homomorphisms in the attribute space and the message space are indeed equivalent. To facilitate our discussion, we formalize the syntax of M-KHS.

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

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ is the same as that of M-KMHS.
- $(\mathsf{apk}, \mathsf{ask}) \leftarrow \mathsf{KGen}_{\mathrm{Auth}}(\mathsf{pp})$ is the same as that of M-KMHS.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}_{\mathrm{Sig}}(\mathsf{pp})$ is the same as that of M-KMHS.
- $\Gamma \leftarrow \mathsf{Auth}(\mathsf{ask}, \mathsf{pk}, X)$ is the same as that of M-KMHS.
- $\Sigma \leftarrow \mathsf{Sig}(\mathsf{sk}, \tau, M)$ is the same as that of M-KMHS.
- $b \leftarrow \mathsf{Vf}(\mathsf{apk}, \mathsf{pk}, \tau, \gamma, \sigma, x, m, f)$ is the same as that of M-KMHS, except that the function g is fixed to the identity function id.
- $\gamma \leftarrow \mathsf{Eval}(f, (\mathsf{apk}_k, \mathsf{pk}_k, X_k, \Gamma_k)_{k=1}^K)$ is the same as that of M-KMHS, except that it only evaluates the attributes.

From the syntax, it is apparent that we can interpret the attribute space of M-KHS as the message space of M-HS, and obtain a construction of M-HS. In particular, we conclude that the existence of ordinary HS and single-authority KHS are equivalent.

4.4 Other Extensions of M-HS

In the previous subsections, we study two extensions of M-HS, namely M-HiHS and M-KMHS, and show that they can both be constructed from M-HS, hence proving the equivalence of their existences. Recall that M-HiHS extends M-HS vertically in the sense that it allows the delegation of signing power down the delegation chains. M-KMHS extends M-HS in another dimension as it introduces additional homomorphism to the key space. We note that other extensions, including but not limited to combing the hierarchy of M-HiHS and the key-homomorphism in M-KMHS, can likely be also constructed from M-HS. This illustrates the power of supporting homomorphism in more than one spaces, and the power of corruption resistance in homomorphic signatures.

5 Decentralized Attribute-based Signatures

Apart from the natural application of allowing delegation of computation on data authenticated by multiple parties, we study the implications of M-HS to other primitives. Specifically, we propose a new construction of decentralized attribute-based signatures (D-ABS). A D-ABS scheme allows multiple authorities to certify different sets of attributes of a signer in a completely distributed manner. After obtaining the certificates from the authorities, the signer can then issue signatures on messages, while at the same time show that its certified attributes satisfy certain access policy. We show how to construct from KHS a D-ABS scheme which supports access policies in the class of admissible functions $\mathcal{F}$ of the KHS scheme.

Yet, this scheme can only achieve a weaker notion of anonymity due to the tag-based nature of KHS. This weaker property, which we call linkable anonymity, states that it is infeasible to learn any information about the attributes behind the signatures except those leaked from the access policies. On one hand, linkability is a useful feature to achieve strong accountability. For example, consider a simple membership system where a user can register by issuing a linkable attribute-based signature, so that the server can use the linkable part of the signature as the identity of the user. Indeed there is a branch of literature which incorporates various forms of linkability into signatures or credentials. On the other hand, one can generically transform this linkable scheme to an unlinkable one: Simply replace the signature by a non-interactive witness-indistinguishable (NIWI) proof of the knowledge of the tag in the KHS.

5.1 Definitions

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

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ inputs the security parameter λ and outputs the public parameter pp, which defines the attribute space $\mathcal{X}$ and the message space $\mathcal{M}$.
- $(\mathsf{apk}, \mathsf{ask}) \leftarrow \mathsf{KGen}_{\mathrm{Auth}}(\mathsf{pp})$ inputs the public parameter and outputs an authority public key apk and an authority secret key ask.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}_{\mathrm{Sig}}(\mathsf{pp})$ inputs the public parameter and outputs a signer public key pk and a signer secret key sk.
- $\mathsf{sk}_X \leftarrow \mathsf{Auth}(\mathsf{ask}, \mathsf{pk}, X)$ inputs an authority secret key ask, a signer public key pk, and a set of attributes $X \in \mathcal{X}^*$. It outputs a secret key sk_X corresponding to the attributes X.
- $\sigma \leftarrow \mathsf{Sig}(\mathsf{sk}, (\mathsf{sk}_{X_k})_{k=1}^K, f, m)$ inputs the signer secret key sk, a set of secret keys sk_{X_k} for the attributes X_k issued by authority k, an access policy $f \in \mathcal{F}$, and a message m. It outputs a signature σ signing m under the attributes $(X_1, \ldots, X_k)$ and policy f.
- $b \leftarrow \mathsf{Vf}((\mathsf{apk}_k)_{k=1}^K, \sigma, f, m)$ inputs the authority public key apk_k of each of the authorities, a signature σ, an access policy f, and a message m. It outputs 1 if the signature is valid, 0 otherwise.

Correctness. For any $\mathsf{pp} \in \mathsf{Setup}(1^\lambda)$, any $K, j = \mathsf{poly}(\lambda)$ and $k \in [K]$, any $(\mathsf{apk}_k, \mathsf{ask}_k) \in \mathsf{KGen}_{\mathrm{Auth}}(\mathsf{pp})$, any $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{KGen}_{\mathrm{Sig}}(\mathsf{pp})$, any $X_k \in \mathcal{X}^*$, any $\mathsf{sk}_{X_k} \in \mathsf{Auth}(\mathsf{ask}_k, \mathsf{pk}, X_k)$, any policy $f \in \mathcal{F}$, and any message $m \in \mathcal{M}$, it holds that if $\sigma \in \mathsf{Sig}(\mathsf{sk}, (\mathsf{sk}_{X_k})_{k=1}^K, f, m)$, then $\mathsf{Vf}((\mathsf{apk}_k)_{k=1}^K, \sigma, f, m) = f(X_1, \ldots, X_k)$.

Unforgeability. Consider the following security game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

- Let $\mathfrak{K} = \mathsf{poly}(\lambda)$ be the maximum number of authorities in the system.
- Initialize an empty dictionary D.
- The challenger $\mathcal{C}$ runs $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $\{(\mathsf{apk}_k, \mathsf{ask}_k) \leftarrow \mathsf{KGen}_{\mathrm{Auth}}(\mathsf{pp})\}_{k=1}^{\mathfrak{K}}$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}_{\mathrm{Sig}}(\mathsf{pp})$, and gives $(\mathsf{apk}_k)_{k=1}^{\mathfrak{K}}$ and pk to $\mathcal{A}$.
- The adversary $\mathcal{A}$ makes four types of queries:
  - $\mathcal{O}_{\mathrm{Auth}}(k)$ The challenger responds with the authority secret key ask_k. Without loss of generality, suppose authority $1, \ldots, K$ are corrupt.
  - $\mathcal{O}_{\mathrm{Sig}}()$ The challenger responds with the signer secret key sk.
  - $\mathcal{O}_{\mathsf{Auth}}(k, X_k)$ If there exists a key for the entry k in the dictionary D, return $\perp$. Otherwise, the challenger returns $\mathsf{sk}_{X_k} \leftarrow \mathsf{Auth}(\mathsf{ask}_h k, \mathsf{pk}, X_k)$ and add sk_{X_k} to the entry k in the dictionary D.
  - $\mathcal{O}_{\mathsf{Sig}}(S, f, m)$ S is a set of authorities chosen by $\mathcal{A}$. If $S \not\subseteq D$, the challenger returns $\perp$. Otherwise, for each $k \in S$, the challenger retrieves sk_{X_k} from the entry k in the dictionary D. It returns $\sigma \in \mathsf{Sig}(\mathsf{sk}_k, (\mathsf{sk}_{X_k})_{k \in D}, f, m)$.
- Eventually the adversary $\mathcal{A}$ outputs a set of authority public keys $(\mathsf{apk}_k^*)_{k \in S^*}$, a signature σ^*, a policy $f^* \in \mathcal{F}$, and a message m^*. It wins the game if the following holds:
  - $\mathsf{Vf}((\mathsf{apk}_k^*)_{k \in S^*}, \sigma^*, f^*, m^*) = 1$, and
  - (S^*, f^*, m^*) was not queried to the sign oracle before, and
  - $1 \notin f^*(\cdot, \ldots, \cdot, X_{K+1}, \ldots, X_{\mathfrak{K}})$ for all (k, X_k) queried to the $\mathcal{O}_{\mathsf{Auth}}$ oracle, or the signer is not corrupted.

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

Linkable Anonymity. We require that there exist a simulator $\mathcal{S} = (\mathcal{S}^{\mathsf{Setup}}, \mathcal{S}^{\mathsf{Sig}})$ such that, for all $K = \mathsf{poly}(\lambda)$, $k \in [K]$, m, X_k, and f, it holds that for any PPT adversaries $\mathcal{A}$,

$$
\left| \Pr \left[\begin{array}{c} \mathcal{A}(\mathsf{pk}, \mathsf{sk}, (\mathsf{apk}_k, \mathsf{ask}_k, \mathsf{sk}_{X_k}, X_k)_{k=1}^K, \sigma, f, m) \to 1 : \\ \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{apk}_k, \mathsf{ask}_k) \leftarrow \mathsf{KGen}_{\mathrm{Auth}}(\mathsf{pp}) \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}_{\mathrm{Sig}}(\mathsf{pp}) \\ \mathsf{sk}_{X_k} \leftarrow \mathsf{Auth}(\mathsf{ask}_k, \mathsf{pk}, X_k) \\ \sigma \leftarrow \mathsf{Sig}(\mathsf{sk}, (\mathsf{sk}_{X_k})_{k=1}^K, f, m) \end{array} \right] - \right.
$$

$$
\left. \Pr \left[\begin{array}{c} \mathcal{A}(\mathsf{pk}, \mathsf{sk}, (\mathsf{apk}_k, \mathsf{ask}_k, \mathsf{sk}_{X_k}, X_k)_{k=1}^K, \sigma, f, m) \to 1 : \\ (\mathsf{pp}, \mathsf{td}) \leftarrow \mathcal{S}^{\mathsf{Setup}}(1^\lambda) \\ (\mathsf{apk}_k, \mathsf{ask}_k) \leftarrow \mathsf{KGen}_{\mathrm{Auth}}(\mathsf{pp}) \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}_{\mathrm{Sig}}(\mathsf{pp}) \\ \mathsf{sk}_{X_k} \leftarrow \mathsf{Auth}(\mathsf{ask}_k, \mathsf{pk}, X_k) \\ \sigma \leftarrow \mathcal{S}^{\mathsf{Sig}}(\mathsf{td}, f, \mathsf{pk}, (\mathsf{apk}_k)_{k=1}^K, f(X_1, \ldots, X_K), m) \end{array} \right] \right| = \mathsf{negl}(\lambda) .
$$

The signatures are linkable in the sense that they leak the signer public key pk.

5.2 Construction

Using M-KHS, we immediately get a generic construction of decentralized attribute-based signatures (D-ABS). Let $\mathcal{HS}.(\mathsf{Setup}, \mathsf{KGen}_{\mathrm{Auth}}, \mathsf{KGen}_{\mathrm{Sig}}, \mathsf{Auth}, \mathsf{Sig}, \mathsf{Vf}, \mathsf{Eval})$ be an M-KHS scheme. We present the construction of D-ABS $\mathcal{HS}'$ as follows:

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
  - Output $\mathsf{pp} \leftarrow \mathcal{HS}.\mathsf{Setup}(1^\lambda)$.
- $(\mathsf{apk}, \mathsf{ask}) \leftarrow \mathsf{KGen}_{\mathrm{Auth}}(\mathsf{pp})$
  - Output $(\mathsf{apk}, \mathsf{ask}) \leftarrow \mathcal{HS}.\mathsf{KGen}_{\mathrm{Auth}}(\mathsf{pp})$.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}_{\mathrm{Sig}}(\mathsf{pp})$

  - Output $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{HS}.\mathsf{KGen}_{\mathrm{Sig}}(\mathsf{pp})$.
- $\mathsf{sk}_X \leftarrow \mathsf{Auth}(\mathsf{ask}, \mathsf{pk}, X)$
  - Run $\Gamma \leftarrow \mathcal{HS}.\mathsf{Auth}(\mathsf{ask}, \mathsf{pk}, X)$.
  - Output $\mathsf{sk}_X = (X, \Gamma)$.
- $\sigma \leftarrow \mathsf{Sig}(\mathsf{sk}, (\mathsf{sk}_{X_k})_{k=1}^K, f, m)$
  - Parse $\mathsf{sk}_{X_k} = (X_k, \Gamma_k)$
  - Compute $\sigma' \leftarrow \mathcal{HS}.\mathsf{Sig}(\mathsf{sk}, 0, m)$.
  - Compute $\gamma \leftarrow \mathcal{HS}.\mathsf{Eval}(f, (\mathsf{apk}_k, \mathsf{pk}, X_k, \Gamma_k)_{k=1}^K)$.
  - Output $\sigma = (\mathsf{pk}, \gamma, \sigma')$.
- $b \leftarrow \mathsf{Vf}(\mathsf{mpk}, \sigma, f, m)$
  - Parse $\sigma = (\mathsf{pk}, \gamma, \sigma')$.
  - Output $b \leftarrow \mathcal{HS}.\mathsf{Vf}(f(\mathsf{apk}_1, \ldots, \mathsf{apk}_K), \mathsf{pk}, 0, \gamma, \sigma', 1, m, f)$.

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

Drawing the connections from Section 4, it is easy to see that the above D-ABS scheme can be extended to support homomorphism in the message space (by M-KMHS). It can also be extended (by M-HiHS) so that there is a hierarchy of authorities certifying multiple layers of attributes.

5.3 Instantiations

We can instantiate the above generic construction from the multi-key generalization of the fully homomorphic signature scheme by Gorbunov *et al.* [GVW15] (GVW). The following discussions are in order.

- We obtain a D-ABS scheme for general circuits based on lattice assumptions in the standard model.
- In our terminology, the generalized GVW is an M-HS scheme unforgeable under 0-bounded corruption. This implies a weak unforgeability of the resulting D-ABS scheme in the sense that the adversary is only allowed to corrupt either the signer or any subset of the authorities.
- In (generalized) GVW, normal verification takes as long as computing the function f. However, as explained in details [GVW15], part of the verification can be pre-computed so that the amortized verification cost with the fixed function f can be made constant. This is suitable for our purpose as the access policies of a verifier in D-ABS typically remain relatively stable.
- The transformation to a fully anonymous scheme can be instantiated by the recent proof system for linear congruences proposed by Libert *et al.* [LLM+16].

6 Concluding Remark

The study of homomorphic signatures (HS) is in a single-key setting concentrating on the homomorphism in the message space. In this work, we have introduced the notion of multi-key homomorphic signatures with a strong security property known as unforgeability under corruption.

Despite of having a relatively simple syntax, multi-key HS turns out to be a central-hub of various seemingly more powerful or at least different variants of homomorphic signatures. Specifically, it implies multi-key hierarchical HS, multi-key key-message-HS, and multi-key key-HS. The equivalence of multi-key HS and multi-key key-HS suggests that message-homomorphism and key-homomorphism in signatures are identical in nature. Thus, existing fully homomorphic signature schemes readily give fully key-homomorphic signature schemes. We also show that it further implies attribute-based signatures for general circuits from standard assumptions. It is unknown that whether there exist some flavors of HS which are not covered by multi-key HS.

We have constructed a multi-key HS scheme from ZK-SNARK, and shown that the existence of corruption resistant HS implies the existence of ZK-SNARG. It will be interesting to design a corruption-resistant HS scheme from different primitives or assumptions.

Acknowledgments

We thank some of the anonymous reviewers of Asiacrypt 2016 for their detailed and helpful comments. We also thank Yvo Desmedt and Daniel Wichs for inspiring discussions.

References

[ABC+12] Jae Hyun Ahn, Dan Boneh, Jan Camenisch, Susan Hohenberger, abhi shelat, and Brent Waters. Computing on authenticated data. In Ronald Cramer, editor, *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, pages 1–20, Taormina, Sicily, Italy, March 19–21, 2012. Springer, Heidelberg, Germany.

[ALP12] Nuttapong Attrapadung, Benoît Libert, and Thomas Peters. Computing on authenticated data: New privacy definitions and constructions. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 367–385, Beijing, China, December 2–6, 2012. Springer, Heidelberg, Germany.

[ALP13] Nuttapong Attrapadung, Benoît Libert, and Thomas Peters. Efficient completely context-hiding quotable and linearly homomorphic signatures. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013: 16th International Conference on Theory and Practice of Public Key Cryptography*, volume 7778 of *Lecture Notes in Computer Science*, pages 386–404, Nara, Japan, February 26 – March 1, 2013. Springer, Heidelberg, Germany.

[BDF+14] Michael Backes, Özgür Dagdelen, Marc Fischlin, Sebastian Gajek, Sebastian Meiser, and Dominique Schröder. Operational signature schemes. *IACR Cryptology ePrint Archive*, 2014:820, 2014.

[BF11a] Dan Boneh and David Mandell Freeman. Homomorphic signatures for polynomial functions. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 149–168, Tallinn, Estonia, May 15–19, 2011. Springer, Heidelberg, Germany.

[BF11b] Dan Boneh and David Mandell Freeman. Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011: 14th International Conference on Theory and Practice of Public Key Cryptography*, volume 6571 of *Lecture Notes in Computer Science*, pages 1–16, Taormina, Italy, March 6–9, 2011. Springer, Heidelberg, Germany.

[BF14] Mihir Bellare and Georg Fuchsbauer. Policy-based signatures. In Krawczyk [Kra14], pages 520–537.

[BFKW09] Dan Boneh, David Freeman, Jonathan Katz, and Brent Waters. Signing a linear subspace: Signature schemes for network coding. In Stanislaw Jarecki and Gene Tsudik, editors, *PKC 2009: 12th International Conference on Theory and Practice of Public Key Cryptography*, volume 5443 of *Lecture Notes in Computer Science*, pages 68–87, Irvine, CA, USA, March 18–20, 2009. Springer, Heidelberg, Germany.

[BFS14] Xavier Boyen, Xiong Fan, and Elaine Shi. Adaptively secure fully homomorphic signatures based on lattices. Cryptology ePrint Archive, Report 2014/916, 2014. http://eprint.iacr.org/2014/916.

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 533–556, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Krawczyk [Kra14], pages 501–519.

[BMS16] Michael Backes, Sebastian Meiser, and Dominique Schröder. Delegatable functional signatures. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016: 19th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 9614 of *Lecture Notes in Computer Science*, pages 357–386, Taipei, Taiwan, March 6–9, 2016. Springer, Heidelberg, Germany.

[CFW12] Dario Catalano, Dario Fiore, and Bogdan Warinschi. Efficient network coding signatures in the standard model. In Fischlin et al. [FBM12], pages 680–696.

[CFW14] Dario Catalano, Dario Fiore, and Bogdan Warinschi. Homomorphic signatures with efficient verification for polynomial functions. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 371–389, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.

[CM15] Michael Clear and Ciaran McGoldrick. Multi-identity and multi-key leveled FHE from learning with errors. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 630–656, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.

[FBM12] Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors. *PKC 2012: 15th International Conference on Theory and Practice of Public Key Cryptography*, volume 7293 of *Lecture Notes in Computer Science*, Darmstadt, Germany, May 21–23, 2012. Springer, Heidelberg, Germany.

[FMNP16] Dario Fiore, Aikaterini Mitrokotsa, Luca Nizzardo, and Elena Pagnin. Multi-key homomorphic authenticators. In *Advances in Cryptology - ASIACRYPT 2016*, 2016. To appear.

[FN16] Dario Fiore and Anca Nitulescu. On the (in)security of SNARKs in the presence of oracles. In *TCC 2016: 13th Theory of Cryptography Conference*, 2016. To appear.

[Fre12] David Mandell Freeman. Improved security for linearly homomorphic signatures: A generic framework. In Fischlin et al. [FBM12], pages 697–714.

[GKKR10] Rosario Gennaro, Jonathan Katz, Hugo Krawczyk, and Tal Rabin. Secure network coding over the integers. In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010: 13th International Conference on Theory and Practice of Public Key Cryptography*, volume 6056 of *Lecture Notes in Computer Science*, pages 142–160, Paris, France, May 26–28, 2010. Springer, Heidelberg, Germany.

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled fully homomorphic signatures from standard lattices. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th Annual ACM Symposium on Theory of Computing*, pages 469–477, Portland, OR, USA, June 14–17, 2015. ACM Press.

[Kra14] Hugo Krawczyk, editor. *PKC 2014: 17th International Conference on Theory and Practice of Public Key Cryptography*, volume 8383 of *Lecture Notes in Computer Science*, Buenos Aires, Argentina, March 26–28, 2014. Springer, Heidelberg, Germany.

[Lam79] Leslie Lamport. Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, October 1979.

[LLM+16] Benoît Libert, San Ling, Fabrice Mouhartem, Khoa Nguyen, and Huaxiong Wang. Signature schemes with efficient protocols and dynamic group signatures from lattice assumptions. Cryptology ePrint Archive, Report 2016/101, 2016.

[LPJY13] Benoît Libert, Thomas Peters, Marc Joye, and Moti Yung. Linearly homomorphic structure-preserving signatures and their applications. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 289–307, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

[LTV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Howard J. Karloff and Toniann Pitassi, editors, *44th Annual ACM Symposium on Theory of Computing*, pages 1219–1234, New York, NY, USA, May 19–22, 2012. ACM Press.

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In *Advances in Cryptology - EUROCRYPT 2016*, pages 735–763, 2016.

[PS16] Chris Peikert and Sina Shiehian. Multi-key FHE from LWE, revisited. Cryptology ePrint Archive, Report 2016/196, 2016.

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd Annual ACM Symposium on Theory of Computing*, pages 387–394, Baltimore, Maryland, USA, May 14–16, 1990. ACM Press.