Key-Homomorphic Signatures and Applications to Multiparty Signatures and Non-Interactive Zero-Knowledge

David Derler and Daniel Slamanig

IAIK, Graz University of Technology, Austria
{david.derler,daniel.slamanig}@tugraz.at

Abstract. Key-homomorphic properties of cryptographic objects have proven to be useful, both from a theoretical as well as a practical perspective. Important cryptographic objects such as pseudorandom functions or (public key) encryption have been studied previously with respect to key-homomorphisms. Interestingly, however, signature schemes have not been explicitly investigated in this context so far.

We close this gap and initiate the study of key-homomorphic signatures, which turns out to be an interesting and versatile concept. In doing so, we firstly propose a definitional framework for key-homomorphic signatures distilling various natural flavours of key-homomorphic properties. Those properties aim to generalize larger classes of existing signature schemes, which makes it possible to infer general statements about signature schemes from those classes by simply making black-box use of the respective properties. We then apply our definitional framework to show elegant and simple compilers from classes of schemes admitting different types of key-homomorphisms to a number of other interesting primitives such as ring signature schemes, (universal) designated verifier signature schemes, simulation sound extractable arguments of knowledge, and multisignature schemes. Additionally, using the formalisms provided by our framework, we can prove a tight implication from single-user security to key-prefixed multi-user security for a class of schemes admitting a certain key-homomorphism.

Moreover, we introduce the notion of multikey-homomorphic signatures. Such schemes provide homomorphic properties on the message space of signatures under different keys. We discuss key-homomorphisms in this context, present some first constructive results from key-homomorphic schemes and discuss potential applications.

Keywords. key-homomorphic signatures · ring signatures · (universal) designated verifier signatures · simulation sound extractable argument systems · multisignatures · multi-user signatures · multikey-homomorphic signatures

The authors have been supported by EU H2020 project Prismacloud, grant agreement n°644962.

Table of Contents

1 Introduction

The design of cryptographic schemes that possess certain homomorphic properties on their message space has witnessed significant research within the last years. In the domain of encryption, the first candidate construction of fully homomorphic encryption (FHE) due to Gentry [Gen09] has initiated a fruitful area of research with important applications to computations on (outsourced) encrypted data. In the domain of signatures, the line of work on homomorphic signatures [JMSW02], i.e., signatures that are homomorphic with respect to the message space, has only quite recently attracted attention. Firstly, due to the introduction of computing on authenticated data [ABC^{+}12]. Secondly, due to the growing interest in the application to verifiable delegation of computations (cf. [Cat14] for a quite recent overview), and, finally, due to the recent construction of fully homomorphic signatures [GVW15, BFS14].

In this paper we are interested in another type of homomorphic schemes, so called key-homomorphic schemes. Specifically, we study key-homomorphic signature schemes, that is, signature schemes which are homomorphic with respect to the key space. As we will show in this paper, this concept turns out to be a very interesting and versatile tool.

While we are the first to explicitly study key-homomorphic properties of signatures, some other primitives have already been studied with respect to key-homomorphic properties previously. Applebaum et al. in [AHI11] studied key-homomorphic symmetric encryption schemes in context of related key attacks (RKAs). Recently, Dodis et al. [DMS16] have shown that any such key-homomorphic symmetric encryption schemes implies public key encryption. Rothblum [Rot11] implicitly uses key malleability to construct (weakly) homomorphic public key bit-encryption schemes from private key ones. Goldwasser et al. in [GLW12], and subsequently Tessaro and Wilson in [TW14], use public key encryption schemes with linear homomorphisms over their keys (and some related properties) to construct bounded-collusion identity-based encryption (IBE). Recently, Boneh et al. introduced the most general notion of fully key-homomorphic encryption [BGG^{+}14]. In such a scheme, when given a ciphertext under a public key pk, anyone can translate it into a ciphertext to the same plaintext under public key $(f(\mathsf{pk}), f)$ for any efficiently computable function f.

Another line of work recently initiated by Boneh et al. [BLMR13] is concerned with key-homomorphic pseudorandom functions (PRFs) and pseudo random generators (PRGs). Loosely speaking, a secure PRF family $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$, is key-homomorphic if the keys live in a group $(\mathcal{K}, +)$, and, given two evaluations $F(k_1, x)$ and $F(k_2, x)$ for the same value under two keys, one can efficiently compute $F(k_1 + k_2, x)$. Such PRFs turn out to yield interesting applications such as distributed PRFs, symmetric key proxy re-encryption or updatable encryption. Continuing the work in this direction, alternative constructions [BP14] and extended functionality in the form of constrained key-homomorphic PRFs have been proposed [BFP^{+}15]. We note that the result from Dodis et al. [DMS16], although not mentioned, answers the open question posed by Boneh et al. [BLMR13] "whether key-homomorphic PRFs whose performance is comparable to real-world block ciphers such as AES exist" in a negative way. Finally, Benhamouda et al. use key-homomorphic projective hash functions to construct aggregator oblivious encryption schemes [BJL16] and inner-product functional encryption schemes [BBL17].

When switching to the field of signatures, we can define key-homomorphisms in various different ways, of which we subsequently sketch two to provide a first intuition. One notion is to require that given two signatures for the same message m valid under some pk_1 and pk_2 respectively, one can publicly compute a signature to message m that is valid for a public key pk' that is obtained via some operation on pk_1 and pk_2. Another variant for instance is to require that, given a signature σ to a message m that verifies under pk, σ can be adapted to a signature to m under pk'. Thereby, pk and pk' have a well defined relationship (cf. Section 3 for the details).

Although key-homomorphic signatures have never been discussed or studied explicitly, some implicit use of key-homomorphisms can be found. A recent work by Kiltz et al. [KMP16] introduces a property for canonical identification schemes denoted as random self-reducibility. This basically formalizes the re-randomization of key-pairs as well as adapting parts of transcripts of identification protocols consistently. Earlier, Fischlin and Fleischhacker in [FF13] used re-randomization of key-pairs implicitly in their meta reduction technique against Schnorr signatures. This concept has recently been formalized, yielding the notion of signatures with re-randomizable keys [FKM$^+$16]. In such schemes the $\mathsf{EUF\text{-}CMA}$ security notion is slightly tweaked, by additionally allowing the adversary to see signatures under re-randomized keys. These signatures with re-randomizable keys are then used as basis of an elegant construction of unlinkable sanitizable signatures (cf. [FKM$^+$16]). Allowing the adversary to also access signatures under re-randomized (related) keys, has earlier been studied in context of security of signature schemes against related-key attacks (RKAs) [BCM11, BPT12]. In this context, the goal is to prevent that signature schemes have key-homomorphic properties that allow to adapt signatures under related keys to signatures under the original key (cf. e.g., [MSM$^+$15]).

Concurrent Work. Fiore et al. [FMNP16] in independent and concurrent work introduce the concept of multi-key homomorphic authenticators (MACs and signatures). As this work is only related to what we discuss in Appendix A, we defer a discussion to this section. In another independent work Lai et al. [LTWC16] study different flavours of multi-key homomorphic signatures with homomorphisms on the message and/or key space and show equivalences of different types of such multi-key homomorphic signatures (which are all implied by zk-SNARKS). What they call multi-key key-message-homomorphic signatures can be seen as related to our notion of key-homomorphisms. Yet, our works target totally different directions. Their approach is top-down, i.e., the focus is on introducing new primitives and showing implications between them. In contrast, our approach is bottom-up, i.e., our focus lies on distilling additional properties of larger classes of existing schemes, to (1) obtain new insights regarding generic construction paradigms involving schemes from those classes, and (2) to obtain new instantiations by solely analyzing schemes with respect to their properties.

1.1 Contribution

We initiate the study of key-homomorphic signature schemes. In doing so, we propose various natural definitions of key-homomorphic signatures, generalizing larger classes of existing signature schemes. This generalization makes it possible to infer general statements about signature schemes from those classes by simply making black-box use of the respective properties. Thereby, we rule out certain combinations of key-homomorphism and existing unforgeability notions of signatures. We then employ the formalisms provided by our definitional framework

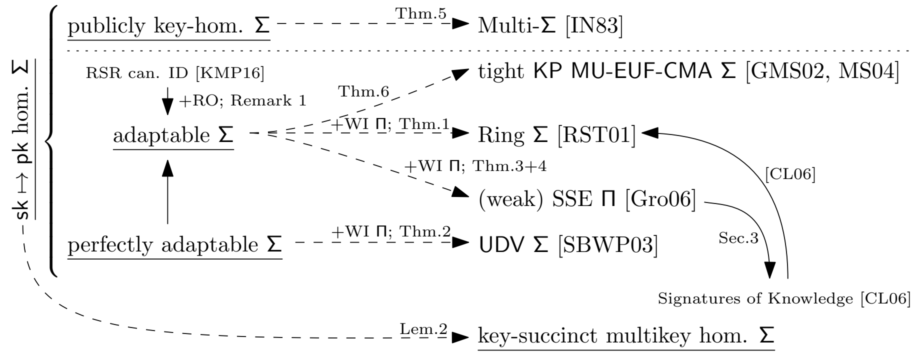

Fig. 1. Overview of contributions and their relations. Legend: underlined. . . variant of key-homomorphism introduced in this paper, ——. . . trivial implication, – – –. . . shown in this paper, sk ↦ pk hom.. . . secret-key to public-key homomorphism, Σ. . . signature scheme, RSR can. ID. . . random-self-reducible canonical identification scheme, RO. . . random oracle, WI. . . witness indistinguishable, Π. . . non interactive argument system, KP. . . key-prefixed, MU. . . multi-user, EUF-CMA. . . existential unforgeability under chosen message attacks, SSE. . . simulation sound extractable, UDV. . . universal designated verifier.

to show various interesting relations and implications. Our results are compactly subsumed in Figure 1. From a theoretical viewpoint our results contribute towards establishing a better understanding of the paradigms which are necessary to construct certain schemes and/or to achieve certain security notions. In particular, we start from very mild security requirements and show how to employ our framework to amplify those to yield relatively strong security guarantees. From a practical viewpoint, our so obtained constructions compare favorably to existing work: they are conceptually extremely easy to understand and therefore less prone to wrong usage. At the same time, our results yield instantiations with no or even reduced overhead when compared to existing work.

More specifically, besides our framework which we see as a contribution on its own, the contributions in this paper are as follows:

Generic Compilers. We present compilers from classes of schemes providing different types of key-homomorphisms to other interesting variants of signature schemes such as ring signatures, (universal) designated verifier signatures or multisignatures. The so obtained constructions, besides being very efficient, are simple and elegant from a construction and security analysis point of view. Basically, for ring signatures, (universal) designated verifier signatures and weakly simulation sound extractable argument systems, one computes a signature using any suitable key-homomorphic scheme under a freshly sampled key and then proves a simple relation over public keys *only*. For simulation sound extractable argument systems we additionally require a strong one-time signature scheme (which, however, also exists under standard assumptions [Gro06]). Multisignatures are directly implied by signatures with certain key-homomorphic properties.

Tight Key-Prefixed Multi-User Security. We prove a theorem which tightly relates the single-user existential unforgeability under chosen message attacks ($\mathsf{EUF\text{-}CMA}$) of a class of schemes admitting a particular key-homomorphism to its key-prefixed multi-user $\mathsf{EUF\text{-}CMA}$ security. This theorem addresses a frequently occurring question in the context of standardization and generalizes existing theorems [Ber15, Lac16] (where such implications are proven for concrete signature schemes) so that it is applicable to a larger class of signature schemes.

(Standard Model & Standard Assumption) Instantiations. We give examples of existing signature schemes admitting types of key-homomorphisms we define. Using our compilers, this directly yields previously unknown instantiations of all variants of signature schemes mentioned above. Most interestingly, we can show that a variant of Waters' signatures in the SXDH setting is perfectly adaptable, which, thus, gives us novel and simple constructions of ring signatures, non-interactive simulation-sound-extractable argument systems, and universal designated verifier signatures without random oracles from standard assumptions (we can use witness-indistinguishable Groth-Sahai [GS08] proofs as argument system). For universal designated verifier signatures, we even obtain the first instantiation from standard assumptions in the standard model. Likewise, our general theorem for multi-user security attests the multi-user security for schemes whose multi-user security was previously unknown. All our instantiations compare favorably to existing constructions regarding conceptual simplicity and come at no or even reduced computational overhead.

Multikey-Homomorphic Signatures. We introduce the notion of multikey-homomorphic signatures. Such schemes provide homomorphic properties on the message space of signatures under different keys. This can be seen as a step towards establishing the signature counterpart of multikey (fully) homomorphic encryption [LTV12, CM15, MW16, PS16a, BP16]. We discuss key-homomorphisms in this context and present some first constructive results from key-homomorphic signatures that yield multikey-homomorphic signatures with a succinct verification key. Finally, we discuss some interesting open problems and highlight that multikey-homomorphic signatures have interesting applications in verifiable delegation of computations. Since we do not see this as the central contribution of this paper, we postpone this section to Appendix A.

2 Preliminaries

We denote algorithms by sans-serif letters, e.g., A, B. If not stated otherwise, all algorithms are required to run in polynomial time and return a special symbol $\perp$ on error. By $y \leftarrow \mathsf{A}(x)$, we denote that y is assigned the output of the potentially probabilistic algorithm A on input x and fresh random coins. Similarly, $y \overset{R}{\leftarrow} S$ means that an element is sampled uniformly at random from a set S and assigned to y, and we use $\mathcal{Q} \overset{\cup}{\leftarrow} z$ as a shorthand for $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{z\}$. We let $[n] := \{1, \ldots, n\}$ and write $\mathsf{Pr}[\Omega : \mathcal{E}]$ to denote the probability of an event $\mathcal{E}$ over the probability space Ω. We use $\mathcal{C}$ to denote challengers of security experiments, and $\mathcal{C}_\kappa$ to make the security parameter explicit. A function $\varepsilon(\cdot) : \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$ is called negligible, iff it vanishes faster than every inverse polynomial, i.e., $\forall\ k : \exists\ n_k : \forall\ n > n_k : \varepsilon(n) < n^{-k}$. We use ρ to denote the success ration of an adversary, i.e., the quotient of its success probability and its running time. Finally, we use $\mathsf{poly}(\cdot)$ to denote a polynomial function.

One-Way Functions. Below, we recall the notion of one-way functions.

Definition 1. *A function $f : \mathsf{Dom}(f) \rightarrow \mathsf{R}(f)$ is called a one-way function, if (1) there exists a PPT algorithm $\mathcal{A}_1$ so that $\forall\ x \in \mathsf{Dom}(f) : \mathcal{A}_1(x) = f(x)$, and if (2) for every PPT algorithm $\mathcal{A}_2$ there is a negligible function $\varepsilon(\cdot)$ such that it holds that*

$$\mathsf{Pr}\left[x \overset{R}{\leftarrow} \mathsf{Dom}(f),\ x^\star \leftarrow \mathcal{A}_2(1^\kappa, f(x))\ :\ f(x) = f(x^\star)\right] \leq \varepsilon(\kappa).$$

Unless stated otherwise, we assume $\mathsf{Dom}(f)$ to be efficiently sampleable.

Signature Schemes. Subsequently, we recall the definition of signature schemes.

Definition 2. *A signature scheme Σ is a triple* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *of PPT algorithms, which are defined as follows:*

$\mathsf{KeyGen}(1^\kappa)$: *This algorithm takes a security parameter κ as input and outputs a secret (signing) key* sk *and a public (verification) key* pk *with associated message space $\mathcal{M}$ (we may omit to make the message space $\mathcal{M}$ explicit).*

$\mathsf{Sign}(\mathsf{sk}, m)$: *This algorithm takes a secret key* sk *and a message $m \in \mathcal{M}$ as input and outputs a signature σ.*

$\mathsf{Verify}(\mathsf{pk}, m, \sigma)$: *This algorithm takes a public key* pk*, a message $m \in \mathcal{M}$ and a signature σ as input and outputs a bit $b \in \{0, 1\}$.*

We note that for a signature scheme many independently generated public keys may be with respect to the same parameters $\textsc{pp}$, e.g., some elliptic curve group parameters. In such a case we introduce an additional algorithm PGen which is run by some (trusted) party to obtain $\textsc{pp} \leftarrow \mathsf{PGen}(1^\kappa)$ and key generation requires PP (which implicitly contain the security parameter) to produce keys as $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\text{PP})$. Moreover, we then assume that PP is included in all public keys.

Besides the usual correctness property, Σ needs to provide some unforgeability notion. Below, we present two standard notions required in our context (ordered from weak to strong). We start with universal unforgeabiltity under no message attacks (UUF-NMA security).

Definition 3 (UUF-NMA). *A signature scheme Σ is* UUF-NMA *secure, if for all PPT adversaries $\mathcal{A}$ there is a negligible function $\varepsilon(\cdot)$ such that*

$$\mathsf{Pr}\left[\begin{array}{l}(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\kappa),\ m^\star \xleftarrow{R} \mathcal{M},\\ \sigma^\star \leftarrow \mathcal{A}(\mathsf{pk}, m^\star)\end{array} : \ \mathsf{Verify}(\mathsf{pk}, m^\star, \sigma^\star) = 1\right] \leq \varepsilon(\kappa).$$

The most common notion is existential unforgeability under adaptively chosen message attacks (EUF-CMA security).

Definition 4 (EUF-CMA). *A signature scheme Σ is* EUF-CMA *secure, if for all PPT adversaries $\mathcal{A}$ there is a negligible function $\varepsilon(\cdot)$ such that*

$$\mathsf{Pr}\left[\begin{array}{l}(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\kappa),\\ (m^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk},\cdot)}(\mathsf{pk})\end{array} : \begin{array}{r}\mathsf{Verify}(\mathsf{pk}, m^\star, \sigma^\star) = 1\ \wedge\\ m^\star \notin \mathcal{Q}^{\mathsf{Sign}}\end{array}\right] \leq \varepsilon(\kappa),$$

where the environment keeps track of the queries to the signing oracle via $\mathcal{Q}^{\mathsf{Sign}}$.

Non-Interactive Proof Systems. Now, we recall a standard definition of non-interactive proof systems Π. Our definitions are inspired by [Gro06]. Therefore, let L_R be an **NP**-language with witness relation R defined as $L_R = \{x \mid \exists\ w : R(x, w) = 1\}$.

Definition 5. *A non-interactive proof system Π is a tuple of algorithms* (Setup, Proof, Verify)*, which are defined as follows:*

$\mathsf{Setup}(1^\kappa)$: *This algorithm takes a security parameter κ as input, and outputs a common reference string* crs.

$\mathsf{Proof}(\mathsf{crs}, x, w)$: *This algorithm takes a common reference string* crs*, a statement x, and a witness w as input, and outputs a proof π.*

$\mathsf{Verify}(\mathsf{crs}, x, \pi)$: *This algorithm takes a common reference string* crs*, a statement x, and a proof π as input, and outputs a bit $b \in \{0, 1\}$.*

Now, we recall formal definitions of the security properties. We thereby relax our definitions to computationally sound proof systems (argument systems).

Definition 6 (Completeness). *A non-interactive proof system Π is* complete, *if for every adversary $\mathcal{A}$ it holds that*

$$\mathsf{Pr}\left[\begin{array}{l}\mathsf{crs} \leftarrow \mathsf{Setup}(1^\kappa),\ (x^\star, w^\star) \leftarrow \mathcal{A}(\mathsf{crs}),\\ \pi \leftarrow \mathsf{Proof}(\mathsf{crs}, x^\star, w^\star)\end{array} : \begin{array}{r}\mathsf{Verify}(\mathsf{crs}, x^\star, \pi) = 1\\ \wedge\ (x^\star, w^\star) \in R\end{array}\right] = 1.$$

Definition 7 (Soundness). *A non-interactive proof system* Π *is* sound*, if for every PPT adversary* $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that*

$$\Pr\left[\mathsf{crs} \leftarrow \mathsf{Setup}(1^\kappa),\ (x^\star, \pi^\star) \leftarrow \mathcal{A}(\mathsf{crs})\ :\ \begin{array}{r}\mathsf{Verify}(\mathsf{crs}, x^\star, \pi^\star) = 1\\ \wedge\ x^\star \notin L_R\end{array}\right] \leq \varepsilon(\kappa).$$

Definition 8 (Adaptive Witness-Indistinguishability). *A non-interactive proof system* Π *is* adaptively witness-indistinguishable*, if for every PPT adversary* $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that*

$$\Pr\left[\mathsf{crs} \leftarrow \mathsf{Setup}(1^\kappa),\ b \xleftarrow{R} \{0,1\},\ b^\star \leftarrow \mathcal{A}^{\mathcal{P}(\mathsf{crs},\cdot,\cdot,\cdot,b)}(\mathsf{crs})\ :\ b = b^\star\right] \leq \varepsilon(\kappa),$$

where $\mathcal{P}(\mathsf{crs}, x, w_0, w_1, b) := \mathsf{Proof}(\mathsf{crs}, x, w_b)$*, and* $\mathcal{P}$ *returns* $\perp$ *if* $(x, w_0) \notin R\ \vee\ (x, w_1) \notin R$*.*

If $\varepsilon = 0$, we have perfect adaptive witness-indistinguishability.

Definition 9 (Adaptive Zero-Knowledge). *A non-interactive proof system* Π *is* adaptively zero-knowledge*, if there exists a PPT simulator* $S = (\mathsf{S}_1, \mathsf{S}_2)$ *such that for every adversary* $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that*

$$\left|\begin{array}{l}\Pr\left[\mathsf{crs} \leftarrow \mathsf{Setup}(1^\kappa)\ :\ \mathcal{A}^{\mathcal{P}(\mathsf{crs},\cdot,\cdot)}(\mathsf{crs}) = 1\right]\ -\\ \Pr\left[(\mathsf{crs}, \tau) \leftarrow \mathsf{S}_1(1^\kappa)\ :\ \mathcal{A}^{\mathcal{S}(\mathsf{crs},\tau,\cdot,\cdot)}(\mathsf{crs}) = 1\right]\end{array}\right| \leq \varepsilon(\kappa),$$

where, τ *denotes a simulation trapdoor. Thereby,* $\mathcal{P}$ *and* $\mathcal{S}$ *return* $\perp$ *if* $(x, w) \notin R$ *or* $\pi \leftarrow \mathsf{Proof}(\mathsf{crs}, x, w)$ *and* $\pi \leftarrow \mathsf{S}_2(\mathsf{crs}, \tau, x)$*, respectively, otherwise.*

If $\varepsilon = 0$, we have perfect adaptive zero-knowledge. It is easy to show that adaptive zero-knowledge implies adaptive witness indistinguishability.

Definition 10 (Proof of Knowledge). *A non-interactive proof system* Π *admits proofs of knowledge, if there exists a PPT extractor* $\mathsf{E} = (\mathsf{E}_1, \mathsf{E}_2)$ *such that for every PPT adversary* $\mathcal{A}$ *there is a negligible function* $\varepsilon_1(\cdot)$ *such that*

$$\left|\begin{array}{l}\Pr\left[\mathsf{crs} \leftarrow \mathsf{Setup}(1^\kappa)\ :\ \mathcal{A}(\mathsf{crs}) = 1\right]\ -\\ \Pr\left[(\mathsf{crs}, \xi) \leftarrow \mathsf{E}_1(1^\kappa)\ :\ \mathcal{A}(\mathsf{crs}) = 1\right]\end{array}\right| \leq \varepsilon_1(\kappa),$$

and for every PPT adversary $\mathcal{A}$ *there is a negligible function* $\varepsilon_2(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l}(\mathsf{crs}, \tau) \leftarrow \mathsf{E}_1(1^\kappa),\ (x^\star, \pi^\star) \leftarrow \mathcal{A}(\mathsf{crs}),\\ w \leftarrow \mathsf{E}_2(\mathsf{crs}, \xi, x^\star, \pi^\star)\end{array}\ :\ \begin{array}{r}\mathsf{Verify}(\mathsf{crs}, x^\star, \pi^\star) = 1\ \wedge\\ (x^\star, w) \notin R\end{array}\right] \leq \varepsilon_2(\kappa).$$

Definition 11 (Simulation Sound Extractability). *An adaptively zero-knowlewdge non-interactive proof system* Π *is simulation sound extractable, if there exists a PPT extractor* $\mathsf{E} = (\mathsf{S}, \mathsf{E})$ *such that for every adversary* $\mathcal{A}$ *it holds that*

$$\left|\begin{array}{l}\Pr\left[(\mathsf{crs}, \tau) \leftarrow \mathsf{S}_1(1^\kappa)\ :\ \mathcal{A}(\mathsf{crs}, \tau) = 1\right]\ -\\ \Pr\left[(\mathsf{crs}, \tau, \xi) \leftarrow \mathsf{S}(1^\kappa)\ :\ \mathcal{A}(\mathsf{crs}, \tau) = 1\right]\end{array}\right| = 0,$$

and for every PPT adversary $\mathcal{A}$ there is a negligible function $\varepsilon_2(\cdot)$ such that

$$\Pr\left[\begin{array}{l}(\mathsf{crs},\tau,\xi)\leftarrow \mathsf{S}(1^\kappa),\\ (x^\star,\pi^\star)\leftarrow \mathcal{A}^{\mathcal{S}(\mathsf{crs},\tau,\cdot)}(\mathsf{crs}),\\ w\leftarrow \mathsf{E}(\mathsf{crs},\xi,x^\star,\pi^\star)\end{array} : \begin{array}{c}\mathsf{Verify}(\mathsf{crs},x^\star,\pi^\star)=1\ \wedge\\ (x^\star,\pi^\star)\notin \mathcal{Q}_{\mathsf{S}}\ \wedge\ (x^\star,w)\notin R\end{array}\right]\le \varepsilon_2(\kappa),$$

where $\mathcal{S}(\mathsf{crs},\tau,x) := \mathsf{S}_2(\mathsf{crs},\tau,x)$ and $\mathcal{Q}_{\mathsf{S}}$ keeps track of the queries to and answers of $\mathcal{S}$.

Definition 12 (Weak Simulation Sound Extractability). *An adaptively zero-knowledge non-interactive proof system Π is weakly simulation sound extractable, if it satisfies Definition 11 with the following modified winning condition:* $\mathsf{Verify}(\mathsf{crs},x^\star,\pi^\star)=1\ \wedge\ (x^\star,\cdot)\notin\mathcal{Q}_{\mathsf{S}}\ \wedge\ (x^\star,w)\notin R$.

Security of Multiparty Signatures. In multiparty signature schemes one often relies on the so called knowledge of secret key (KOSK) assumption within security proofs, where the adversary is required to reveal the secret keys it utilizes to the environment. This is important to prevent rogue-key attacks, i.e., attacks where the adversary constructs public keys based on existing public keys in the system so that it is not required to know the secret key corresponding to the resulting public keys.

To prevent such rogue-key attacks, Ristenpart and Yilek [RY07] introduced and formalized an abstract key-registration concept for multiparty signatures. Any such key-registration protocol is represented as a pair of interactive algorithms $(\mathsf{RegP},\mathsf{RegV})$. A party registering a key runs RegP with inputs public key pk and private key sk. A certifying authority (CA) runs RegV, where the last message is from RegV to RegP and contains either a pk or $\perp$. For instance, in the plain model $\mathsf{RegP}(\mathsf{pk},\mathsf{sk})$ simply sends pk to the CA and RegV on receiving pk simply returns pk. For the KOSK assumption, $\mathsf{RegP}(\mathsf{pk},\mathsf{sk})$ simply sends $(\mathsf{pk},\mathsf{sk})$ to the CA, which checks if $(\mathsf{sk},\mathsf{pk})\in\mathsf{KeyGen}(\textsf{PP})$ and if so replies with pk and $\perp$ otherwise.

To resemble the KOSK assumption in real protocols without revealing the secret key, one can require the adversary to prove knowledge of it's secret key in a way that it can be straight-line extracted by the environment. We require this for all our mulitparty signature schemes in this paper. Yet, we do not make it explicit to avoid complicated models and we simply introduce an RKey oracle that allows the adversary to register key pairs. We stress that our goal is not to study multiparty signatures with respect to real world key-registration procedures, as done in [RY07].

3 Key-Homomorphic Signatures

In this section, we introduce a definitional framework for key-homomorphic signature schemes. In doing so, we propose different natural notions and relate the definitions to previous work that already implicitly used functionality that is related or covered by our definitions.[1]

We focus on signature schemes $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$, where the secret and public key elements live in groups $(\mathbb{H}, +)$ and $(\mathbb{G}, \cdot)$, respectively. We start with the notion of an efficiently computable homomorphism between secret keys and public keys in analogy to the use within IBE in [TW14]. Such a functionality has been implicitly used recently in [FKM+16] to define the notion of signatures with re-randomizable keys.

Definition 13 (Secret Key to Public Key Homomorphism). *A signature scheme Σ provides a secret key to public key homomorphism, if there exists an efficiently computable map $\mu : \mathbb{H} \to \mathbb{G}$ such that for all $\mathsf{sk}, \mathsf{sk}' \in \mathbb{H}$ it holds that $\mu(\mathsf{sk} + \mathsf{sk}') = \mu(\mathsf{sk}) \cdot \mu(\mathsf{sk}')$, and for all $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}$, it holds that $\mathsf{pk} = \mu(\mathsf{sk})$.*

We stress that secret keys and public keys may be vectors containing elements of $\mathbb{H}$ and $\mathbb{G}$ respectively. Then, the operations $+$, $\cdot$ and the map μ are applied componentwise. To keep the definitions compact, we however do not make this explicit.

In the discrete logarithm setting, where we often have $\mathsf{sk} \overset{R}{\leftarrow} \mathbb{Z}_p$ and $\mathsf{pk} = g^{\mathsf{sk}}$ with g being the generator of some prime order p group $\mathbb{G}$, it is obvious that there exists $\mu : \mathsf{sk} \mapsto g^{\mathsf{sk}}$ that is efficiently computable.

Now, we can introduce the first flavour of key-homomorphic signatures, where we focus on the class of functions Φ^+ representing linear shifts and note that one could easily adapt our definition to other suitable classes Φ of functions instead of linear shifts. We stress that we consider Φ as a finite set of functions, all with the same domain and range, and they usually depend on the public key of the signature scheme (which we will not make explicit). Moreover, Φ admits an efficient membership test, is efficiently samplable, and, its functions are efficiently computable. Definition 14 together with the adaptability of signatures (Definition 15) or perfect adaption (Definition 16) are inspired by key-homomorphic encryption schemes [AHI11].

Definition 14 (Φ^+-Key-Homomorphic Signatures). *A signature scheme is called Φ^+-key-homomorphic, if it provides a secret key to public key homomorphism and an additional PPT algorithm* Adapt*, defined as:*

$\mathsf{Adapt}(\mathsf{pk}, m, \sigma, \Delta)$: *Takes a public key pk, a message m, a signature σ, and a function $\Delta \in \Phi^+$ as input, and outputs a public key pk' and a signature σ',*

such that for all $\Delta \in \Phi^+$ and all $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$, all messages m and all $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$ and $(\mathsf{pk}', \sigma') \leftarrow \mathsf{Adapt}(\mathsf{pk}, m, \sigma, \Delta)$ it holds that

$$\Pr[\mathsf{Verify}(\mathsf{pk}', m, \sigma') = 1] = 1 \quad \wedge \quad \mathsf{pk}' = \Delta(\mathsf{pk}).$$

[1] We note that the first parts (up to Definition 15) are slightly more general versions of definitions that we earlier have used in context of redactable signatures [DKS16].

For simplicity we sometimes identify a function $\Delta \in \Phi^+$ with its "shift amount" $\Delta \in \mathbb{H}$.

An interesting property in the context of key-homomorphic signatures is whether adapted signatures look like freshly generated signatures. Therefore, we introduce two different flavours of such a notion, inspired by the context hiding notion for P-homomorphic signatures [ABC⁺12, ALP12] as well as the adaptability notion from [FHS15] for equivalence class signatures [HS14].

Definition 15 (Adaptability of Signatures). *A Φ^+-key-homomorphic signature scheme provides adaptability of signatures, if for every $\kappa \in \mathbb{N}$ and every message m, it holds that $\mathsf{Adapt}(\mathsf{pk}, m, \mathsf{Sign}(\mathsf{sk}, m), \Delta)$ and $(\mathsf{pk} \cdot \mu(\Delta), \mathsf{Sign}(\mathsf{sk} + \Delta, m))$ as well as $(\mathsf{sk}, \mathsf{pk})$ and $(\mathsf{sk}', \mu(\mathsf{sk}'))$ are identically distributed, where $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$, $\mathsf{sk}' \xleftarrow{R} \mathbb{H}$, and $\Delta \xleftarrow{R} \Phi^+$.*

Remark 1. Kiltz et al. [KMP16] have recently used a notion related to Definition 15 (denoted as random self-reducibility) in context of canonical identification schemes. We observe that when turning canonical identification schemes into signature schemes in the random oracle model using the Fiat-Shamir heuristic, every random-self-reducible scheme also satisfies adaptability. However, the contrary is not true as our notion covers a broader class of signature schemes, and in particular including schemes in the standard model (cf. Appendix B).

Thus, the examples of random-self-reducible canonical identification schemes given in [KMP16] directly yield examples of adaptable signatures in the ROM. In Appendix B.1 and B.3 we explicitly show that the Schnorr signatures [Sch91] scheme, as well as a signature scheme due to Katz and Wang [KW03, GJKW07] are adaptable according to the definition above.

An even stronger notion for the indistinguishability of fresh signatures and adapted signatures on the same message is achieved when requiring the distributions to be indistinguishable *even* when the initial signature used in Adapt is known. All schemes that satisfy this stronger notion (stated below) also satisfy Definition 15.

Definition 16 (Perfect Adaption). *A Φ^+-key-homomorphic signature scheme provides perfect adaption, if for every $\kappa \in \mathbb{N}$, every message m, and every signature $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$, it holds that $(\sigma, \mathsf{Adapt}(\mathsf{pk}, m, \sigma, \Delta))$ and $(\sigma, \mathsf{pk} \cdot \mu(\Delta), \mathsf{Sign}(\mathsf{sk} + \Delta, m))$ as well as $(\mathsf{sk}, \mathsf{pk})$ and $(\mathsf{sk}', \mu(\mathsf{sk}'))$ are identically distributed, where $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$, $\mathsf{sk}' \xleftarrow{R} \mathbb{H}$, and $\Delta \xleftarrow{R} \Phi^+$.*

One immediately sees that signatures from random-self-reducible canonical identification schemes, and, thus, Schnorr signatures as well as Katz-Wang signatures, do not satisfy Definition 16 as the commitment sent in the first phase remains fixed. However, we note that there are various existing schemes that satisfy Definition 16. For example, BLS signatures [BLS04], the recent re-randomizable scheme by Pointcheval and Sanders [PS16b], a variant of the well known Waters signatures [Wat05], or the CL signature variant from [CHP12] to name some (cf. Appendix B for a formal treatment of these schemes).

When looking at Definition 14, one could ask whether it is possible to replace Δ in the Adpat algorithm with its public key $\mu(\Delta)$. However, it is easily seen that the existence of such an algorithm contradicts even the weakest security guarantees the underlying signature scheme would need to provide, i.e., universal unforgeability under no-message attacks ($\mathsf{UUF\text{-}NMA}$ security).

Lemma 1. *There cannot be an* $\mathsf{UUF\text{-}NMA}$ *secure* Φ^+*-key-homomorphic signature scheme* Σ *for which there exists a modified PPT algorithm* Adapt' *taking* $\mu(\Delta)$ *instead of* Δ *that still satisfies Definition 14.*

Proof. We prove this by showing that any such scheme implies an adversary against $\mathsf{UUF\text{-}NMA}$ security of Σ. Let us assume that an $\mathsf{UUF\text{-}NMA}$ challenger provides a public key $\mathsf{pk}^\star$ and a target message $m^\star$. Run $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$ being compatible with public key $\mathsf{pk}^\star$, compute $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m^\star)$, then compute $\mathsf{pk}' \leftarrow \mathsf{pk}^\star \cdot \mathsf{pk}^{-1}$ and obtain a forgery $\sigma^\star$ for message $m^\star$ under the target public key $\mathsf{pk}^\star$ by running $(\sigma^\star, \mathsf{pk}^\star) \leftarrow \mathsf{Adapt}(\mathsf{pk}, m^\star, \sigma, \mathsf{pk}')$. $\square$

Now, we move to a definition that covers key-homomorphic signatures where the adaption of a *set of* signatures, each to the same message, to a signature for the same message under a combined public key does not even require the knowledge of the relation between the secret signing keys.

Definition 17 (Publicly Key-Homomorphic Signatures). *A signature scheme is called publicly key-homomorphic, if it provides a secret key to public key homomorphism and an additional PPT algorithm* $\mathsf{Combine}$*, defined as:*

$\mathsf{Combine}((\mathsf{pk}_i)_{i=1}^n, m, (\sigma_i)_{i=1}^n)$: *Takes public keys* $(\mathsf{pk}_i)_{i\in[n]}$*, a message* m*, signatures* $(\sigma_i)_{i\in[n]}$ *as input, and outputs a public key* $\hat{\mathsf{pk}}$ *and a signature* $\hat{\sigma}$*,*

such that for all $n > 1$*, all* $((\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{KeyGen}(1^\kappa))_{i=1}^n$*, all messages* m *and all* $(\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}_i, m))_{i\in[n]}$ *and* $(\hat{\mathsf{pk}}, \hat{\sigma}) \leftarrow \mathsf{Combine}((\mathsf{pk}_i)_{i=1}^n, m, (\sigma_i)_{i=1}^n)$ *it holds that*

$$\hat{\mathsf{pk}} = \prod_{i=1}^{n} \mathsf{pk}_i \quad \wedge \quad \Pr[\mathsf{Verify}(\hat{\mathsf{pk}}, m, \hat{\sigma}) = 1] = 1.$$

Analogously to Definitions 15 and 16, one can define indistinguishability of fresh and combined signatures, but we omit it here as it is straight forward. We want to mention that Definition 17 is, for instance, satisfied by BLS signatures, Waters' signatures with shared Waters' hash parameters (cf. [LOS+06]), as well as the scheme with shared parameters assuming synchronized time in [CHP12] being a variant of the CL signature scheme [CL04] (cf. Appendix B for a more formal treatment of these schemes).

4 Applications

In this section we show how the various key-homomorphic properties defined in the previous section facilitate the black-box construction of ring signatures, universal designated verifier signatures as well as multisignatures.

4.1 Ring Signatures

Ring signature schemes [RST01] allow a member of an ad-hoc group $\mathcal{R}$ (the so called ring), defined by the member's public verification keys, to anonymously sign a message on behalf of $\mathcal{R}$. Given a ring signature and all public keys for $\mathcal{R}$, one can verify the validity of such a signature with respect to $\mathcal{R}$, but it is infeasible to identify the actual signer, i.e., the signer is unconditionally anonymous. Due to this anonymity feature ring signatures have proven to be an interesting tool for numerous applications, most notable for whistleblowing. The two main lines of work in the design of ring signatures target reducing the signature size or removing the requirement for random oracles (e.g., [DKNS04, CGS07, GK15]). We provide a construction that does not require random oracles and has linear signature size. It provides an alternative very simple generic framework to construct ring signatures in addition to existing ones (cf. [BKM09, BK10]). In Appendix B, we provide various suitable candidate schemes to obtain novel instantiations.

Subsequently, we formally define ring signature schemes (adopting [BKM09]) and note that the model implicitly assumes knowledge of secret keys [RY07] as discussed in Section 2.

Definition 18. *A ring signature scheme* RS *is a tuple* $\mathsf{RS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *of PPT algorithms, which are defined as follows.*

$\mathsf{Setup}(1^\kappa)$: *This algorithm takes as input a security parameter* κ *and outputs public parameters* PP.

$\mathsf{Gen}(\mathsf{PP})$: *This algorithm takes as input the public parameter* PP *and outputs a keypair* $(\mathsf{sk}, \mathsf{pk})$.

$\mathsf{Sign}(\mathsf{PP}, \mathsf{sk}_i, m, \mathcal{R})$: *This algorithm takes as input the public parameters* PP*, a secret key* sk_i*, a message* $m \in \mathcal{M}$ *and a ring* $\mathcal{R} = (\mathsf{pk}_j)_{j\in[n]}$ *of* n *public keys such that* $\mathsf{pk}_i \in \mathcal{R}$*. It outputs a signature* σ.

$\mathsf{Verify}(\mathsf{PP}, m, \sigma, \mathcal{R})$: *This algorithm takes as input the public parameters* PP*, a message* $m \in \mathcal{M}$*, a signature* σ *and a ring* $\mathcal{R}$*. It outputs a bit* $b \in \{0, 1\}$.

A secure ring signature scheme needs to be correct, unforgeable, and anonymous. While we omit the obvious correctness definition, we subsequently provide formal definitions for the remaining properties following [BKM09]. We note that Bender et al. in [BKM09] have formalized multiple variants of these properties, where we always use the strongest one.

Unforgeability requires that without any secret key sk_i that corresponds to a public key $\mathsf{pk}_i \in \mathcal{R}$, it is infeasible to produce valid signatures with respect to arbitrary such rings $\mathcal{R}$.

Definition 19 (Unforgeability). *A ring signature scheme provides unforgeability, if for all PPT adversaries* $\mathcal{A}$*, there exists a negligible function* $\varepsilon(\cdot)$ *such that it holds that*

$$\Pr\left[\begin{array}{l} \{(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\kappa)\}_{i\in[\mathsf{poly}(\kappa)]}, \\ \mathcal{O} \leftarrow \{\mathsf{Sig}(\cdot,\cdot,\cdot), \mathsf{Key}(\cdot)\}, \\ (m^\star, \sigma^\star, \mathcal{R}^\star) \leftarrow \mathcal{A}^{\mathcal{O}}(\{\mathsf{pk}_i\}_{i\in[\mathsf{poly}(\kappa)]}) \end{array} : \begin{array}{r} \mathsf{Verify}(m^\star, \sigma^\star, \mathcal{R}^\star) = 1 \;\wedge \\ (\cdot, m^\star, \mathcal{R}^\star) \notin \mathcal{Q}^{\mathsf{Sign}} \;\wedge \\ \mathcal{R}^\star \subseteq \{\mathsf{pk}_i\}_{i\in[\mathsf{poly}(\kappa)]\setminus\mathcal{Q}^{\mathsf{Key}}} \end{array}\right] \leq \varepsilon(\kappa),$$

where $\mathsf{Sig}(i, m, \mathcal{R}) := \mathsf{Sign}(\mathsf{sk}_i, m, \mathcal{R})$, Sig *returns* $\perp$ *if* $\mathsf{pk}_i \notin \mathcal{R} \ \vee \ i \notin [\mathsf{poly}(\kappa)]$, *and* $\mathcal{Q}^{\mathsf{Sig}}$ *records the queries to* Sig. *Furthermore,* $\mathsf{Key}(i)$ *returns* sk_i *and* $\mathcal{Q}^{\mathsf{Key}}$ *records the queries to* Key.

Anonymity requires that it is infeasible to tell which ring member produced a certain signature as long as there are at least two honest members in the ring.

Definition 20 (Anonymity). *A ring signature scheme provides anonymity, if for all PPT adversaries $\mathcal{A}$ and for all polynomials $n(\cdot)$, there exists a negligible function $\varepsilon(\cdot)$ such that it holds that*

$$\Pr\left[\begin{array}{l} \{(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{Gen}(1^\kappa)\}_{i\in[\mathsf{poly}(\kappa)]}, \\ b \xleftarrow{R} \{0,1\},\ \mathcal{O} \leftarrow \{\mathsf{Sig}(\cdot,\cdot,\cdot)\}, \\ (m, j_0, j_1, \mathcal{R}, \mathsf{st}) \leftarrow \mathcal{A}^{\mathcal{O}}(\{\mathsf{pk}_i\}_{i\in[\mathsf{poly}(\kappa)]}), \\ \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_{j_b}, m, \mathcal{R}), \\ b^\star \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{st}, \sigma, \{\mathsf{sk}_i\}_{i\in[\mathsf{poly}(\kappa)]\setminus j_0}) \end{array} \ : \ \begin{array}{r} b = b^\star \ \wedge \\ \{\mathsf{pk}_{j_0}, \mathsf{pk}_{j_1}\} \subseteq \mathcal{R} \end{array}\right] \leq 1/2 + \varepsilon(\kappa),$$

where $\mathsf{Sig}(i, m, \mathcal{R}) := \mathsf{Sign}(\mathsf{sk}_i, m, \mathcal{R})$.

Our Construction. In Scheme 1 we present our black-box construction of ring signatures from any Φ^+-key-homomorphic EUF-CMA secure signature scheme Σ with adaptable signatures and any witness indistinguishable argument system Π that admits proofs of knowledge. The idea behind the scheme is as follows. A ring signature for message m with respect to ring $\mathcal{R}$ consists of a signature for $m||\mathcal{R}$ using Σ with a randomly generated key pair together with a proof of knowledge attesting the knowledge of the "shift amount" from the random public key to (at least) one of the public keys in $\mathcal{R}$.[2] Very briefly, unforgeability then holds because—given a valid ring signature—one can always extract a valid signature of one of the ring members. Anonymity holds because the witness indistinguishability of the argument system guarantees that signatures of different ring members are indistinguishable.

Upon signing, we need to prove knowledge of a witness for the following **NP** relation R.

$$((\mathsf{pk}, \mathsf{cpk}, \mathcal{R}), \mathsf{sk}') \in R \iff \exists\ \mathsf{pk}_i \in \mathcal{R} \cup \{\mathsf{cpk}\} \ : \ \mathsf{pk}_i = \mathsf{pk} \cdot \mu(\mathsf{sk}')$$

For the sake of compactness, we assume that the relation is implicitly defined by the scheme. One can obtain a straight forward instantiation by means of disjunctive proofs of knowledge [CDS94] (similar as it is done in many known constructions). Therefore one could use the following **NP** relation R.

$$((\mathsf{pk}, \mathsf{cpk}, \mathcal{R}), \mathsf{sk}') \in R \iff \left(\vee_{\mathsf{pk}_i \in \mathcal{R}} \ \mathsf{pk}_i = \mathsf{pk} \cdot \mu(\mathsf{sk}')\right) \ \vee \ \mathsf{cpk} = \mathsf{pk} \cdot \mu(\mathsf{sk}')$$

Using this approach, however, yields signatures of linear size. To reduce the signature size, one could, e.g., follow the approach of [DKNS04].

Theorem 1. *If Σ is correct,* EUF-CMA *secure, and provides adaptability of signatures, Π is complete and witness indistinguishable and admits proofs of knowledge, then Scheme 1 is correct, unforgeable, and anonymous.*

We prove the theorem above in Appendix C.1.

[2] For technical reasons we need an additional public key cpk in the public parameters.

$\mathsf{Setup}(1^\kappa)$: Run $\mathsf{crs} \leftarrow \Pi.\mathsf{Setup}(1^\kappa)$, $(\mathsf{csk}, \mathsf{cpk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$, set $\textsc{pp} \leftarrow (1^\kappa, \mathsf{crs}, \mathsf{cpk})$ and return $\textsc{pp}$.

$\mathsf{Gen}(\textsc{pp})$: Run $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa)$ and return $(\mathsf{sk}_i, \mathsf{pk}_i)$.

$\mathsf{Sign}(\textsc{pp}, \mathsf{sk}_i, m, \mathcal{R})$: Parse $\textsc{pp}$ as $(1^\kappa, \mathsf{crs}, \mathsf{cpk})$ and return $\perp$ if $\mu(\mathsf{sk}_i) \notin \mathcal{R}$. Otherwise, return $\sigma \leftarrow (\delta, \mathsf{pk}, \pi)$, where

$$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\kappa),\ \ \delta \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}, m||\mathcal{R}),\ \text{and}$$
$$\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathsf{pk}, \mathsf{cpk}, \mathcal{R}), (\mathsf{sk}_i - \mathsf{sk})).$$

$\mathsf{Verify}(\textsc{pp}, m, \sigma, \mathcal{R})$: Parse $\textsc{pp}$ as $(1^\kappa, \mathsf{crs}, \mathsf{cpk})$ and σ as $(\delta, \mathsf{pk}, \pi)$ and return 1 if the following holds, and 0 otherwise:

$$\Sigma.\mathsf{Verify}(\mathsf{pk}, m||\mathcal{R}, \delta) = 1 \quad \wedge \quad \Pi.\mathsf{Verify}(\mathsf{crs}, (\mathsf{pk}, \mathsf{cpk}, \mathcal{R}), \pi) = 1.$$

Scheme 1: Black-Box Construction of Ring Signatures

4.2 Universal Designated Verifier Signatures

In designated verifier signatures [JSI96] a signer chooses a designated verifier upon signing a message and, given this signature, only the designated verifier is convinced of its authenticity. The idea behind those constructions is to ensure that the designated verifier can "fake" signatures which are indistinguishable from signatures of the original signer. Universal designated verifier signatures (UDVS) [SBWP03] further extend this concept by introducing an additional party, which performs the designation process by converting a conventional signature to a designated-verifier one. There exists quite a lot of work on UDVS, and, most notably, in [SS08] it was shown how to convert a large class of signature schemes to UDVS. Their approach can be seen as related to our approach, yet they do not rely on key-homomorphisms and they only achieve weaker security guarantees.[3]

While one can interpret designated verifier signatures as a special case of ring signatures where $|\mathcal{R}| = 2$, i.e., the ring is composed of the public keys of signer and designated verifier (as noted in [RST01, BKM09]), there seems to be no obvious black-box relation turning ring signatures into UDVS. Mainly, since UDVS require the functionality to convert standard signatures to designated verifier ones.[4]

To this end, we explicitly treat constructions of UDVS from key-homomorphic signatures subsequently. We start by recalling the security model from [SBWP03]

[3] We also note that [SS08] informally mention that their approach is also useful to construct what they call hierarchical ring signatures. However their paradigm is not useful to construct ring signatures as we did in the previous section.

[4] We, however, note that an extension of the UDVS model to universal designated verifier *ring* signatures would be straight forward and also our scheme would be straight forwardly extensible using the same techniques as in Scheme 1.

including some notational adaptations and a strengthened version of the DV-unforgeability notion which we introduce here.

Definition 21. *A universal designated verifier signature scheme* UDVS *builds up on a conventional signature scheme* $\Sigma = (\mathsf{PGen}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *and additionally provides the PPT algorithms* $(\mathsf{DVGen}, \mathsf{Desig}, \mathsf{Sim}, \mathsf{DVerify})$*, which are defined as follows.*

$\mathsf{DVGen}(\mathsf{PP})$: *This algorithm takes the public parameters* PP *as input and generates and outputs a designated-verifier key pair* $(\mathsf{vsk}, \mathsf{vpk})$*.*

$\mathsf{Desig}(\mathsf{pk}, \mathsf{vpk}, m, \sigma)$: *This algorithm takes a signer public key* pk*, a designated-verifier public key* vpk*, a message* m*, and a valid signature* σ *as input, and outputs a designated-verifier signature* δ*.*

$\mathsf{Sim}(\mathsf{pk}, \mathsf{vsk}, m)$: *This algorithm takes a signer public key* pk*, a designated-verifier secret key* vsk*, and a message* m *as input, and outputs a designated-verifier signature* δ*.*

$\mathsf{DVerify}(\mathsf{pk}, \mathsf{vsk}, m, \delta)$: *This algorithm takes a signer public key* pk*, a designated-verifier secret key* vsk*, a message* m*, and a designated-verifier signature* δ *as input, and outputs a bit* $b \in \{0, 1\}$*.*

Subsequently we formally recall the security properties, where we omit the obvious correctness notion. For the remaining notions we largely follow [SBWP03, SS08].

DV-unforgeability captures the intuition that it should be infeasible to come up with valid designated verifier signatures where no corresponding original signature exists. Subsequently, we introduce a stronger variant of DV-unforgeability, which we term *simulation-sound DV-unforgeability.* This notion additionally provides the adversary with an oracle to simulate designated-verifier signatures on other messages for the targeted designated verifier. It is easy to see that our notion implies DV-unforgeability in the sense of [SBWP03].

Definition 22 (Simulation-Sound DV-Unforgeability). *An* UDVS *provides simulation-sound DV-unforgeability, if for all PPT adversaries* $\mathcal{A}$*, there exists a negligible function* $\varepsilon(\cdot)$ *such that it holds that*

$$\Pr\left[\begin{array}{l} \mathsf{PP} \leftarrow \mathsf{PGen}(1^\kappa), \\ (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{PP}), \\ (\mathsf{vsk}, \mathsf{vpk}) \leftarrow \mathsf{DVGen}(\mathsf{PP}), \\ \mathcal{O} \leftarrow \{\mathsf{Sig}(\mathsf{sk}, \cdot),\ \mathsf{Vrfy}(\mathsf{pk}, \mathsf{vsk}, \cdot, \cdot), \\ \mathsf{S}(\mathsf{pk}, \mathsf{vsk}, \cdot)\}, \\ (m^\star, \delta^\star) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pk}, \mathsf{vpk}) \end{array} : \begin{array}{r} \mathsf{DVerify}(\mathsf{pk}, \mathsf{vsk}, m^\star, \delta^\star) = 1\ \wedge \\ m^\star \notin \mathcal{Q}^{\mathsf{Sig}}\ \wedge\ m^\star \notin \mathcal{Q}^{\mathsf{Sim}} \end{array}\right] \leq \varepsilon(\kappa),$$

where $\mathsf{Sig}(\mathsf{sk}, m) := \mathsf{Sign}(\mathsf{sk}, m)$*,* $\mathsf{Vrfy}(\mathsf{pk}, \mathsf{vsk}, m, \delta) := \mathsf{DVerify}(\mathsf{pk}, \mathsf{vsk}, m, \delta)$*, and* $\mathsf{S}(\mathsf{pk}, \mathsf{vsk}, m) := \mathsf{Sim}(\mathsf{pk}, \mathsf{vsk}, m)$*. Furthermore, the environment keeps tracks of the messages queried to* Sig *and* S *via* $\mathcal{Q}^{\mathsf{Sig}}$ *and* $\mathcal{Q}^{\mathsf{Sim}}$*, respectively.*

Non-transferability privacy models the requirement that the designated verifier can simulate signatures which are indistinguishable from honestly designated signatures.

Definition 23 (Non-Transferability Privacy). *An* UDVS *provides non-transferability privacy, if for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\varepsilon(\cdot)$ such that it holds that*

$$\Pr\left[\begin{array}{l} \mathsf{PP} \leftarrow \mathsf{PGen}(1^\kappa),\ (\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{PP}), \\ b \xleftarrow{R} \{0,1\},\ \mathcal{O} \leftarrow \{\mathsf{Sig}(\mathsf{sk},\cdot), \mathsf{RKey}(\cdot,\cdot,\cdot)\}, \\ (m^\star, \mathsf{st}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pk}),\ \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m^\star), \\ b^\star \leftarrow \mathcal{A}^{\mathcal{O} \cup \{\mathsf{SoD}(\mathsf{pk},\cdot,m^\star,\sigma,b)\}}(\mathsf{st}) \end{array} : \begin{array}{r} b = b^\star\ \wedge \\ m^\star \notin \mathcal{Q}^{\mathsf{Sig}} \end{array}\right] \leq 1/2 + \varepsilon(\kappa),$$

where the oracles are defined as follows:

$\mathsf{Sig}(\mathsf{sk}, m)$: *This oracle computes $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$ and returns σ.*

$\mathsf{RKey}(i, \mathsf{vsk}, \mathsf{vpk})$: *This oracle checks whether $\mathtt{DVK}[i] \neq \perp$ and returns $\perp$ if so. Otherwise, it checks whether* (vsk, vpk) *is a valid output of* DVGen *and sets $\mathtt{DVK}[i] \leftarrow (\mathsf{vsk}, \mathsf{vpk})$ if so.*

$\mathsf{SoD}(\mathsf{pk}, i, m, \sigma, b)$: *This oracle obtains $(\mathsf{vsk}, \mathsf{vpk}) \leftarrow \mathtt{DVK}[i]$ and returns $\perp$ if no entry for i exists. Then, if $b = 0$, it computes $\delta \leftarrow \mathsf{Sim}(\mathsf{pk}, \mathsf{vsk}, m)$, and, if $b = 1$ it computes $\delta \leftarrow \mathsf{Desig}(\mathsf{pk}, \mathsf{vpk}, m, \sigma)$. In the end it returns δ. This oracle can only be called once.*

Further, the environment maintains a list $\mathcal{Q}^{\mathsf{Sig}}$ keeping track of the Sig *queries.*

The notion above captures non-transferability privacy in the sense of [SS08]. This notion can be strengthened to what we call *strong non-transferability privacy* which allows multiple calls to SoD (as in [SBWP03]). While non-transferability privacy is often sufficient in practice, we will prove that our construction provides strong non-transferability privacy (clearly implying non-transferability privacy) to obtain the most general result.

Our Construction. In Scheme 2, we present our construction of UDVS from any Φ^+-key-homomorphic EUF-CMA secure Σ with perfect adaption of signatures, any witness indistinguishable argument system Π that admits proofs of knowledge, and any one-way function f. Our construction uses the "OR-trick" [JSI96], known from DVS.[5] Upon computing designations and simulations of designated-verifier signatures, we require to prove knowledge of witnesses for the following **NP** relation R:

$$((\mathsf{pk}, \mathsf{vpk}), (\mathsf{sk}, \mathsf{vsk})) \in R \iff \mathsf{pk} = \mu(\mathsf{sk}) \ \vee\ \mathsf{vpk} = f(\mathsf{vsk}).$$

For brevity we assume that the parameters PP generated upon setup are implicit in every pk and vpk generated by Gen and DVGen respectively. Furthermore, we assume that R is implicitly defined by the scheme.

Theorem 2. *If Σ is* EUF-CMA *secure and perfectly adapts signatures, f is a one-way function, and Π is witness indistinguishable and admits proofs of knowledge, then Scheme 2 is correct, simulation-sound DV-unforgeable, and provides strong non-transferability privacy.*

[5] We note that our construction is inspired by earlier work of us on a variant of redactable signatures [DKS16].

$\mathsf{PGen}(1^\kappa)$: Run $\text{PP}' \leftarrow \Sigma.\mathsf{PGen}(1^\kappa)$, $\mathsf{crs} \leftarrow \Pi.\mathsf{Setup}(1^\kappa)$, and return $\text{PP} \leftarrow (\text{PP}', \mathsf{crs})$.

$\mathsf{DVGen}(\text{PP})$: Run $\mathsf{vsk} \xleftarrow{R} \mathsf{Dom}(f)$, set $\mathsf{vpk} \leftarrow f(\mathsf{vsk})$ and return $(\mathsf{vsk}, \mathsf{vpk})$.

$\mathsf{Desig}(\mathsf{pk}, \mathsf{vpk}, m, \sigma)$: Output $\delta \leftarrow (\mathsf{pk}', \sigma_\mathsf{R}, \pi)$, where

$$(\mathsf{sk}', \mathsf{pk}') \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa),\ (\mathsf{pk}_\mathsf{R}, \sigma_\mathsf{R}) \leftarrow \Sigma.\mathsf{Adapt}(\mathsf{pk}, m, \sigma, \mathsf{sk}'),$$
$$\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathsf{pk}', \mathsf{vpk}), (\mathsf{sk}', \bot)).$$

$\mathsf{Sim}(\mathsf{pk}, \mathsf{vsk}, m)$: Output $\delta \leftarrow (\mathsf{pk}', \sigma_\mathsf{R}, \pi)$, where

$$(\mathsf{sk}_\mathsf{R}, \mathsf{pk}_\mathsf{R}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa),\ \mathsf{pk}' \leftarrow \mathsf{pk}_\mathsf{R} \cdot \mathsf{pk}^{-1},\ \sigma_\mathsf{R} \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}_\mathsf{R}, m),$$
$$\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathsf{pk}', f(\mathsf{vsk})), (\bot, \mathsf{vsk})).$$

$\mathsf{DVerify}(\mathsf{pk}, \mathsf{vsk}, m, \delta)$: Parse δ as $(\mathsf{pk}', \sigma_\mathsf{R}, \pi)$ and return 1 if the following holds, and 0 otherwise:

$$\Sigma.\mathsf{Verify}(\mathsf{pk} \cdot \mathsf{pk}', m, \sigma_\mathsf{R}) = 1 \quad \wedge \quad \Pi.\mathsf{Verify}(\mathsf{crs}, (\mathsf{pk}', f(\mathsf{vsk})), \pi) = 1.$$

Scheme 2: Black-Box Construction of UDVS

We prove the theorem above in Appendix C.2. We note that if non-transferability privacy is sufficient, Σ only needs to be adaptable. Then, besides the candidate schemes presented in Appendix B, one can, e.g., also instantiate Scheme 2 with the very efficient Schnorr signature scheme.

4.3 Simulation Sound Extractable Argument Systems

The constructions in the previous sections implicitly use techniques to ensure that even though we have to simulate proofs within our security reduction, we can still extract the required witness for the forgery. In this section we isolate the essence of this techniques and show that they are generally applicable to extend witness indistinguishable argument systems admitting proofs of knowledge to (weak) simulation sound extractable argument systems using EUF-CMA secure signature schemes that adapt signatures. This makes our techniques useful in a broader range of applications.

For the stronger variant of simulation sound extractability we additionally require strong one-time signatures. We start by defining such schemes. Then we proceed in showing that for weak simulation sound extractability, we do not even require stong one-time signatures.

Definition 24 (Strong One-Time Signature Scheme). *A strong one-time signature scheme Σ_ot provides the same interface as a conventional signature scheme Σ and satisfies the following unforgeability notion: For all PPT adversaries $\mathcal{A}$ there is a negligible function $\varepsilon(\cdot)$ such that*

$$\Pr\left[\begin{matrix}(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\kappa), \\ (m^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}, \cdot)}(\mathsf{pk})\end{matrix} \ : \ \begin{matrix}\mathsf{Verify}(\mathsf{pk}, m^\star, \sigma^\star) = 1 \wedge \\ (m^\star, \sigma^\star) \notin \mathcal{Q}^{\mathsf{Sign}}\end{matrix}\right] \leq \varepsilon(\kappa),$$

where the oracle $\mathsf{Sign}(\mathsf{sk},\mathsf{m}) \coloneqq \Sigma.\mathsf{Sign}(\mathsf{sk},\mathsf{m})$ *can only be called once.*

An efficient example of a strong one-time signature scheme can be found in [Gro06]. For our construction, first let L be an arbitrary **NP**-language $L = \{x \mid \exists\ w : R(x,w) = 1\}$, for which we aim to construct a simulation sound extractable argument system, and let L' be defined as follows:

$$((x,\mathsf{cpk},\mathsf{pk}),(w,\mathsf{csk}-\mathsf{sk})) \in L' \iff (x,w) \in R \ \vee\ \mathsf{cpk} = \mathsf{pk}\cdot\mu(\mathsf{csk}-\mathsf{sk}).$$

In Scheme 3 we present our construction of a simulation sound extractable argument system Π_{sse} for L. Our technique is inspired by [GMY03, GMY06, Gro06] but conceptually simpler. This is mainly due to the fact that the adaptability of the used signature scheme allows us to get rid of the encryption scheme, and, consequently, also the requirement to prove statements about encrypted values.

Essentially, the intuition of our construction is the following. We use a combination of an adaptable EUF-CMA secure signature scheme Σ and a strong one-time signature scheme Σ_{ot} to add the required non-malleability guarantees to the underlying argument system.[6] Upon each proof computation, we use Σ to "certify" the public key of a newly generated key pair of Σ_{ot}. The associated secret key of Σ_{ot} is then used to sign the parts of the proof which must be non-malleable. Adaptability of Σ makes it possible to also use newly generated keys of Σ upon each proof computation. In particular, the relation associated to L' is designed so that the second clause in the OR statement is the "shift amount" required to shift such signatures to signatures under a key cpk in the crs. A proof for $x \in L$ is easy to compute when given w such that $(x,w) \in R$. One does not need a satisfying assignment for the second clause in the OR statement, and can thus compute all signatures under newly generated keys. To simulate proofs, however, we can set up crs in a way that we know csk corresponding to cpk, compute the "shift amount" and use it as a satisfying witness for the second clause in the OR statement. Under this strategy, the witness indistinguishability of the underlying argument system for L', the crs indistinguishability provided by the proof of knowledge property, and the secret-key to public-key homomorphism of Σ guarantees the zero-knowledge property of our argument system for L.

What remains is to argue that we can use the extractor of the underlying argument system for L' as an extractor for L in the simulation sound extractability setting. In fact, under the strategy we use, we never have to simulate proofs for statements outside L' which is sufficient for the extractor for L' to work with overwhelming probability. Furthermore, we can show that the probability to extract a valid witness for the second clause in the OR statement is negligible, as this either yields a forgery with respect to Σ_{ot} under some $\mathsf{pk}_{\mathsf{ot}}$ previously obtained from the simulator (if the adversary modified any of the non-malleable parts of a proof previously obtained via the simulator) or for Σ under cpk (if $\mathsf{pk}_{\mathsf{ot}}$ has never been certified). Now we know, however, that the extractor for L' works with overwhelming probability by definition, which means that we will extract a satisfying witness for $x \in L$ with overwhelming probability.

[6] Σ_{ot} is only required as the signatures produced by Σ may be malleable on their own.

$\mathsf{Setup}(1^\kappa)$: Run $\mathsf{crs}_\Pi \leftarrow \Pi.\mathsf{Setup}(1^\kappa)$, $(\mathsf{csk}, \mathsf{cpk}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa)$ and return $\mathsf{crs} \leftarrow (\mathsf{crs}_\Pi, \mathsf{cpk})$.

$\mathsf{Proof}(\mathsf{crs}, x, w)$: Run $(\mathsf{sk}, \mathsf{pk}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa)$, $(\mathsf{sk}_\mathsf{ot}, \mathsf{pk}_\mathsf{ot}) \leftarrow \Sigma_\mathsf{ot}.\mathsf{KeyGen}(1^\kappa)$, and return $\pi \leftarrow (\pi_\Pi, \mathsf{pk}, \sigma, \mathsf{pk}_\mathsf{ot}, \sigma_\mathsf{ot})$, where

$$\pi_\Pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (x, \mathsf{cpk}, \mathsf{pk}), (w, \perp)),\ \sigma \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}, \mathsf{pk}_\mathsf{ot}),\ \text{and}$$
$$\sigma_\mathsf{ot} \leftarrow \Sigma_\mathsf{ot}.\mathsf{Sign}(\mathsf{sk}_\mathsf{ot}, \pi_\Pi || x || \mathsf{pk} || \sigma).$$

$\mathsf{Verify}(\mathsf{crs}, x, \pi)$: Parse π as $(\pi_\Pi, \mathsf{pk}, \sigma, \mathsf{pk}_\mathsf{ot}, \sigma_\mathsf{ot})$ and return 1 if the following holds and 0 otherwise:

$$\Pi.\mathsf{Verify}(\mathsf{crs}, (x, \mathsf{cpk}, \mathsf{pk}), \pi_\Pi) = 1 \ \wedge\ \Sigma.\mathsf{Verify}(\mathsf{pk}, \mathsf{pk}_\mathsf{ot}) = 1 \ \wedge$$
$$\Sigma_\mathsf{ot}.\mathsf{Verify}(\mathsf{pk}_\mathsf{ot}, \pi_\Pi || x || \mathsf{pk} || \sigma) = 1.$$

$\mathsf{S}_1(1^\kappa)$: Run $(\mathsf{crs}_\Pi, \perp) \leftarrow \Pi.\mathsf{E}_1(1^\kappa)$, $(\mathsf{csk}, \mathsf{cpk}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa)$ and return (crs, τ), where

$$\mathsf{crs} \leftarrow (\mathsf{crs}_\Pi, \mathsf{cpk}) \text{ and } \tau \leftarrow \mathsf{csk}.$$

$\mathsf{S}_2(\mathsf{crs}, \tau, x)$: Parse τ as csk, run $(\mathsf{sk}, \mathsf{pk}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa)$, $(\mathsf{sk}_\mathsf{ot}, \mathsf{pk}_\mathsf{ot}) \leftarrow \Sigma_\mathsf{ot}.\mathsf{KeyGen}(1^\kappa)$, and return $\pi \leftarrow (\pi_\Pi, \mathsf{pk}, \sigma, \mathsf{pk}_\mathsf{ot}, \sigma_\mathsf{ot})$, where

$$\pi_\Pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (x, \mathsf{cpk}, \mathsf{pk}), (\perp, \mathsf{csk} - \mathsf{sk})),\ \sigma \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}, \mathsf{pk}_\mathsf{ot}),\ \text{and}$$
$$\sigma_\mathsf{ot} \leftarrow \Sigma_\mathsf{ot}.\mathsf{Sign}(\mathsf{sk}_\mathsf{ot}, \pi_\Pi || x || \mathsf{pk} || \sigma).$$

$\mathsf{S}(1^\kappa)$: Run $(\mathsf{crs}_\Pi, \xi) \leftarrow \Pi.\mathsf{E}_1(1^\kappa)$, $(\mathsf{csk}, \mathsf{cpk}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa)$ and return $(\mathsf{crs}, \tau, \xi)$, where

$$\mathsf{crs} \leftarrow (\mathsf{crs}_\Pi, \mathsf{cpk}) \text{ and } \tau \leftarrow \mathsf{csk}.$$

$\mathsf{E}(\mathsf{crs}, \xi, x, \pi)$: Run $(w, \perp) \leftarrow \Pi.\mathsf{E}_2(\mathsf{crs}, \xi, x, \pi)$ and return w.

Scheme 3: Simulation Sound Extractable Argument System Π_sse.

Theorem 3. *Let Π be a complete, witness indistinguishable non-interactive argument system that admits proofs of knowledges for the language L', let Σ be an* EUF-CMA *secure signature scheme that adapts signatures, and let Σ_ot be a strong one-time signature scheme, then the argument system Π_sse is a complete, simulation sound extractable argument system for language L.*

We prove the theorem in Appendix C.3.

Weak Simulation Sound Extractability. If one allows the proofs to be malleable and only requires non-malleability with respect to the statements one can omit the strong one-time signature scheme and directly sign $\pi_\Pi || x || \mathsf{pk}$ using Σ. We refer to this modified argument system as Π_wsse.

Theorem 4. *Let Π be a complete, witness indistinguishable non-interactive argument system that admits proofs of knowledges for the language L', and let Σ be an* EUF-CMA *secure signature scheme that adapts signatures, then the argument system Π_wsse is a complete, weakly simulation sound extractable argument system for language L.*

Proof (Sketch). The proof for the theorem above is exactly the same as the one for simulation sound extractability in Appendix C.3, except that we do not need to engage with challengers for the one-time signature scheme (i.e., in Game 2 nothing is changed) and $\mathsf{Pr}[F_2]$ is exactly the same as extracting a forgery for Σ in the transition between Game 3 and Game 4. $\square$

Signatures of Knowledge. Also note that using our techniques in a non-black box way directly yields signatures of knowledge [CL06]. That is, a signature of knowledge on a message m with respect to statement x is simply a proof with respect to x, where m is additionally included upon computing the signature using Σ_{ot}, i.e., one signs $\pi_{\Pi}||x||\mathsf{pk}||\sigma||m$. Then one obtains signatures of knowledge in the strong sense [BCC^{+}15], where even the signature (i.e., the proof) is non-malleable. If security in the original sense—the counterpart of weak simulation-sound extractability where the signature (i.e., the proof π) itself may be malleable—is sufficient, one can even omit the strong one-time signature scheme and directly sign $\pi_{\Pi}||x||\mathsf{pk}||m$ using Σ.

As already mentioned in [CL06], a straight forward application of signatures of knowledge is the construction of ring signatures. Obtaining such a construction based on our techniques presented in this section can thus be seen as an alternative to the direct construction in Section 4.1.

Additionally we note that our technique provides nice properties when it comes to converting Groth-Sahai proofs [GS08] over pairing product equations to simulation-sound extractable arguments of knowledge. We can use Waters' signatures as described in Appendix B.4. Here the secret keys as well as the shift amounts are group elements and the required relations can be proven using a few simple pairing product equations. Thus our technique constitutes an alternative to known techniques and yields conceptually simpler constructions with favorable properties regarding efficiency when, e.g., compared to [Gro06] for SSE NIZKs or [BFG13] for signatures of knowledge without random oracles.

4.4 Multisignatures

A multisignature scheme [IN83] is a signature scheme that allows a group of signers to jointly compute a compact signature for a message. Well known schemes are the BMS [Bol03] and the WMS [LOS^{+}06] schemes that are directly based on the BLS [BLS04] and variants of the Waters' signature scheme [Wat05] respectively. Both of them are secure under the knowledge of secret key (KOSK) assumption, but can be shown to also be secure under (slightly tweaked) real-world proofs of possession protocols [RY07].

Our construction can be seen as a generalization of the paradigm behind all existing multisignature schemes. Making this paradigm explicit eases the search for new schemes, i.e., one can simply check whether a particular signature scheme is publicly key-homomorphic. For instance, as we show in Appendix B.6, the modified CL signature scheme from [CHP12] provides this key-homomorphism, and, therefore, directly yields a new instantiation of multisignatures.

We now give a formal definition of multisignatures, where we follow Ristenpart and Yilek [RY07]. As already noted in Section 2, we use the KOSK modeled via RKey for simplicity. Nevertheless, we stress that we could use any other key-registration that provides extractability or also the extractable key-verification notion by Bagherzandi and Jarecki [BJ08]. This does not make any difference for our subsequent discussion as long as the secret keys are extractable.

Definition 25. *A multisignature scheme* MS *is a tuple* $(\mathsf{PGen}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *of PPT algorithms, which are defined as follows:*

$\mathsf{PGen}(1^\kappa)$: *This paramter generation algorithm takes a security parameter* κ *and produces global parameters* PP *(including the security parameters and a description of the message space* $\mathcal{M}$*).*

$\mathsf{KeyGen}(\mathsf{PP})$: *This algorithm takes the global parameters* PP *as input and outputs a secret (signing) key* sk *and a public (verification) key* pk*.*

Sign : *This is an interactive multisignature algorithm executed by a group of signers who intend to sign the same message* m*. Each signer* S_i *executes* Sign *on public inputs* PP*, public key multiset* PK*, message* m *and secret input its secret* sk_i *and outputs a multisignature* σ*.*

$\mathsf{Verify}(\mathsf{PP}, \mathsf{PK}, m, \sigma)$: *This algorithm takes public parameters* PP*, a public key multiset* PK*, a message* m *and a multisignature* σ *as input and outputs a bit* $b \in \{0, 1\}$*.*

The above tuple of algorithms must satisfy correctness, which basically states that $\mathsf{Verify}(\mathsf{PP}, \mathsf{PK}, m, \mathsf{Sign}(\mathsf{PP}, \mathsf{PK}, m, \mathsf{sk})) = 1$ for any m, any honestly generated PP and when every participant correctly follows the algorithms. Besides correctness, we require existential unforgeability under a chosen message attack against a single honest player.

Definition 26 ($\mathsf{MSEUF\text{-}CMA}$). *A multisignature scheme* MS *is* $\mathsf{MSEUF\text{-}CMA}$ *secure, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l} \mathsf{PP} \leftarrow \mathsf{PGen}(1^\kappa), \\ (\mathsf{sk}^\star, \mathsf{pk}^\star) \leftarrow \mathsf{KeyGen}(1^\kappa), \\ \mathcal{O} \leftarrow \{\mathsf{Sign}(\cdot,\cdot), \mathsf{RKey}(\cdot,\cdot,\cdot)\}, \\ (\mathsf{PK}^\star, m^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{PP}, \mathsf{pk}^\star) \end{array} : \begin{array}{c} \mathsf{Verify}(\mathsf{PP}, \mathsf{PK}^\star, m^\star, \sigma^\star) = 1 \;\wedge \\ \mathsf{pk}^\star \in \mathsf{PK}^\star \;\wedge\; m^\star \notin \mathcal{Q}^{\mathsf{Sign}} \;\wedge \\ (\mathsf{PK}^\star \setminus \{\mathsf{pk}^\star\}) \setminus \mathcal{Q}^{\mathsf{RKey}} = \emptyset) \end{array}\right] \leq \varepsilon(\kappa),$$

where the environment keeps track of signing and registration queries via $\mathcal{Q}^{\mathsf{Sign}}$ *and* $\mathcal{Q}^{\mathsf{RKey}}$*, respectively. The adversary has access to the following oracles:*

$\mathsf{Sign}(\mathsf{PK}, m)$: *This oracle obtains a public key set* PK *and returns* $\perp$ *if* $\mathsf{pk}^\star \notin \mathsf{PK}$*. Otherwise it simulates a new instance of* $\mathsf{Sign}(\mathsf{PP}, \mathsf{PK}, m, \mathsf{sk}^\star)$ *forwarding messages to and from* $\mathcal{A}$ *appropriately and sets* $\mathcal{Q}^{\mathsf{Sign}} \overset{\cup}{\leftarrow} m$*.*

$\mathsf{RKey}(\mathsf{sk}, \mathsf{pk})$: *This oracle checks if* $(\mathsf{sk}, \mathsf{pk}) \in \mathsf{KeyGen}(\mathsf{PP})$ *and sets* $\mathcal{Q}^{\mathsf{RKey}} \overset{\cup}{\leftarrow} \mathsf{pk}$ *if so.*

Our Construction. Subsequently, we restrict ourselves to non-interactive Sign protocols, which basically means that every signer S_i locally computes a signatures σ_i and then broadcasts it to all other signers in PK. Furthermore, we consider the signature scheme Σ to work with common parameters $\textsc{pp}$ and in Scheme 4 let us for the sake of presentation assume that $\textsc{pk} := (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ is an ordered set instead of a multiset.

$\mathsf{PGen}(1^\kappa)$: Run $\textsc{pp} \leftarrow \Sigma.\mathsf{PGen}(1^\kappa)$ and return $\textsc{pp}$.

$\mathsf{KeyGen}(\textsc{pp})$: Run $(\mathsf{sk}, \mathsf{pk}) \leftarrow \Sigma.\mathsf{KeyGen}(\textsc{pp})$ and return $(\mathsf{sk}, \mathsf{pk})$.

$\mathsf{Sign}(\textsc{pp}, \textsc{pk}, m, \mathsf{sk})$: Let $i \in [n]$. Every participating S_i with $\mathsf{pk}_i \in \textsc{pk}$ proceeds as follows:

- Compute $\sigma_i \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}_i, m)$ and broadcast σ_i.
- Receive all signatures σ_j for $j \neq i$.
- Compute $(\mathsf{pk}, \sigma) \leftarrow \mathsf{Combine}(\textsc{pk}, m, (\sigma_\ell)_{\ell \in [n]})$ and output σ.

$\mathsf{Verify}(\textsc{pp}, \textsc{pk}, m, \sigma)$: Return 1 if the following holds and 0 otherwise:

$$\Sigma.\mathsf{Verify}\left(\prod_{\mathsf{pk} \in \textsc{pk}} \mathsf{pk}, m, \sigma\right) = 1.$$

Scheme 4: Black-Box Construction of Multisignatures

Theorem 5. *If Σ is correct,* EUF-CMA *secure, and publicly key-homomorphic, then Scheme 4 is* MSEUF-CMA *secure.*

Proof. We show that an efficient adversary $\mathcal{A}$ against MSEUF-CMA can be efficiently turned into an efficient EUF-CMA adversary for Σ. To do so, we simulate the environment for $\mathcal{A}$ by obtaining $\mathsf{pk}^\star$ from an EUF-CMA challenger of Σ, then setting $\textsc{pp}$ accordingly, and starting $\mathcal{A}$ on $(\textsc{pp}, \mathsf{pk}^\star)$. Additionally, we record the secret keys provided to RKey in a list $\mathtt{KEY}$ indexed by the respective public keys, i.e., $\mathtt{KEY}[\mathsf{pk}] \leftarrow \mathsf{sk}$. Whenever a signature with respect to $\mathsf{pk}^\star$ is required we use the Sign oracle provided by the challenger. Eventually, the adversary outputs $(\textsc{pk}^\star, m^\star, \sigma^\star)$ such that $\Sigma.\mathsf{Verify}(\prod_{\mathsf{pk} \in \textsc{pk}^\star} \mathsf{pk}, m^\star, \sigma^\star) = 1$, $\mathsf{pk}^\star \in \textsc{pk}^\star$, all other keys in $\textsc{pk}^\star$ were registered, yet $m^\star$ was never queried to the signing oracle. We compute $\mathsf{sk}' \leftarrow \sum_{\mathsf{pk} \in \textsc{pk}^\star \setminus \{\mathsf{pk}^\star\}\}} -\mathtt{KEY}[\mathsf{pk}]$, compute $\sigma' \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}', m^\star)$, obtain $(\mathsf{pk}^\star, \sigma) \leftarrow \mathsf{Combine}((\prod_{\mathsf{pk} \in \textsc{pk}^\star} \mathsf{pk}, \prod_{\mathsf{pk} \in \textsc{pk}^\star \setminus \{\mathsf{pk}^\star\}} \mathsf{pk}^{-1}), m^\star, (\sigma^\star, \sigma'))$ and output $(m^\star, \sigma)$ as a forgery. $\square$

4.5 Tight Multi-User Security from Key-Homomorphisms

When using signature schemes in practice, it is often argued that EUF-CMA security does not appropriately capture the requirements appearing in practical settings [GMS02, MS04]. Currently we experience a growing interest in the multi-user setting (e.g., [BJLS16, GHKW16, KMP16]), where an adversary can attack one out of various public keys instead of a single one. This setting is also a frequently discussed topic on the mailing list of the CFRG.[7]

[7] https://www.ietf.org/mail-archive/web/cfrg/current/maillist.html

Since many schemes have already been investigated regarding their single-user security, an important question in this context is whether one can infer statements about the multi-user security of a certain scheme based on its single-user security. Without using any further properties of the signature scheme, every naïve reduction looses a factor of N, where N is the number of users in the system [GMS02].[8] Such a reduction is non-tight and drastically reduces the security guarantees a scheme provably provides. Thus, it is important to come up with tight security reductions. This was done in [GMS02], where a tight implication from single-user $\mathsf{EUF\text{-}CMA}$ to multi-user $\mathsf{EUF\text{-}CMA}$ for Schnorr signatures was proven. Unfortunately, a flaw in this proof was discovered by Bernstein in [Ber15], where it was also shown that single-user $\mathsf{EUF\text{-}CMA}$ tightly implies key-prefixed mulit-user $\mathsf{EUF\text{-}CMA}$ for Schnorr signatures. Recently, Lacharité in [Lac16] showed this tight implication under key-prefixing for BLS [BLS04] signatures and BGLS [BGLS03] aggregate signatures. Subsequent to the work in [Ber15], Kiltz et al. [KMP16] studied multi-user security of random self-reducible canonical identification schemes when turned to signatures in the random oracle model using the Fiat-Shamir heuristic. They show that for such schemes single-user security tightly implies multi-user security without key-prefixing. This, in particular, holds for Schnorr signatures.

Our theorem essentially generalizes the work of [Ber15, Lac16] to be applicable to a larger class of signature schemes. For example, using our results from Appendix B, it attests the multi-user $\mathsf{EUF\text{-}CMA}$ security of various variants of Water's signatures [Wat05], PS signatures [PS16b], and the CL signature [CL04] variant from [CHP12], which were previously unknown to provide tight multi-user security. Furthermore, it can be seen as orthogonal to the work of [KMP16], where the requirement of key-prefixing is avoided at the cost of tailoring the results to a class of signature schemes from specific canonical identification schemes in the random oracle model.

Subsequently, we will first recall a definition of multi-user $\mathsf{EUF\text{-}CMA}$ and then prove Theorem 6, which formalizes the main result of this section.

Definition 27 ($\mathsf{MU\text{-}EUF\text{-}CMA}$). *A signature scheme Σ is $\mathsf{MU\text{-}EUF\text{-}CMA}$ secure, if for all PPT adversaries $\mathcal{A}$ there is a negligible function $\varepsilon(\cdot)$ such that*

$$\Pr\left[\begin{array}{l}\{(\mathsf{sk}_i,\mathsf{pk}_i)\leftarrow\mathsf{KeyGen}(1^\kappa)\}_{i\in[\mathsf{poly}(\kappa)]},\\ (i^\star,m^\star,\sigma^\star)\leftarrow\mathcal{A}^{\mathsf{Sign}(\cdot,\cdot)}(\{\mathsf{pk}_i\}_{i\in[\mathsf{poly}(\kappa)]})\end{array} : \begin{array}{r}\mathsf{Verify}(\mathsf{pk}_{i^\star},m^\star,\sigma^\star)=1\ \wedge\\ (i^\star,m^\star)\notin\mathcal{Q}^{\mathsf{Sign}}\end{array}\right]\le\varepsilon(\kappa),$$

where $\mathsf{Sign}(i,m) := \Sigma.\mathsf{Sign}(\mathsf{sk}_i,m)$ and the environment keeps track of the queries to the signing oracle via $\mathcal{Q}^{\mathsf{Sign}}$.

Theorem 6. *Let $\Sigma = (\mathsf{KeyGen},\mathsf{Sign},\mathsf{Verify})$ be a signature scheme which provides adaptability of signatures where the success ratio of any $\mathsf{EUF\text{-}CMA}$ adversary is ρ. Then the success ratio of any adversary against $\mathsf{MU\text{-}EUF\text{-}CMA}$ of $\Sigma' = (\mathsf{KeyGen}',\mathsf{Sign}',\mathsf{Verify}')$ is $\rho' \approx \rho$, where $\mathsf{KeyGen}'(1^\kappa) := \mathsf{KeyGen}(1^\kappa)$, $\mathsf{Sign}'(\mathsf{sk},m) := \mathsf{Sign}(\mathsf{sk},\mu(\mathsf{sk})||m)$, and $\mathsf{Verify}(\mathsf{pk},m,\sigma) := \mathsf{Verify}(\mathsf{pk},\mathsf{pk}||m,\sigma)$.*

[8] For instance, assuming 2^{30} keys in a system, such a reduction loss requires to significantly increase the parameters.

Proof. First, our reduction $\mathcal{R}$ obtains a public key pk_1 from an $\mathsf{EUF\text{-}CMA}$ challenger $\mathcal{C}$ and initializes an empty list $\mathtt{SK}$. It sets $\mathtt{SK}[1] \leftarrow 0$, and for $2 \leq i \leq \mathsf{poly}(\kappa)$, it chooses $\mathtt{SK}[i] \xleftarrow{R} \mathbb{H}$, and sets $\mathsf{pk}_i \leftarrow \mathsf{pk}_1 \cdot \mu(\mathtt{SK}[i])$. Then, it starts $\mathcal{A}$ on $\{\mathsf{pk}_i\}_{i\in[\mathsf{poly}(\kappa)]}$ and simulates Sign' inside the $\mathsf{Sign}(\cdot,\cdot)$ oracle as follows (where $\mathcal{C}.\mathsf{Sign}(\cdot)$ denotes the signing oracle provided by $\mathcal{C}$).

$\mathsf{Sign}(i, m)$: Obtain $\sigma \leftarrow \mathcal{C}.\mathsf{Sign}(\mathsf{pk}_i||m)$, compute $(\mathsf{pk}_i, \sigma') \leftarrow \mathsf{Adapt}(\mathsf{pk}_1, \mathsf{pk}_i||m, \sigma, \mathtt{SK}[i])$, and return σ'.

Eventually, $\mathcal{A}$ outputs a forgery $(i^\star, m^\star, \sigma^\star)$, where $(i^\star, m^\star) \notin \mathcal{Q}^{\mathsf{Sign}}$ by definition. Thus, $\mathcal{R}$ has never sent $\mathsf{pk}_{i^\star}||m^\star$ to the sign oracle of $\mathcal{C}$ and can obtain $(\mathsf{pk}_1, \sigma'^\star) \leftarrow \mathsf{Adapt}(\mathsf{pk}_{i^\star}, \mathsf{pk}_{i^\star}||m^\star, \sigma^\star, -\mathtt{SK}[i])$ and output $(\mathsf{pk}_{i^\star}||m^\star, \sigma'^\star)$ as an $\mathsf{EUF\text{-}CMA}$ forgery. Due to adaptability of signatures the simulation of the oracle is perfect; the running time of $\mathcal{R}$ is approximately the same as the running time of $\mathcal{A}$ which concludes the proof. $\square$

It is quite straight forward to see that such an implication can also be proven for weaker unforgeability notions. Essentially the security proof would be analogous, but without the need to simulate the signing oracle. Furthermore, it is important to note that for key-recovery attacks, where no signatures need to be simulated, a secret key to public key homomorphism would be sufficient to tightly relate the single-user setting to the key-prefixed multi-user setting.

5 Conclusion

In this paper we introduce a definitional framework distilling various natural flavours of key-homomorphisms for signatures, and, thereby, generalize larger classes of existing signature schemes. We present elegant and simple compilers turning classes of schemes admitting particular key-homomorphisms into ring signatures, universal designated verifier signatures, simulation sound extractable argument systems, as well as multisignatures. Furthermore, we also prove a tight implication from single-user security to key-prefixed multi-user security for a class of schemes admitting a certain key-homomorphism. We give examples of existing signature schemes admitting the introduced key-homomorphisms, which yields to novel instantiations of the various schemes, including instantiations under standard assumptions without random oracles and even standard model instantiations under standard assumptions. They favorably compare to existing instantiations regarding conceptual simplicity and efficiency. Furthermore, our results attest the tight multi-user security of various schemes which were previously unknown to provide tight multi-user security. Finally, we introduce the notion of multikey homomorphic signatures (cf. Appendix A) and show that a secret-key to public-key homomorphism implies the existence of key-succinct multikey-homomorphic signatures.

Acknowledgements. We thank the anonymous referees from TCC 2016-B and PKC 2017 for their valuable comments.

References

[ABC$^+$12] Jae Hyun Ahn, Dan Boneh, Jan Camenisch, Susan Hohenberger, Abhi Shelat, and Brent Waters. Computing on authenticated data. In *TCC*, 2012.

[AHI11] Benny Applebaum, Danny Harnik, and Yuval Ishai. Semantic Security under Related-Key Attacks and Applications. In *ICS*, 2011.

[ALP12] Nuttapong Attrapadung, Benoît Libert, and Thomas Peters. Computing on Authenticated Data: New Privacy Definitions and Constructions. In *ASIACRYPT*, 2012.

[BBL17] Fabrice Benhamouda, Florian Bourse, and Helger Lipmaa. CCA-secure inner-product functional encryption from projective hash functions. PKC, 2017.

[BCC$^+$15] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, Jens Groth, and Christophe Petit. Short accountable ring signatures based on DDH. In *ESORICS*, 2015.

[BCM11] Mihir Bellare, David Cash, and Rachel Miller. Cryptography Secure against Related-Key Attacks and Tampering. In *ASIACRYPT*, 2011.

[Ber15] Daniel J. Bernstein. Multi-user schnorr security, revisited. *IACR Cryptology ePrint Archive*, 2015, 2015.

[BF11] Dan Boneh and David Mandell Freeman. Homomorphic signatures for polynomial functions. In *EUROCRYPT*. 2011.

[BFG13] David Bernhard, Georg Fuchsbauer, and Essam Ghadafi. Efficient signatures of knowledge and DAA in the standard model. In *ACNS*, 2013.

[BFKW09] Dan Boneh, David Mandell Freeman, Jonathan Katz, and Brent Waters. Signing a Linear Subspace: Signature Schemes for Network Coding. In *PKC*, 2009.

[BFP$^+$15] Abhishek Banerjee, Georg Fuchsbauer, Chris Peikert, Krzysztof Pietrzak, and Sophie Stevens. Key-Homomorphic Constrained Pseudorandom Functions. In *TCC*, 2015.

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

[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 *EUROCRYPT*, 2014.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In *PKC*, 2014.

[BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *EUROCRYPT*, 2003.

[BGR98] Mihir Bellare, Juan A. Garay, and Tal Rabin. Fast batch verification for modular exponentiation and digital signatures. In *EUROCRYPT*, 1998.

[BJ08] Ali Bagherzandi and Stanislaw Jarecki. Multisignatures using proofs of secret key possession, as secure as the diffie-hellman problem. In *SCN*, 2008.

[BJL16] Fabrice Benhamouda, Marc Joye, and Benoît Libert. A new framework for privacy-preserving aggregation of time-series data. *ACM Trans. Inf. Syst. Secur.*, 18(3), 2016.

[BJLS16] Christoph Bader, Tibor Jager, Yong Li, and Sven Schäge. On the impossibility of tight cryptographic reductions. In *EUROCRYPT*, 2016.

[BK10] Zvika Brakerski and Yael Tauman Kalai. A framework for efficient signatures, ring signatures and identity based encryption in the standard model. *IACR Cryptology ePrint Archive*, 2010.

[BKM09] Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. *J. Cryptology*, 22(1), 2009.

[BLMR13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key Homomorphic PRFs and Their Applications. In *CRYPTO*, 2013.

[BLS04] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. *J. Cryptology*, 17(4), 2004.

[Bol03] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *PKC*, 2003.

[BP14] Abhishek Banerjee and Chris Peikert. New and Improved Key-Homomorphic Pseudorandom Functions. In *CRYPTO*, 2014.

[BP16] Zvika Brakerski and Renen Perlman. Lattice-based fully dynamic multi-key FHE with short ciphertexts. In *CRYPTO*, 2016.

[BPT12] Mihir Bellare, Kenneth G. Paterson, and Susan Thomson. RKA security beyond the linear barrier: Ibe, encryption and signatures. In *ASIACRYPT*, 2012.

[Cat14] Dario Catalano. Homomorphic signatures and message authentication codes. In *SCN*, 2014.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO*, 1994.

[CFW14] Dario Catalano, Dario Fiore, and Bogdan Warinschi. Homomorphic signatures with efficient verification for polynomial functions. In *CRYPTO*. 2014.

[CGS07] Nishanth Chandran, Jens Groth, and Amit Sahai. Ring signatures of sublinear size without random oracles. In *ICALP*, 2007.

[CHKM10] Sanjit Chatterjee, Darrel Hankerson, Edward Knapp, and Alfred Menezes. Comparing two pairing-based aggregate signature schemes. *Des. Codes Cryptography*, 55(2-3), 2010.

[CHP12] Jan Camenisch, Susan Hohenberger, and Michael Østergaard Pedersen. Batch verification of short signatures. *J. Cryptology*, 25(4), 2012.

[CL04] Jan Camenisch and Anna Lysyanskaya. Signature Schemes and Anonymous Credentials from Bilinear Maps. In *CRYPTO*, 2004.

[CL06] Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In *CRYPTO*, 2006.

[CM15] Michael Clear and Ciaran McGoldrick. Multi-identity and multi-key leveled FHE from learning with errors. In *CRYPTO*, 2015.

[DKNS04] Yevgeniy Dodis, Aggelos Kiayias, Antonio Nicolosi, and Victor Shoup. Anonymous identification in ad hoc groups. In *EUROCRYPT*, 2004.

[DKS16] David Derler, Stephan Krenn, and Daniel Slamanig. Signer-Anonymous Designated-Verifier Redactable Signatures for Cloud-Based Data Sharing. In *CANS*, 2016.

[DMS16] Yevgeniy Dodis, Ilya Mironov, and Noah Stephens-Davidowitz. Message transmission with reverse firewalls - secure communication on corrupted machines. In *CRYPTO*, 2016.

[FF13] Marc Fischlin and Nils Fleischhacker. Limitations of the meta-reduction technique: The case of schnorr signatures. In *EUROCRYPT*, 2013.

[FHS15] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Practical round-optimal blind signatures in the standard model. In *CRYPTO*, 2015.

[FKM$^+$16] Nils Fleischhacker, Johannes Krupp, Giulio Malavolta, Jonas Schneider, Dominique Schröder, and Mark Simkin. Efficient Unlinkable Sanitizable Signatures from Signatures with Re-randomizable Keys. In *PKC*, 2016.

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

[Fre12] David Mandell Freeman. Improved security for linearly homomorphic signatures: A generic framework. 2012.

[Gen09] Craig Gentry. Fully Homomorphic Encryption using Ideal Lattices. In *STOC*, 2009.

[GHKW16] Romain Gay, Dennis Hofheinz, Eike Kiltz, and Hoeteck Wee. Tightly cca-secure encryption without pairings. In *EUROCRYPT*, 2016.

[GJKW07] Eu-Jin Goh, Stanislaw Jarecki, Jonathan Katz, and Nan Wang. Efficient signature schemes with tight reductions to the diffie-hellman problems. *J. Cryptology*, 20(4), 2007.

[GK15] Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In *EUROCRYPT*, 2015.

[GLW12] Shafi Goldwasser, Allison B. Lewko, and David A. Wilson. Bounded-Collusion IBE from Key Homomorphism. In *TCC*, 2012.

[GMS02] Steven D. Galbraith, John Malone-Lee, and Nigel P. Smart. Public key signatures in the multi-user setting. *Inf. Process. Lett.*, 83(5), 2002.

[GMY03] Juan A. Garay, Philip D. MacKenzie, and Ke Yang. Strengthening zero-knowledge protocols using signatures. In *EUROCRYPT*, 2003.

[GMY06] Juan A. Garay, Philip D. MacKenzie, and Ke Yang. Strengthening zero-knowledge protocols using signatures. *J. Cryptology*, 19(2), 2006.

[Gro06] Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In *ASIACRYPT*, 2006.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In *EUROCRYPT*, 2008.

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled fully homomorphic signatures from standard lattices. In *STOC*, 2015.

[HKW15] Susan Hohenberger, Venkata Koppula, and Brent Waters. Universal signature aggregators. In *EUROCRYPT*, 2015.

[HS14] Christian Hanser and Daniel Slamanig. Structure-preserving signatures on equivalence classes and their application to anonymous credentials. In *ASIACRYPT*, 2014.

[IN83] K. Itakura and K. Nakamura. A public-key cryptosystem suitable for digital multisignatures. *NEC Research & Development*, 71, 1983.

[JMSW02] Robert Johnson, David Molnar, Dawn Xiaodong Song, and David Wagner. Homomorphic signature schemes. In *CT-RSA*, 2002.

[JSI96] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated verifier proofs and their applications. In *EUROCRYPT*, 1996.

[KMP16] Eike Kiltz, Daniel Masny, and Jiaxin Pan. Optimal security proofs for signatures from identification schemes. In *CRYPTO*, 2016.

[KW03] Jonathan Katz and Nan Wang. Efficiency improvements for signature schemes with tight security reductions. In *CCS*, 2003.

[Lac16] Marie-Sarah Lacharité. Security of bls and bgls signatures in a multi-user setting. Arcticcrypt 2016 Talk, http://arcticcrypt.b.uib.no/files/2016/07/Slides-Lacharite.pdf, 2016.

[LOS$^+$06] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. In *EUROCRYPT*, 2006.

[LTV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *STOC*, 2012.

[LTWC16] Russell W. F. Lai, Raymond K. H. Tai, Harry W. H. Wong, and Sherman S. M. Chow. A zoo of homomorphic signatures: Multi-key and key-homomorphism. Cryptology ePrint Archive, Report 2016/834, 2016.

[MS04] Alfred Menezes and Nigel P. Smart. Security of signature schemes in a multi-user setting. *Des. Codes Cryptography*, 33(3), 2004.

[MSM$^+$15] Hiraku Morita, Jacob C. N. Schuldt, Takahiro Matsuda, Goichiro Hanaoka, and Tetsu Iwata. On the security of the schnorr signature scheme and DSA against related-key attacks. In *ICISC*, 2015.

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In *EUROCRYPT*, 2016.

[PS16a] Chris Peikert and Sina Shiehian. Multi-key FHE from lwe, revisited. *IACR Cryptology ePrint Archive*, 2016.

[PS16b] David Pointcheval and Olivier Sanders. Short Randomizable Signatures. In *CT-RSA*, 2016.

[Rot11] Ron Rothblum. Homomorphic Encryption: From Private-Key to Public-Key. In *TCC*, 2011.

[RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In *ASIACRYPT*, 2001.

[RY07] Thomas Ristenpart and Scott Yilek. The Power of Proofs-of-Possession: Securing Multiparty Signatures against Rogue-Key Attacks. In *EUROCRYPT*, 2007.

[SBWP03] Ron Steinfeld, Laurence Bull, Huaxiong Wang, and Josef Pieprzyk. Universal designated-verifier signatures. In *ASIACRYPT*, 2003.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3), 1991.

[SS08] Siamak Fayyaz Shahandashti and Reihaneh Safavi-Naini. Construction of universal designated-verifier signatures and identity-based signatures from standard signatures. In *PKC*, 2008.

[TW14] Stefano Tessaro and David A. Wilson. Bounded-collusion identity-based encryption from semantically-secure public-key encryption: Generic constructions with short ciphertexts. In *PKC*, 2014.

[Wat05] Brent Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT*, 2005.

A Homomorphisms on Key and Message Space

As already mentioned in Section 1, signature schemes with homomorphic properties on their message space [JMSW02] are well known. With such schemes, it is possible for anyone to derive a signature for a message m' from signatures on messages $(m_i)_{i\in[n]}$ under some public key pk as long as $m' = f(m_1, \ldots, m_n)$ for $f \in \mathcal{F}$, where $\mathcal{F}$ is the set of so called admissible functions (determined by the scheme). Among others (cf. [ABC$^+$12, ALP12]) there are schemes for linear functions [BFKW09, Fre12], polynomial functions of higher degree [BF11, CFW14] and meanwhile even (levelled) fully homomorphic signatures supporting arbitrary functions [GVW15, BFS14]. However, all existing constructions consider these homomorphisms under *a single* key. While in context of encryption, constructions working with distinct keys, i.e., so called multikey-homomorphic encryption schemes [LTV12, CM15, MW16, PS16a], are known, such a feature has never been investigated in context of signatures so far.

In this section we close this gap and initiate the study of so called multikey-homomorphic signatures and in particular propose a definitional framework for such schemes that support a homomorphic property on the message space under *distinct* keys. Moreover, we discuss potential applications of such schemes.

Concurrent Work. In independent and concurrent work, Fiore et al. [FMNP16] introduced the concept of multikey-homomorphic authenticators, which also covers multikey-homomorphic signatures. They also present a construction of multikey-homomorphic signatures from standard lattices based on the fully homomorphic signatures in [GVW15]. Their model and construction focuses on achieving succinct combined signatures, whereas the focus of our construction (feasibility result) is on achieving succinct combined keys. We also note that the independent and concurrent work of Lai et al. [LTWC16] yields a multikey-homomorphic signature scheme with succinct combined keys and signatures. However, they require rather heavy tools (and assumptions) such as zk-SNARKS, while our feasibility result for succinct combined keys only requires a very mild assumption.

A.1 Multikey-Homomorphic Signatures

Below we present and discuss what we call *multikey-homomorphic signatures*, where the homomorphic property on the message space is defined with respect to a class $\mathcal{F}$ of admissible functions (e.g., represented as arithmetic circuits). In contrast to the notions from Section 3, which capture additional properties of conventional signature schemes, multikey-homomorphic signatures are a separate building block. To this end we explicitly formalize the algorithms as well as the required correctness and unforgeability notion. We stress that, as the focus of this work lies on key-homomorphic schemes, we will also focus on these aspects in this section. In particular, while we present a general definition of multikey-homomorphic schemes which, in analogy to the encryption case, i.e., [LTV12, CM15, MW16, PS16a, BP16], support the input of a set of public keys into the verification of a combined signature, we focus on schemes who use a *succinct* representation of a combined public key in the verification below.

Definition 28 (Multikey-Homomorphic Signatures). *A multikey-homomorphic signature scheme for a class $\mathcal{F}$ of admissible functions, is a tuple of the following PPT algorithms:*

$\mathsf{PGen}(1^\kappa)$: *Takes a security parameter κ as input, and outputs parameters* PP.

$\mathsf{KeyGen}(\mathsf{PP})$: *Takes parameters* PP *as input, and outputs a keypair* $(\mathsf{sk}, \mathsf{pk})$ *(we assume that* PP *is included in* pk*).*

$\mathsf{Sign}(\mathsf{sk}, m, \tau)$: *Takes a secret key* sk*, a message m, and a tag τ as input, and outputs a signature σ.*

$\mathsf{Verify}(\mathsf{pk}, m, \sigma, \tau)$: *Takes a public key* pk *a message m, a signature σ, and a tag τ as input, and outputs a bit b.*

$\mathsf{Combine}((\mathsf{pk}_i)_{i\in[n]}, (m_i)_{i\in[n]}, f, (\sigma_i)_{i\in[n]}, \tau)$: *Takes public keys $(\mathsf{pk}_i)_{i\in[n]}$, messages $(m_i)_{i\in[n]}$, a function $f \in \mathcal{F}$, signatures $(\sigma_i)_{i\in[n]}$, and a tag τ as input, and outputs a public key $\hat{\mathsf{pk}}$ and a signature $\hat{\sigma}$.*

$\mathsf{Verify}'(\hat{\mathsf{pk}}, \hat{m}, f, \hat{\sigma}, \tau)$: *Takes a combined public key $\hat{\mathsf{pk}}$, a message $\hat{m}$, a function f, a signature $\hat{\sigma}$, and a tag τ as input, and outputs a bit b.*

Subsequently, we formalize the security properties one would expect from such schemes.

Definition 29 (Correctness). *A multikey-homomorphic signature scheme for a class $\mathcal{F}$ of admissible functions is correct, if for all security parameters κ, for all $1 \le n \le \mathsf{poly}(\kappa)$, all $((\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{KeyGen}(1^\kappa))_{i\in[n]}$, all messages $(m_i)_{i\in[n]}$, all tags τ, all functions $f \in \mathcal{F}$, all functions $f' \notin \mathcal{F}$, and all signatures $(\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}_i, m_i, \tau))_{i=1}^n$ and results $(\hat{\mathsf{pk}}, \hat{\sigma}) \leftarrow \mathsf{Combine}((\mathsf{pk}_i)_{i\in[n]}, (m_i)_{i\in[n]}, f, (\sigma_i)_{i\in[n]}, \tau)$ it holds that*

$$(\mathsf{Verify}(\mathsf{pk}_i, m_i, \sigma_i, \tau) = 1)_{i\in[n]} \;\wedge\; (\mathsf{pk}_i \in \hat{\mathsf{pk}})_{i\in[n]} \;\wedge$$
$$\mathsf{Verify}'(\hat{\mathsf{pk}}, \hat{m}, f, \hat{\sigma}, \tau) = 1 \;\wedge\; \mathsf{Verify}'(\cdot, \cdot, f', \cdot, \cdot) = 0,$$

where $\hat{m} = f(m_1, \ldots, m_n)$.

We note that the predicate "$\in$" for the check $\mathsf{pk} \in \hat{\mathsf{pk}}$ needs to be explicitly defined by every concrete scheme (i.e., it is not necessarily a simple set membership check).

Definition 30 (Unforgeability). *A multikey-homomorphic signature scheme for a class $\mathcal{F}$ of admissible functions is unforgeable, if for every PPT adversary $\mathcal{A}$ there exists a negligible function $\epsilon(\cdot)$ such that it holds that*

$$\Pr\left[\begin{array}{ll} \mathsf{PP} \leftarrow \mathsf{PGen}(1^\kappa), & \mathsf{Verify}'(\hat{\mathsf{pk}}^\star, \hat{m}^\star, f^\star, \hat{\sigma}^\star, \tau^\star) = 1 \;\wedge \\ (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{PP}), & \quad (\mathsf{pk} \in \hat{\mathsf{pk}}^\star \;\wedge\; \nexists\, m \in \mathcal{M} : \\ \mathcal{O} \leftarrow \{\mathsf{Sig}(\cdot,\cdot)\}, & \quad (\hat{m}^\star \in \mathsf{R}(f^\star(\cdots, m, \cdots)) \;\wedge \\ (\hat{\mathsf{pk}}^\star, \hat{m}^\star, f^\star, \hat{\sigma}^\star, \tau^\star) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pk}), & (m, \tau^\star) \in \mathcal{Q}^{\mathsf{Sig}})) \;\vee\; \hat{m}^\star \notin \mathsf{R}(f^\star) \end{array}\right] \le \epsilon(\kappa),$$

where $\mathsf{Sig}(m, \tau) := \mathsf{Sign}(\mathsf{sk}, m, \tau)$ *and* $\mathcal{Q}^{\mathsf{Sig}}$ *records the* Sig *queries.*

Observe that Definition 28 neither puts restrictions on the size of signatures $\hat{\sigma}$ nor public keys $\hat{\mathsf{pk}}$. To really benefit from the functionality provided by multikey-homomorphic signatures, one may additionally require that $\hat{\mathsf{pk}}$ is succinct. Inspired by [BGI14], we subsequently provide a formal definition.

Definition 31 (Key Succinctness). *A multikey-homomorphic signature scheme is called key succinct, if for all $\kappa \in \mathbb{N}$, for all $n \leq \mathsf{poly}(\kappa)$, for all $\mathsf{PP} \leftarrow \mathsf{PGen}(1^\kappa)$, for all $((\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{PP}))_{i\in[n]}$, for all $(m_i)_{i\in[n]} \in \mathcal{M}^n$, all $(\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}_i, m_i))_{i\in[n]}$, all $(\hat{\mathsf{pk}}, \hat{\sigma}) \leftarrow \mathsf{Combine}((\mathsf{pk}_i)_{i\in[n]}, (m_i)_{i\in[n]}, f, (\sigma_i)_{i\in[n]})$ it holds that*

$$|\hat{\mathsf{pk}}| \leq \mathsf{poly}(\kappa).$$

It turns out that secret key to public key homomorphic signature schemes already imply the existence of key succinct multikey-homomorphic signature schemes for a class $\mathcal{F}$ of functions with polynomially many members.

Lemma 2. *If there exists an* EUF-CMA *secure secret key to public key homomorphic signature scheme Σ, then there exists a key succinct multikey-homomor-phic signature scheme $\Sigma_\mathcal{F}$ for a class $\mathcal{F}$ of functions with polynomially many members.*

Proof. We prove this lemma by constructing such a scheme. In particular, we base the construction on a wrapped version $\Sigma_\mathcal{F} = (\mathsf{KeyGen}_\mathcal{F}, \mathsf{Sign}_\mathcal{F}, \mathsf{Verify}_\mathcal{F})$ of the secret key to public key homomorphic signature scheme $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$, where $\mathsf{KeyGen}_\mathcal{F}(1^\kappa) := \mathsf{KeyGen}(1^\kappa)$, $\mathsf{Sign}_\mathcal{F}(\mathsf{sk}, m, \tau) := \mathsf{Sign}(\mathsf{sk}, m||\tau||\mathcal{F})$ and $\mathsf{Verify}_\mathcal{F}(\mathsf{pk}, m, \sigma, \tau) := \mathsf{Verify}(\mathsf{pk}, m||\tau||\mathcal{F}, \sigma)$. Then Combine and Verify$'$ can be defined as follows:

$\mathsf{Combine}((\mathsf{pk}_i)_{i\in[n]}, (m_i)_{i\in[n]}, f, (\sigma_i)_{i\in[n]}, \tau)$: If $f \notin \mathcal{F}$ return $\perp$. Otherwise, compute $\hat{\sigma} \leftarrow ((\mathsf{pk}_i, m_i, \sigma_i))_{i\in[n]}$ and $\hat{\mathsf{pk}} \leftarrow \prod_{i=1}^n \mathsf{pk}_i$ and return $\hat{\mathsf{pk}}$ and $\hat{\sigma}$.

$\mathsf{Verify}'(\hat{\mathsf{pk}}, \hat{m}, f, \hat{\sigma}, \tau)$: Return 1, if $(\mathsf{Verify}_\mathcal{F}(\mathsf{pk}_i, m_i, \sigma_i, \tau) = 1)_{i\in[n]} \ \wedge\ \hat{m} = f(m_1, \ldots, m_n) \ \wedge\ \hat{\mathsf{pk}} = \prod_{i=1}^n \mathsf{pk}_i \ \wedge\ f \in \mathcal{F}$, and 0 otherwise.

It is immediate that correctness holds. For unforgeability, note that since $\mathsf{Verify}'(\hat{\mathsf{pk}}^\star, \hat{m}^\star, f^\star, \hat{\sigma}^\star, \tau^\star) = 1$ by definition, we know that $\hat{\mathsf{pk}} = \prod_{i\in[n]} \mathsf{pk}_i$, where $(\mathsf{pk}_i)_{i\in[n]}$ is contained in the signature. Thus, we can simply engage with an EUF-CMA challenger to obtain pk and simulate the game without knowing sk by using the Sign oracle provided by the EUF-CMA challenger. If the adversary eventually outputs a forgery, we either have an EUF-CMA forgery which happens with negligible probability or a message $\hat{m}^\star \notin \mathsf{R}(f^\star)$ which happens with probability 0 as Verify$'$ does not accept such an input. Thus, the overall success probability of any PPT adversary is negligible. □

While this proves the existence of key succinct multikey-homomorphic signatures, one could also ask for signature succinctness as defined below.

Definition 32 (Signature Succinctness). *A multikey-homomorphic signature scheme is called signature succinct, if for all $\kappa \in \mathbb{N}$, for all $n \leq \mathsf{poly}(\kappa)$,* *for all* $\mathsf{PP} \leftarrow \mathsf{PGen}(1^\kappa)$*, for all* $((\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{PP}))_{i\in[n]}$*, for all* $(m_i)_{i\in[n]} \in \mathcal{M}^n$*, all* $(\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}_i, m_i))_{i\in[n]}$*, all* $(\hat{\mathsf{pk}}, \hat{\sigma}) \leftarrow \mathsf{Combine}((\mathsf{pk}_i)_{i\in[n]}, (m_i)_{i\in[n]}, f, (\sigma_i)_{i\in[n]})$ *it holds that*

$$|\hat{\sigma}| \leq \mathsf{poly}(\kappa).$$

It seems non-trivial to construct schemes that satisfy both succinctness definitions, i.e., provide succinct keys *and* signatures. We suspect, that this requires a trick à la Dodis et al. [DKNS04] which was used in the context of constructing constant-size ring signatures. Dodis et al. argue that the sets of keys (rings) will often be determined a priori in practice and can thus be represented by some compact description. Consequently, only a compact description of such a set of keys is required in the actual signature and the linear dependency on the ring size is removed. A similar argumentation also holds in many application scenarios for multi-key homomorphic signatures (e.g., scenarios where the joint verification key $\hat{\mathsf{pk}}$ is already pre-distributed as also discussed below).

Finally, one could also define a notion in the vein of function privacy in the context of functional signatures [BGI14], i.e., although $\mathsf{Combine}$ takes a function f, the output of $\mathsf{Combine}$ would be required to be indistinguishable for any f' that evaluates to the same output on the same input. Ultimately, one could even ask for a stronger property requiring that the signatures output by $\mathsf{Combine}$ look identical to signatures produced by Sign.

A.2 Discussion

We consider it to be interesting to find constructions of the various flavors of multikey-homomorphic signatures discussed above. It seems that using indistinguishability obfuscation in similar fashion as it is done in the context of universal signature aggregators [HKW15] is a viable direction to obtain signature succinctness. However, as the focus in this paper lies on key-homomorphisms, we leave a thorough investigation as future work. Subsequently, we informally discuss some further observations.

Related Concepts. Firstly, it seems that our notions are related to the properties one would expect from aggregate signatures [BGLS03] and the related notion of screening [BGR98]. Furthermore, they also seem to be related to batch verification of signatures [CHP12] and the recent notion of universal signature aggregators [HKW15].

Application to Delegation of Computation. Secondly, the concept of multi-key-homomorphic signatures seem to be a very interesting concept in the domain of verifiable delegation of computation on outsourced data.

Let us recall that homomorphic signatures for a class $\mathcal{F}$ can be used to certify computations on signed data for any $f \in \mathcal{F}$. Assume that some entity who holds a data set $(m_1, \ldots, m_n)$, is in possession of a secret key sk and produces signatures $(\sigma_1, \ldots, \sigma_n)$ for each respective message in the data set. Then, she can outsource the authenticated data set $(m_1, \sigma_1), \ldots, (m_n, \sigma_n)$ to some remote server (e.g., the cloud). Later, for any function $f \in \mathcal{F}$, the server can be asked to compute $\hat{m} = f(m_1, \ldots, m_n)$ and is able to deliver a succinct proof (signature) $\hat{\sigma}$ certifying the correctness of the computation. Anyone, given the public key pk of the data holder, the result $\hat{m}$, corresponding signature $\hat{\sigma}$ and the function f, can then verify whether the computation by the server has been performed correctly without needing to know the original data.

Now, there are many scenarios with many different signers each of them holding a distinct secret key sk_i and each of them periodically authenticates some data item $m_{i,j}$ and sends it to a server. Then, the server could compute a function f over inputs authenticated by different secret keys. Think for instance of environmental sensors that periodically send authenticated measurements to a server and this server can then compute on these authenticated measurements. The result can then be verified under the respective public keys or in case of a scheme with key succinctness the results are verifiable for anyone under a compact public key $\hat{\mathsf{pk}}$ (which can be computed from all the single public keys once and pre-distributed). Consequently, the concept of multikey-homomorphic signatures seems to be an interesting and viable direction for extending the scope of verifiable delegation of computation on outsourced data based on signatures.

B Examples of Key-Homomorphic Signature Schemes

Subsequently we give some examples of signature schemes providing key-homomorphic properties. Therefore let BGGen be a bilinear group generator which on input of a security parameter 1^κ and a type parameter $\mathsf{t} \in \{1, 2, 3\}$ outputs a bilinear group description BG. If $\mathsf{t} = 2$, BG is defined as $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g, \tilde{g}, \psi)$, where $\mathbb{G}_1 = \langle g \rangle, \mathbb{G}_2 = \langle \tilde{g} \rangle$, and $\mathbb{G}_T$ are three groups of prime order p with $\kappa = \log_2 p$, e is a bilinear map $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, and ψ is an isomorphism $\mathbb{G}_2 \to \mathbb{G}_1$. If $\mathsf{t} = 3$ the isomorphism ψ is missing. If $\mathsf{t} = 1$ we have that $\mathbb{G}_1 = \mathbb{G}_2$ denoted as $\mathbb{G}$.

B.1 Schnorr Signatures [Sch91]

In Scheme 5 we recall the Schnorr signature scheme.

$\mathsf{PGen}(1^\kappa)$: Choose a group $\mathbb{G}$ of prime order p with $\kappa = \log_2 p$, an elements $g \xleftarrow{R} \mathbb{G}$, and a hash function $H : \mathbb{G} \times \mathcal{M} \to \{0,1\}^n$ uniformly at random from hash function family $\{H_k\}_k$. Set and return $\mathsf{PP} \leftarrow (\mathbb{G}, g, H)$.

$\mathsf{KeyGen}(\mathsf{PP})$: Parse PP as $(\mathbb{G}, g, H)$, choose $x \xleftarrow{R} \mathbb{Z}_p$, set $\mathsf{pk} \leftarrow (\mathsf{PP}, g^x)$, $\mathsf{sk} \leftarrow (\mathsf{pk}, x)$ and output $(\mathsf{sk}, \mathsf{pk})$.

$\mathsf{Sign}(\mathsf{sk}, m)$: Parse sk as x, choose $r \xleftarrow{R} \mathbb{Z}_p$, compute $R \leftarrow g^r$, $c \leftarrow H(R, m)$, $y \leftarrow r + x \cdot c \bmod p$, and output $\sigma \leftarrow (c, y)$

$\mathsf{Verify}(\mathsf{pk}, m, \sigma)$: Parse pk as (PP, g^x) and σ as (c, y), verify whether $c = H((g^x)^{-c} g^y, m)$ and output 1 if so and 0 otherwise.

Scheme 5: Schnorr Signatures

Lemma 3. *Schnorr signatures are adaptable according to Definition 15.*

Proof. We prove the lemma above by presenting an Adapt algorithm satisfying the perfect adaptability notion.

$\mathsf{Adapt}(\mathsf{pk}, m, \sigma, \Delta)$: Let $\Delta \in \mathbb{Z}_p$ and $\mathsf{pk} = (\textsc{pp}, g^x)$. Return (pk', σ'), where $\mathsf{pk}' \leftarrow (\textsc{pp}, g^x \cdot g^{\Delta})$ and $\sigma' \leftarrow (c, y')$ with $y' \leftarrow y + c \cdot \Delta \bmod p$.

It is immediate that adapted signatures are identical to fresh signatures under $\mathsf{pk}' = (\textsc{pp}, g^{x+\Delta})$ as long as the initial signature is unknown. □

B.2 BLS Signatures [BLS04]

In Scheme 6 we recall BLS signatures in a Type 3 setting (cf. [CHKM10] for a treatment of security of this BLS variant). We stress that the properties which we discuss below are equally valid for the original BLS scheme in [BLS04] instantiated in a Type 2 setting.

$\mathsf{PGen}(1^\kappa)$: Run $\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\kappa, 3)$, choose a hash function $H : \mathcal{M} \rightarrow \mathbb{G}_1$ uniformly at random from hash function family $\{H_k\}_k$, set $\textsc{pp} \leftarrow (\mathsf{BG}, H)$.
$\mathsf{KeyGen}(\textsc{pp})$: Parse $\textsc{pp}$ as (BG, H), choose $x \xleftarrow{R} \mathbb{Z}_p$, set $\mathsf{pk} \leftarrow (\textsc{pp}, \tilde{g}^x)$, $\mathsf{sk} \leftarrow (\mathsf{pk}, x)$, and return $(\mathsf{sk}, \mathsf{pk})$.
$\mathsf{Sign}(\mathsf{sk}, m)$: Parse sk as x and return $\sigma \leftarrow H(m)^x$.
$\mathsf{Verify}(\mathsf{pk}, m, \sigma)$: Parse pk as $(\textsc{pp}, \tilde{g}^x)$, verify whether $e(H(m), \tilde{g}^x) = e(\sigma, \tilde{g})$ and return 1 if so and 0 otherwise.

Scheme 6: Type 3 BLS Signatures

Lemma 4. *BLS signatures are perfectly adaptable according to Definition 16.*

Proof. We prove the lemma above by presenting an Adapt algorithm satisfying the perfect adaptability notion.

$\mathsf{Adapt}(\mathsf{pk}, m, \sigma, \Delta)$: Let $\Delta \in \mathbb{Z}_p$ and $\mathsf{pk} = (\textsc{pp}, \tilde{g}^x)$. Return (pk', σ'), where $\mathsf{pk}' \leftarrow (\textsc{pp}, \tilde{g}^x \cdot \tilde{g}^{\Delta})$ and $\sigma' \leftarrow \sigma \cdot H(m)^{\Delta}$.

It is immediate that adapted signatures are identical to fresh signatures under $\mathsf{pk}' = (\textsc{pp}, \tilde{g}^{x+\Delta})$. □

Lemma 5. *BLS signatures are publicly key-homomorphic according to Definition 17.*

Proof. We prove the lemma above by presenting a suitable $\mathsf{Combine}$ algorithm.

$\mathsf{Combine}((\mathsf{pk}_i)_{i=1}^n, m, (\sigma_i)_{i=1}^n)$: Let $\mathsf{pk}_i = (\textsc{pp}, \tilde{g}^{x_i})$. Run $\hat{\mathsf{pk}} \leftarrow (\textsc{pp}, \prod_{i=1}^n \tilde{g}^{x_i})$, and $\hat{\sigma} \leftarrow \prod_{i=1}^n \sigma_i$ and return $\hat{\mathsf{pk}}$ and $\hat{\sigma}$. □

B.3 Katz-Wang Signatures [KW03, GJKW07]

Katz and Wang in [KW03] presented a signature scheme that enjoys a tight security reduction to the DDH problem. Basically, the public key of the scheme represents a Diffie-Hellman (DH) tuple and the signature is a non-interactive zero-knowledge proof (obtained using the Fiat-Shamir heuristic) that the public key indeed forms a DH tuple. We present the version from [GJKW07] (Section 4) in Scheme 7, which in contrast to the original one in [KW03] does not include the statement (public key) in the Fiat-Shamir transform.

$\mathsf{PGen}(1^\kappa)$: Choose a group $\mathbb{G}$ of prime order p with $\kappa = \log_2 p$, two elements $g, h \overset{R}{\leftarrow} \mathbb{G}$ and a hash function $H : \mathbb{G} \times \mathbb{G} \times \mathcal{M} \rightarrow \{0,1\}^n$ uniformly at random from hash function family $\{H_k\}_k$. Set and return $\mathsf{PP} \leftarrow (\mathbb{G}, g, h, H)$.

$\mathsf{KeyGen}(\mathsf{PP})$: Parse PP as $(\mathbb{G}, g, h, H)$, choose $x \overset{R}{\leftarrow} \mathbb{Z}_p$, set $\mathsf{pk} \leftarrow (\mathsf{PP}, g^x, h^x)$, $\mathsf{sk} \leftarrow (\mathsf{pk}, x)$, and return $(\mathsf{sk}, \mathsf{pk})$.

$\mathsf{Sign}(\mathsf{sk}, m)$: Parse sk as x, choose $r \overset{R}{\leftarrow} \mathbb{Z}_p$, set $A \leftarrow g^r$, $B \leftarrow h^r$, $c \leftarrow H(A, B, m)$, compute $s \leftarrow cx + r \bmod p$ and return $\sigma \leftarrow (c, s)$.

$\mathsf{Verify}(\mathsf{pk}, m, \sigma)$: Parse pk as (PP, y_1, y_2) and σ as (c, s) with $c \in \{0,1\}^n$ and $s \in \mathbb{Z}_p$. Compute $A \leftarrow g^s y_1^{-c}$, $B \leftarrow h^s y_2^{-c}$ and return 1 if $c = H(A, B, m)$ and 0 otherwise.

Scheme 7: Katz-Water Signatures.

Lemma 6. *Katz-Wang signatures are adaptable according to Definition 15.*

Proof. We prove the lemma above by presenting an Adapt algorithm satisfying the adaptability notion.

$\mathsf{Adapt}(\mathsf{pk}, m, \sigma, \Delta)$: Let $\Delta \in \mathbb{Z}_p$, $\mathsf{pk} = (\mathsf{PP}, y_1, y_2)$ and $\sigma = (c, s)$. Return (pk', σ'), where $\mathsf{pk}' \leftarrow (\mathsf{PP}, y_1 \cdot g^\Delta, y_2 \cdot h^\Delta)$ and $\sigma' \leftarrow (c, s + c\Delta \pmod p)$.

It is immediate that adapted signatures are identical to fresh signatures under $\mathsf{pk}' = (\mathsf{PP}, y_1 \cdot g^\Delta, y_2 \cdot h^\Delta)$ as long as the initial signature is unknown. $\square$

B.4 Waters Signatures [Wat05]

Below we recall Waters signatures with shared hashing parameters in the Type-3 bilinear group setting as used in [BFG13] (a similar variant is presented in [CHKM10]). We note that for Waters' signatures without shared hash parameters [Wat05] it seems to be impossible to define an Adapt algorithm satisfying Definition 14.

Lemma 7. *Waters signatures with shared hash parameters are perfectly adaptable according to Definition 16.*

Proof. We prove the lemma above by presenting an Adapt algorithm satisfying the perfect adaptability notion.

$\mathsf{PGen}(1^\kappa)$: Run $\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\kappa, 3)$, choose $U = (u_0, \ldots u_n) \xleftarrow{R} \mathbb{G}_1^k$, and define $H : \mathcal{M} \to \mathbb{G}_1$ as $H(m) \coloneqq u_0 \cdot \prod_{i=1}^n u_i^{m_i}$, where $\mathcal{M} = \{0,1\}^n$. Set $\textsf{PP} \leftarrow (\mathsf{BG}, U, H)$.
$\mathsf{KeyGen}(\textsf{PP})$: Parse $\textsf{PP}$ as (BG, U, H), choose $x \xleftarrow{R} \mathbb{Z}_p$, set $\mathsf{pk} \leftarrow (\textsf{PP}, \tilde{g}^x)$, $\mathsf{sk} \leftarrow (\mathsf{pk}, x)$, and return $(\mathsf{sk}, \mathsf{pk})$.
$\mathsf{Sign}(\mathsf{sk}, m)$: Parse sk as (pk, x), choose $r \xleftarrow{R} \mathbb{Z}_p$, set $\alpha \leftarrow g^x \cdot H(m)^r$, $\beta \leftarrow \tilde{g}^r, \gamma \leftarrow g^r$ and return $\sigma \leftarrow (\alpha, \beta, \gamma)$.
$\mathsf{Verify}(\mathsf{pk}, m, \sigma)$: Parse pk as $(\textsf{PP}, \tilde{g}^x)$ and σ as (α, β, γ). Verify whether $e(\alpha, \tilde{g}) = e(g, \tilde{g}^x) \cdot e(H(m), \beta) \ \wedge \ e(\gamma, \tilde{g}) = e(g, \beta)$ and return 1 if it holds and 0 otherwise.

Scheme 8: Waters Signatures with Shared Hash Parameters

$\mathsf{Adapt}(\mathsf{pk}, m, \sigma, \Delta)$: Let $\Delta \in \mathbb{Z}_p$ and compute $\Delta_1 \leftarrow g^\Delta$ and $\Delta_2 \leftarrow \tilde{g}^\Delta$. Otherwise let $\sigma = (\alpha, \beta, \gamma)$, and $\mathsf{pk} = (\textsf{PP}, \tilde{g}^x)$. Choose $r' \xleftarrow{R} \mathbb{Z}_p$, compute $\sigma' \leftarrow (\alpha \cdot \Delta_1 \cdot H(m)^{r'}, \beta \cdot \tilde{g}^{r'}, \gamma \cdot g^{r'})$ and $\mathsf{pk}' \leftarrow (\textsf{PP}, \tilde{g}^x \cdot \Delta_2)$.

Signatures output by Adapt are identically distributed as fresh signatures under randomness $r + r'$ und key $\mathsf{pk} = (\textsf{PP}, \tilde{g}^x \cdot \Delta_2,)$, which proves the lemma. $\square$

When instantiating our argument system from Section 4.3 with Groth-Sahai proofs, it is beneficial to use $\mathsf{sk} \leftarrow (\mathsf{pk}, g^x, \tilde{g}^x)$ as secret key. The associated secret key space would then be all tuples $(\Delta_1, \Delta_2) \in \mathbb{H} \subset \mathbb{G}_1 \times \mathbb{G}_2$ where $e(\Delta_1, \tilde{g}) = e(g, \Delta_2)$ and also $\Delta \in \mathbb{H}$. This is favourable regarding the extractability properties of the Groth-Sahai proof system.

Lemma 8. *Waters signatures are publicly key-homomorphic according to Definition 17.*

Proof. We prove the lemma above by presenting a suitable $\mathsf{Combine}$ algorithm.

$\mathsf{Combine}((\mathsf{pk}_i)_{i=1}^n, m, (\sigma_i)_{i=1}^n)$: Let $\sigma_i = (\alpha_i, \beta_i, \gamma_i)$ and $\mathsf{pk}_i = (\textsf{PP}, \tilde{g}^{x_i})$. Run $\hat{\mathsf{pk}} \leftarrow (\textsf{PP}, \prod_{i=1}^n \tilde{g}^{x_i}))$ and $\hat{\sigma} \leftarrow (\prod_{i=1}^n \alpha_i, \prod_{i=1}^n \beta_i, \prod_{i=1}^n \gamma_i)$ and return $\hat{\mathsf{pk}}$ and $\hat{\sigma}$. $\square$

B.5 PS Signatures [PS16b]

In Scheme 9 we recall a recent signature scheme from [PS16b], which provides perfect adaption, but is not publicly key-homomorphic.

$\mathsf{PGen}(1^\kappa)$: Run $\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\kappa, 3)$ set $\textsf{PP} \leftarrow \mathsf{BG}$.
$\mathsf{KeyGen}(\textsf{PP})$: Parse $\textsf{PP}$ as BG, choose $x, y \xleftarrow{R} \mathbb{Z}_p$, compute $\tilde{X} \leftarrow \tilde{g}^x$, $\tilde{Y} \leftarrow \tilde{g}^y$ and set $\mathsf{pk} \leftarrow (\textsf{PP}, \tilde{X}, \tilde{Y})$, $\mathsf{sk} \leftarrow (\mathsf{pk}, x, y)$, and return $(\mathsf{sk}, \mathsf{pk})$.
$\mathsf{Sign}(\mathsf{sk}, m)$: Parse sk as (pk, x, y), choose $h \xleftarrow{R} \mathbb{G}_1^*$ and return $\sigma \leftarrow (h, h^{(x+y\cdot m)})$.
$\mathsf{Verify}(\mathsf{pk}, m, \sigma)$: Parse pk as $(\textsf{PP}, \tilde{X}, \tilde{Y})$ and σ as (σ_1, σ_2). Check whether $\sigma_1 \neq 1_{\mathbb{G}_1}$ and $e(\sigma_1, \tilde{X} \cdot \tilde{Y}^m) = e(\sigma_2, \tilde{g})$ holds. If both checks hold return 1 and 0 otherwise.

Scheme 9: PS Signatures

Lemma 9. *PS signatures are perfectly adaptable according to Definition 16.*

Proof. We prove the lemma above by presenting an Adapt algorithm satisfying the perfect adaptability notion.

$\mathsf{Adapt}(\mathsf{pk}, m, \sigma, \Delta)$: Parse pk as $(\textsf{PP}, \tilde{X}, \tilde{Y})$, σ as (σ_1, σ_2) and Δ as $(\Delta_1, \Delta_2) \in \mathbb{Z}_p^2$ and choose $r \xleftarrow{R} \mathbb{Z}_p$. Compute $\mathsf{pk}' \leftarrow (\textsf{PP}, \tilde{X} \cdot \tilde{g}^{\Delta_1}, \tilde{Y} \cdot \tilde{g}^{\Delta_2})$ and $\sigma' \leftarrow (\sigma_1^r, (\sigma_2 \cdot \sigma_1^{\Delta_1 + \Delta_2 m})^r)$ and return (pk', σ').

The key $\mathsf{pk}' = (\tilde{g}^{x+\Delta_1}, \tilde{g}^{y+\Delta_2})$ and $\sigma' = (h^r, (h^r)^{x+\Delta_1+m(y+\Delta_2)})$ output by the Adapt algorithm is identically distributed to a fresh signature under randomness h^r and pk'. □

It is easy to see, that PS signatures are, however, not publicly key-homomorphic as independently generated signatures are computed with respect to different bases h with unknown discrete logarithms. Consequently, there is no efficient means to obtain a succinct representation of $\hat{\sigma}$ that is suitable for Verify.

B.6 CL Signature Variant [CHP12]

While the original pairing-based CL signature scheme [CL04] does not satisfy any of the key-homomorphic properties discussed in this paper, we recall a CL signature variant from [CHP12] in Scheme 10 which does.

$\mathsf{PGen}(1^\kappa)$: Run $\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\kappa, 1)$, choose some polynomially bound set Ψ and hash functions $H_1 : \Psi \to \mathbb{G}$, $H_2 : \Psi \to \mathbb{G}$, $H_3 : \mathcal{M} \times \Psi \to \mathbb{Z}_p$ uniformly at random from suitable hash function families. Set $\textsf{PP} \leftarrow (\mathsf{BG}, H_1, H_2, H_3)$.

$\mathsf{KeyGen}(\textsf{PP})$: Parse $\textsf{PP}$ as $(\mathsf{BG}, H_1, H_2, H_3)$, choose $x \xleftarrow{R} \mathbb{Z}_p$ and set $\mathsf{pk} \leftarrow (\textsf{PP}, g^x)$, $\mathsf{sk} \leftarrow (\mathsf{pk}, x)$, and return $(\mathsf{sk}, \mathsf{pk})$.

$\mathsf{Sign}(\mathsf{sk}, (m, \psi))$: If it is the first call to Sign during time period $\psi \in \Psi$, then parse sk as (pk, x), compute $w \leftarrow H_3(m, \psi)$, $a \leftarrow H_1(\psi)$, $b \leftarrow H_2(\psi)$ and return $\sigma \leftarrow a^x b^{xw}$. Otherwise abort.

$\mathsf{Verify}(\mathsf{pk}, (m, \psi), \sigma)$: Parse pk as $(\textsf{PP}, X)$ and compute $w \leftarrow H_3(m, \psi)$, $a \leftarrow H_1(\psi)$, $b \leftarrow H_2(\psi)$ and check whether $e(\sigma, g) = e(a, X) \cdot e(b, X)^w$ holds. If so return 1 and 0 otherwise.

Scheme 10: CL Signature Variant

Lemma 10. *Adapted CL signatures are perfectly adaptable according to Definition 16.*

Proof. We prove the lemma above by presenting an Adapt algorithm satisfying the perfect adaptability notion.

$\mathsf{Adapt}(\mathsf{pk}, (m, \psi), \sigma, \Delta)$: Parse pk as $(\textsf{PP}, X)$ and compute $w \leftarrow H_3(m, \psi)$, $a \leftarrow H_1(\psi)$, $b \leftarrow H_2(\psi)$. Compute $\mathsf{pk}' \leftarrow (\textsf{PP}, X \cdot g^\Delta)$ and $\sigma' \leftarrow \sigma \cdot a^\Delta \cdot b^{\Delta \cdot w}$ and return (pk', σ').

It is easy to see that adapted signatures are identical to fresh signatures under $\mathsf{pk}' = (\mathsf{PP}, X \cdot g^{\Delta})$. □

Lemma 11. *Adapted CL signatures are publicly key-homomorphic according to Definition 17.*

Proof. We prove the lemma above by presenting a suitable $\mathsf{Combine}$ algorithm.

$\mathsf{Combine}((\mathsf{pk}_i)_{i=1}^n, m, (\sigma_i)_{i=1}^n)$: Let $\mathsf{pk}_i = (\mathsf{PP}, g^{x_i})$. Run $\hat{\mathsf{pk}} \leftarrow (\mathsf{PP}, \prod_{i=1}^n g^{x_i}$ and $\hat{\sigma} \leftarrow (\prod_{i=1}^n \sigma_i)$ and return $\hat{\mathsf{pk}}$ and $\hat{\sigma}$. □

C Security Proofs

C.1 Proof of Theorem 1

We show that Theorem 1 holds by proving the subsequent lemmas.

Lemma 12. *If Σ is correct, and Π is complete, then Scheme 1 is correct.*

Lemma 12 follows from inspection and the proof is therefore omitted.

Lemma 13. *If Σ is* EUF-CMA *secure, and provides adaptability of signatures, and Π is witness indistinguishable, then Scheme 1 is unforgeable.*

Proof. We prove unforgeability using a sequence of games where we let $q_{\mathsf{s}} \leq \mathsf{poly}(\kappa)$ be the number of Sign queries.

Game 0: The original unforgeability game.
Game 1: As Game 0, but upon setup we store csk and simulate Sign using the following modified algorithm Sign', which additionally takes csk as input:
$\mathsf{Sign}'(\mathsf{PP}, \mathsf{sk}_i, m, \mathcal{R}, \boxed{\mathsf{csk}})$: Parse PP as $(1^\kappa, \mathsf{crs}, \mathsf{cpk})$ and return $\bot$ if $\mu(\mathsf{sk}_i) \notin \mathcal{R}$. Otherwise, return $\sigma \leftarrow (\delta, \mathsf{pk}, \pi)$, where

$$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\kappa),\ \delta \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}, m||\mathcal{R}),\ \text{and}$$
$$\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathsf{pk}, \mathsf{cpk}, \mathcal{R}), (\boxed{\mathsf{csk}} - \mathsf{sk})).$$

Transition - Game 0 → Game 1: A distinguisher between $\mathcal{D}^{0\to1}$ is a distinguisher for adaptive witness indistinguishability of Π, i.e., $|\Pr[S_0] - \Pr[S_1]| \leq \varepsilon_{\mathsf{wi}}(\kappa)$.
Game 2: As Game 1, but instead of generating crs upon setup, we obtain $(\mathsf{crs}, \xi) \leftarrow \Pi.\mathsf{E}_1(1^\kappa)$ and store ξ.
Transition - Game 1 → Game 2: A distinguisher between Game 1 and 2 distinguishes an honest crs from an extraction crs, i.e., $|\Pr[S_1] - \Pr[S_2]| \leq \varepsilon_{\mathsf{e1}}(\kappa)$.
Game 3: As Game 2, but whenever the adversary outputs a forgery $(m^\star, \sigma^\star, \mathcal{R}^\star)$, where $\sigma^\star = (\delta^\star, \mathsf{pk}^\star, \pi^\star)$ we extract a witness $\mathsf{sk}' \leftarrow \Pi.\mathsf{E}_2(\mathsf{crs}, \xi, (\mathsf{pk}^\star, \mathsf{cpk}, \mathcal{R}^\star), \pi^\star)$ and abort if the extractor fails.
Transition - Game 2 → Game 3: Game 2 and Game 3 proceed identically, unless the extractor fails, i.e., $|\Pr[S_2] - \Pr[S_3]| \leq \varepsilon_{\mathsf{e2}}(\kappa)$.
Game 4: As Game 3, but we further modify Sign' as follows:

$\mathsf{Sign}'(\textsc{pp}, \boxed{i}, m, \mathcal{R}, \mathsf{csk})$: Parse $\textsc{pp}$ as $(1^\kappa, \mathsf{crs}, \mathsf{cpk})$ and return $\perp$ if $\mathsf{pk}_i \notin \mathcal{R}$. Otherwise, return $\sigma \leftarrow (\delta, \mathsf{pk}, \pi)$, where

$$\boxed{\mathsf{sk} \xleftarrow{R} \mathbb{H},\ \delta' \leftarrow \Sigma.\mathsf{Sign}(\mathsf{csk}, m||\mathcal{R})},$$
$$\boxed{(\mathsf{pk}, \delta) \leftarrow \Sigma.\mathsf{Adapt}(\mathsf{cpk}, m||\mathcal{R}, \delta', -\mathsf{sk})},\ \text{and}$$
$$\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathsf{pk}, \mathsf{cpk}, \mathcal{R}), (\boxed{\mathsf{sk}})).$$

Transition - Game 3 → Game 4: Under adaptability of signatures, this game change is conceptual, i.e., $\Pr[S_3] = \Pr[S_4]$.

Game 5: As Game 4, but we abort whenever we extract an sk' so that $\mathsf{cpk} = \mathsf{pk} \cdot \mu(\mathsf{sk}')$.

Transition - Game 4 → Game 5: Game 4 and Game 5 proceed identical, unless abort event E_1 happens. For the sake of contradiction assume that E_1 occurs with non-negligible probability. Then we can engage with an EUF-CMA challenger $\mathcal{C}_\kappa^{\mathsf{f}}$ to obtain cpk upon setup and simulate Sign as follows:

$\mathsf{Sign}'(\textsc{pp}, i, m, \mathcal{R}, \boxed{\perp})$: Parse $\textsc{pp}$ as $(1^\kappa, \mathsf{crs}, \mathsf{cpk})$ and return $\perp$ if $\mathsf{pk}_i \notin \mathcal{R}$. Otherwise, return $\sigma \leftarrow (\delta, \mathsf{pk}, \pi)$, where

$$\mathsf{sk} \xleftarrow{R} \mathbb{H},\ \boxed{\delta' \leftarrow \mathcal{C}_{\mathsf{f}}^{\kappa}.\mathsf{Sign}(m||\mathcal{R})},$$
$$(\mathsf{pk}, \delta) \leftarrow \Sigma.\mathsf{Adapt}(\mathsf{cpk}, m||\mathcal{R}, \delta', -\mathsf{sk}),\ \text{and}$$
$$\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathsf{pk}, \mathsf{cpk}, \mathcal{R}), (\mathsf{sk})).$$

Now, whenever E_1 happens, we use the forgery $(m^\star, \sigma^\star, \mathcal{R}^\star)$, where $\sigma^\star = (\delta^\star, \mathsf{pk}^\star, \pi^\star)$ to obtain $(\mathsf{cpk}, \delta) \leftarrow \mathsf{Adapt}(\mathsf{pk}, m^\star||\mathcal{R}^\star, \delta^\star, \mathsf{sk}')$ and return $(m^\star||\mathcal{R}^\star, \delta)$ as an EUF-CMA forgery to $\mathcal{C}_{\mathsf{f}}^{\kappa}$ with probability $\Pr[E_1]$. That is, $|\Pr[S_4] - \Pr[S_5]| \leq \epsilon_{\mathsf{f}}(\kappa)$.

Game 6: As Game 5, but we guess the index $i^\star$ the adversary will attack at the beginning of the game, and abort if our guess is wrong.

Transition - Game 5 → Game 6: The success probability in Game 5 is the same as in Game 6, unless our guess is wrong, i.e., $\Pr[S_6] = \frac{1}{\mathsf{poly}(\kappa)} \cdot \Pr[S_5]$.

Game 7: As Game 6, but instead of running KeyGen for user $i^\star$, we engage with an EUF-CMA challenger of Σ to obtain $\mathsf{pk}_{i^\star}$.

Transition - Game 6 → Game 7: This change is conceptual, i.e., $\Pr[S_6] = \Pr[S_7]$.

If the adversary outputs a forgery $(m^\star, \sigma^\star, \mathcal{R}^\star)$ in Game 7, we compute $(\mathsf{pk}_{i^\star}, \sigma_{i^\star}) \leftarrow \mathsf{Adapt}(\mathsf{pk}^\star, m^\star||\mathcal{R}^\star, \delta^\star, \mathsf{sk}')$ and return $(\sigma_{i^\star}, m^\star||\mathcal{R}^\star)$ as a valid forgery for Σ. That is, $\Pr[S_7] \leq \varepsilon_{\mathsf{f}}(\kappa)$ and we obtain $\Pr[S_0] \leq \mathsf{poly}(\kappa) \cdot \varepsilon_{\mathsf{f}}(\kappa) + \varepsilon_{\mathsf{wi}}(\kappa) + \varepsilon_{\mathsf{e1}}(\kappa) + \varepsilon_{\mathsf{e2}}(\kappa) + \varepsilon_{\mathsf{f}}(\kappa)$ as a bound for the success probability which concludes the proof. □

Lemma 14. *If Σ provides adaptability of signatures and Π is witness indistinguishable, then Scheme 1 is anonymous.*

Proof. We show that a simulation of the anonymity game for $b = 0$ is indistinguishable from a simulation of the anonymity game with $b = 1$.

Game 0: The anonymity game with $b = 0$.

Game 1: As Game 0, but instead of generating crs upon setup, we obtain crs from a witness indistinguishability challenger $\mathcal{C}_\kappa^{\mathsf{wi}}$ upon Setup.

Transition - Game 0 → Game 1: This change is conceptual, i.e., $\mathsf{Pr}[S_0] = \mathsf{Pr}[S_1]$.

Game 2: As Game 1, but instead of obtaining σ via Sign, we execute the following modified algorithm Sign', which, besides $\textsf{PP}$, m and $\mathcal{R}$, takes sk_0 and sk_1 as input:

$\mathsf{Sign}'(\textsf{PP}, \mathsf{sk}_0, \mathsf{sk}_1, m, \mathcal{R})$: Parse $\textsf{PP}$ as $(1^\kappa, \mathsf{crs})$ and return $\perp$ if $\mu(\mathsf{sk}_0) \notin \mathcal{R}$ $\boxed{\vee\ \mu(\mathsf{sk}_1) \notin \mathcal{R}}$. Otherwise, return $\sigma \leftarrow (\delta, \mathsf{pk}, \pi)$, where

$$(\mathsf{sk}, \mathsf{pk}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa),\ \ \delta \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}, m||\mathcal{R}),\ \text{ and}$$
$$\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathsf{pk}, \mathcal{R}), (\boxed{\mathsf{sk}_1} - \mathsf{sk})).$$

Transition - Game 1 → Game 2: A distinguisher between $\mathcal{D}^{1\to2}$ is a distinguisher for adaptive witness indistinguishability of Π, i.e., $|\mathsf{Pr}[S_2] - \mathsf{Pr}[S_1]| \leq \varepsilon_{\mathsf{wi}}(\kappa)$.

In Game 2, we have a simulation for $b = 1$; $|\mathsf{Pr}[S_2] - \mathsf{Pr}[S_0]| \leq \varepsilon_{\mathsf{wi}}(\kappa)$, which proves the lemma. □

C.2 Proof of Theorem 2

We subsequently show that Theorem 2 holds where we note that if non-transferability privacy is sufficient, Σ only needs to be adaptable.

Lemma 15. *If Σ is correct, and Π is complete, then Scheme 2 is correct.*

Lemma 15 follows from inspection and the proof is therefore omitted.

Lemma 16. *If Σ is* EUF-CMA *secure and adapts signatures, f is a one-way function, and Π is witness indistinguishable, then Scheme 2 is simulation-sound DV-unforgeable.*

Proof. We followingly bound the success probability of an adversary using a sequence of games, where we let $q_{\mathsf{Sim}} \leq \mathsf{poly}(\kappa)$ be the number Sim queries.

Game 0: The original DV-unforgeability game.

Game 1: As Game 0, but inside the S oracle we execute the following modified Sim algorithm Sim', which additionally takes sk as input.

$\mathsf{Sim}'(\mathsf{pk}, \mathsf{vsk}, m, \boxed{\mathsf{sk}})$: Output $\delta = (\mathsf{pk}', \sigma_{\mathsf{R}}, \pi)$, where

$$(\mathsf{sk}_{\mathsf{R}}, \mathsf{pk}_{\mathsf{R}}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa),\ \ \mathsf{pk}' \leftarrow \mathsf{pk}_{\mathsf{R}} \cdot \mathsf{pk}^{-1},\ \ \sigma_{\mathsf{R}} \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}_{\mathsf{R}}, m),$$
$$\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathsf{pk}', f(\mathsf{vsk})), \boxed{(\mathsf{sk}_{\mathsf{R}} - \mathsf{sk}, \perp)}).$$

Transition - Game 0 → Game 1: A distinguisher between $\mathcal{D}^{0\to1}$ is a distinguisher for adaptive witness indistinguishability of Π, i.e., $|\mathsf{Pr}[S_0] - \mathsf{Pr}[S_1]| \leq \varepsilon_{\mathsf{wi}}(\kappa)$.

Game 2: As Game 1, but instead of generating crs upon PGen, we obtain $(\mathsf{crs}, \xi) \leftarrow \Pi.\mathsf{E}_1(1^\kappa)$ and store ξ.

Transition - Game 1 → Game 2: A distinguisher between Game 1 and 2 distinguishes an honest crs from an extraction crs, i.e., $|\Pr[S_1] - \Pr[S_2]| \leq \varepsilon_{\mathsf{e1}}(\kappa)$.

Game 3: As Game 2, but whenever the adversary outputs a forgery $(m^\star, \delta^\star)$, where $\delta^\star = (\mathsf{pk}'^\star, \sigma_\mathsf{R}^\star, \pi^\star)$ we extract a witness $(\mathsf{sk}'^\star, \mathsf{vsk}'^\star) \leftarrow \Pi.\mathsf{E}_2(\mathsf{crs}, \xi, (\mathsf{pk}'^\star, \mathsf{vpk}^\star), \pi^\star)$ and abort if the extractor fails.

Transition - Game 2 → Game 3: Game 2 and Game 3 proceed identically, unless the extractor fails, i.e., $|\Pr[S_1] - \Pr[S_2]| \leq \varepsilon_{\mathsf{e2}}(\kappa)$.

Game 4: As Game 3, but we further modify Sim' as follows:

```
$\mathsf{Sim}'(\mathsf{pk}, \mathsf{vsk}, m, \mathsf{sk})$ : Output $\delta = (\mathsf{pk}', \sigma_\mathsf{R}, \pi)$, where
    $\boxed{\sigma \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}, m)}$,
    $\boxed{(\mathsf{sk}', \mathsf{pk}') \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa),\ (\mathsf{pk}_\mathsf{R}, \sigma_\mathsf{R}) \leftarrow \Sigma.\mathsf{Adapt}(\mathsf{pk}, m, \sigma, \mathsf{sk}')}$,
    $\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathsf{pk}', f(\mathsf{vsk})), (\boxed{\mathsf{sk}'}, \perp))$.
```

Transition Game 3 → Game 4: Under adaptability of signatures, this change is conceptual and $\Pr[S_3] = \Pr[S_4]$.

Game 5: As Game 4, but instead of generating $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(\mathsf{PP}')$, we obtain pk from an $\mathsf{EUF\text{-}CMA}$ challenger. Further, whenever a signature under pk is required, we use the Sign oracle provided by the challenger.

Transition - Game 4 → Game 5: This change is conceptual, i.e., $\Pr[S_4] = \Pr[S_5]$.

Game 6: As Game 5, but we obtain vpk from a one-wayness challenger and set $\mathsf{vsk} = \perp$. In addition, we simulate the Vrfy oracle by using vpk instead of $f(\mathsf{vsk})$ inside the $\mathsf{DVerify}$ algorithm.

Transition - Game 5 → Game 6: This change is conceptual, i.e., $\Pr[S_5] = \Pr[S_6]$.

In Game 6, we we either have either extracted $\mathsf{vsk}^\star$ so that $f(\mathsf{vsk}^\star) = \mathsf{vpk}$ and we can output $\mathsf{vsk}^\star$ to the one-wayness challenger, or we have extracted $\mathsf{sk}'^\star$ such that $\mu(\mathsf{sk}'^\star) = \mathsf{pk}'^\star$ and can obtain $(\mathsf{pk}, \sigma) \leftarrow \Sigma.\mathsf{Adapt}(\mathsf{pk} \cdot \mathsf{pk}'^\star, m^\star, \sigma_\mathsf{R}^\star, -\mathsf{sk}'^\star)$ and output $(m^\star, \sigma)$ as a forgery for Σ. Taking the union bound yields $\Pr[S_6] \leq \varepsilon_\mathsf{f}(\kappa) + \varepsilon_\mathsf{ow}(\kappa)$, and we obtain $\Pr[S_0] \leq \varepsilon_\mathsf{f}(\kappa) + \varepsilon_\mathsf{ow}(\kappa) + \varepsilon_\mathsf{wi}(\kappa) + \varepsilon_\mathsf{e1}(\kappa) + \varepsilon_\mathsf{e2}(\kappa) +$ which is negligible. □

Lemma 17. *If Σ perfectly adapts signatures, and Π is witness indistinguishable, then Scheme 2 is strongly non-transferable private.*

Proof. We bound the success probability using a sequence of games.

Game 0: The original non-transferability privacy game.

Game 1: As Game 0, but instead of generating crs upon setup, we obtain crs from a witness indistinguishability challenger $\mathcal{C}_\kappa^{\mathsf{wi}}$ upon Setup.

Transition - Game 0 → Game 1: This change is conceptual, i.e., $\Pr[S_0] = \Pr[S_1]$.

Game 2: As Game 1, but inside SoD we execute the following modified the Desig algorithm Desig' which additionally takes vsk as input:

$\mathsf{Desig}'(\mathsf{pk}, \mathsf{vpk}, m, \sigma, \boxed{\mathsf{vsk}})$: Output $\delta \leftarrow (\mathsf{pk}', \sigma_\mathsf{R}, \pi)$, where

$$(\mathsf{sk}', \mathsf{pk}') \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa),\ (\mathsf{pk}_\mathsf{R}, \sigma_\mathsf{R}) \leftarrow \Sigma.\mathsf{Adapt}(\mathsf{pk}, m, \sigma, \mathsf{sk}'),$$
$$\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathsf{pk}', \mathsf{vpk}), \boxed{(\perp, \mathsf{vsk})}).$$

Transition - Game 1 → Game 2: A distinguisher between $\mathcal{D}^{1\to 2}$ is a distinguisher for adaptive witness indistinguishability of Π, i.e., $|\Pr[S_2] - \Pr[S_1]| \le \varepsilon_\mathsf{wi}(\kappa)$.

Game 3: As Game 2, but we further modify Desig' as follows:

$\mathsf{Desig}'(\mathsf{pk}, \mathsf{vpk}, m, \sigma, \mathsf{vsk})$: Output $\delta \leftarrow (\mathsf{pk}', \sigma_\mathsf{R}, \pi)$, where

$$\boxed{(\mathsf{sk}_\mathsf{R}, \mathsf{pk}_\mathsf{R}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa),\ \mathsf{pk}' \leftarrow \mathsf{pk}_\mathsf{R} \cdot \mathsf{pk}^{-1},\ \sigma_\mathsf{R} \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}_\mathsf{R}, m)},$$
$$\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathsf{pk}', \mathsf{vpk}), (\perp, \mathsf{vsk})).$$

Transition - Game 2 → Game 3: By the perfect adaption of signatures, this change is conceptual, i.e., $\Pr[S_2] = \Pr[S_3]$.

In Game 3, Desig' is identical to Sim. This means that SoD is simulated independently of b and $|\Pr[S_3] - \Pr[S_0]| \le \varepsilon_\mathsf{wi}(\kappa)$, which proves the lemma. □

C.3 Proof of Theorem 3

We show that Theorem 3 holds by proving the subsequent lemmas.

Lemma 18. *If Π is complete and Σ is correct, Π_sse is complete.*

The lemma above follows from inspection and the proof is therefore omitted.

Lemma 19. *If Π is witness indistinguishable and admits proofs of knowledge, and Σ provides a secret-key to public-key homomorphism, then Π_sse is zero-knowledge.*

Proof. We subsequently prove that zero-knowledge follows from witness indistinguishability.

Game 0: The zero-knowledge game, where we use the real $\mathsf{Proof}(\mathsf{crs}, \cdot, \cdot)$ algorithm on witnesses $(w, \perp)$ to reply to queries of the adversary.

Game 1: As Game 0, but we store csk upon Setup.

Transition Game 0 → Game 1: This change is conceptual, i.e., $\Pr[S_0] = \Pr[S_1]$.

Game 2: As Game 1, but use the following modified Proof algorithm Proof' which additionally takes csk as input:

$\mathsf{Proof}'(\mathsf{crs}, x, w, \boxed{\mathsf{csk}})$: Run $(\mathsf{sk}, \mathsf{pk}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa)$, $(\mathsf{sk}_\mathsf{ot}, \mathsf{pk}_\mathsf{ot}) \leftarrow \Sigma_\mathsf{ot}.\mathsf{KeyGen}(1^\kappa)$, and return $\pi \leftarrow (\pi_\Pi, \mathsf{pk}, \sigma, \mathsf{pk}_\mathsf{ot}, \sigma_\mathsf{ot})$, where

$$\pi_\Pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (x, \mathsf{cpk}, \mathsf{pk}), \boxed{(\perp, \mathsf{csk} - \mathsf{sk})}),\ \sigma \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}, \mathsf{pk}_\mathsf{ot}),\ \text{and}$$
$$\sigma_\mathsf{ot} \leftarrow \Sigma_\mathsf{ot}.\mathsf{Sign}(\mathsf{sk}_\mathsf{ot}, \pi_\Pi || x || \mathsf{pk} || \sigma).$$

Transition - Game 1 → Game 2: We present a hybrid game which shows that both games are indistinguishable under the witness indistinguishability of the argument system. First, we conceptually change the Setup algorithm to Setup' which obtains crs_Π from a witness indistinguishability challenger:

$\mathsf{Setup}(1^\kappa)$: Run $\boxed{\mathsf{crs}_\Pi \leftarrow \mathcal{C}^{\mathsf{wi}}_\kappa}$, $(\mathsf{csk}, \mathsf{cpk}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa)$, return $\mathsf{crs} \leftarrow (\mathsf{crs}_\Pi, \mathsf{cpk})$.

The change above is only conceptual. Furthermore, we use the following Proof'' algorithm instead of Proof':

$\mathsf{Proof}''(\mathsf{crs}, x, w, \boxed{\mathsf{csk}})$: Run $(\mathsf{sk}, \mathsf{pk}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa)$, $(\mathsf{sk}_{\mathsf{ot}}, \mathsf{pk}_{\mathsf{ot}}) \leftarrow \Sigma_{\mathsf{ot}}.\mathsf{KeyGen}(1^\kappa)$, and return $\pi \leftarrow (\pi_\Pi, \mathsf{pk}, \sigma, \mathsf{pk}_{\mathsf{ot}}, \sigma_{\mathsf{ot}})$, where

$$\boxed{\pi_\Pi \leftarrow \mathcal{C}^{\mathsf{wi}}_\kappa((x, \mathsf{cpk}, \mathsf{pk}), (w, \perp), (\perp, \mathsf{csk} - \mathsf{sk}))},\ \sigma \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}, \mathsf{pk}_{\mathsf{ot}}),\ \text{and}$$
$$\sigma_{\mathsf{ot}} \leftarrow \Sigma_{\mathsf{ot}}.\mathsf{Sign}(\mathsf{sk}_{\mathsf{ot}}, \pi_\Pi || x || \mathsf{pk} || \sigma).$$

Now depending of whether the challenger uses the first witness ($b = 0$) or the second witness ($b = 1$) we either simulate Game 1 or Game 2. More precisely, Proof'' produces the identical distribution as Proof if $b = 0$ and the identical distribution to Proof' if $b = 1$. That is $|\mathsf{Pr}[S_1] - \mathsf{Pr}[S_2]| \leq \epsilon_{\mathsf{wi}}(\kappa)$.

Game 3: As Game 2, but we further modify Proof' so that it no longer takes w as input:

$\mathsf{Proof}'(\mathsf{crs}, x, \mathsf{csk})$: Run $(\mathsf{sk}, \mathsf{pk}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa)$, $(\mathsf{sk}_{\mathsf{ot}}, \mathsf{pk}_{\mathsf{ot}}) \leftarrow \Sigma_{\mathsf{ot}}.\mathsf{KeyGen}(1^\kappa)$, and return $\pi \leftarrow (\pi_\Pi, \mathsf{pk}, \sigma, \mathsf{pk}_{\mathsf{ot}}, \sigma_{\mathsf{ot}})$, where

$$\pi_\Pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (x, \mathsf{cpk}, \mathsf{pk}), (\perp, \mathsf{csk} - \mathsf{sk})),\ \sigma \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}, \mathsf{pk}_{\mathsf{ot}}),\ \text{and}$$
$$\sigma_{\mathsf{ot}} \leftarrow \Sigma_{\mathsf{ot}}.\mathsf{Sign}(\mathsf{sk}_{\mathsf{ot}}, \pi_\Pi || x || \mathsf{pk} || \sigma).$$

Transition - Game 2 → Game 3: This change is conceptual, i.e., $\mathsf{Pr}[S_2] = \mathsf{Pr}[S_3]$.

Game 4: As Game 3, but instead of obtaining crs using Setup, we obtain $(\mathsf{crs}, \tau) \leftarrow \mathsf{S}_1$ (observe that $\tau = \mathsf{csk}$, so we still know csk). Now the setup is already as in the second distribution of the zero-knowledge game.

Transition - Game 3 → Game 4: A crs output by S_1 is indistinguishable from an honest crs under the CRS indistinguishability provided by the proof of knowledge property (observe that S_1 internally uses E_1 to obtain crs). Thus, $|\mathsf{Pr}[S_3] - \mathsf{Pr}[S_4]| \leq \varepsilon_{\mathsf{pok1}}(\kappa)$.

Game 5: As Game 4, but we further modify Proof' as follows:

$\mathsf{Proof}'(\mathsf{crs}, \boxed{\tau}, x)$: $\boxed{\text{Parse } \tau \text{ as } \mathsf{csk}}$. Run $(\mathsf{sk}, \mathsf{pk}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa)$, $(\mathsf{sk}_{\mathsf{ot}}, \mathsf{pk}_{\mathsf{ot}}) \leftarrow \Sigma_{\mathsf{ot}}.\mathsf{KeyGen}(1^\kappa)$, and return $\pi \leftarrow (\pi_\Pi, \mathsf{pk}, \sigma, \mathsf{pk}_{\mathsf{ot}}, \sigma_{\mathsf{ot}})$, where

$$\pi_\Pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (x, \mathsf{cpk}, \mathsf{pk}), (\perp, \mathsf{csk} - \mathsf{sk})),\ \sigma \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}, \mathsf{pk}_{\mathsf{ot}}),\ \text{and}$$
$$\sigma_{\mathsf{ot}} \leftarrow \Sigma_{\mathsf{ot}}.\mathsf{Sign}(\mathsf{sk}_{\mathsf{ot}}, \pi_\Pi || x || \mathsf{pk} || \sigma).$$

Now Proof' is equivalent to S_2.

Transition - Game 4 → Game 5: This change is conceptual, i.e., $\mathsf{Pr}[S_4] = \mathsf{Pr}[S_5]$.

In Game 0 we simulate the first distribution of the zero-knowledge game whereas in Game 5 we simulate the second distribution. We have that $|\mathsf{Pr}[S_0] - \mathsf{Pr}[S_5]| \leq \varepsilon_{\mathsf{wi}}(\kappa) + \varepsilon_{\mathsf{pok1}}(\kappa)$ which concludes the proof. □

Now, we have already established the existence of a simulator by proving zero-knowledge and can go on by proving simulation sound extractability.

Lemma 20. *If* Π *is witness indistinguishable and admits proof of knowledge and* Σ *is* EUF-CMA *secure and adapts signatures, then* Π_{sse} *is simulation sound extractable.*

Proof. We show that even when the adversary sees simulated proofs for arbitrary statements, we are still able to extract a witness w from a proof $\pi^\star$ for a statement $x^\star$ so that $R(x^\star, w) = 1$ as long as $(x^\star, \pi^\star)$ does not correspond to an query-answer pair of the simulation oracle. By Lemma 19, we know that $(\mathsf{S}_1, \mathsf{S}_2)$ is a suitable zero-knowledge simulator. In addition, we observe that the output of S is identical to S_1 when restricted to (crs, τ). This completes CRS indistinguishability part of the proof. To prove the second part of simulation sound extractability we proceed using a sequence of games where we let $q \leq \mathsf{poly}(\kappa)$ be the number of queries to the simulator.

Game 0: The original simulation sound extractability game.

Game 1: As Game 0, but we engage with an EUF-CMA challenger within S. That is, we execute the following modified S algorithm S':

$\mathsf{S}'(1^\kappa)$: Run $(\mathsf{crs}_\Pi, \xi) \leftarrow \Pi.\mathsf{E}_1(1^\kappa)$, $\mathsf{cpk} \leftarrow \mathcal{C}^{\mathsf{f}}_\kappa$, and return $(\mathsf{crs}, \tau, \xi)$, where

$$\mathsf{crs} \leftarrow (\mathsf{crs}_\Pi, \mathsf{cpk}) \text{ and } \tau \leftarrow \bot.$$

This also requires us to modify the S_2 algorithm used for simulation to obtain S_2'. Essentially, we leverage the adaptability of signatures to shift signatures obtained from the signing oracle provided by the EUF-CMA challenger under cpk to signatures under a random key. The "shift-amount" is then a valid witness for the relation.

$\mathsf{S}_2'(\mathsf{crs}, x)$: Obtain $\boxed{\mathsf{sk}' \xleftarrow{R} \mathbb{H},\ \mathsf{pk} \leftarrow \mathsf{cpk} \cdot \mu(\mathsf{sk}')}$. Further, run $(\mathsf{sk}_{\mathsf{ot}}, \mathsf{pk}_{\mathsf{ot}}) \leftarrow \Sigma_{\mathsf{ot}}.\mathsf{KeyGen}(1^\kappa)$, and return $\pi \leftarrow (\pi_\Pi, \mathsf{pk}, \sigma, \mathsf{pk}_{\mathsf{ot}}, \sigma_{\mathsf{ot}})$, where

$$\pi_\Pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (x, \mathsf{cpk}, \mathsf{pk}), (\bot, \boxed{-\mathsf{sk}'})),\ \boxed{\sigma' \leftarrow \mathcal{C}^{\mathsf{f}}_\kappa.\mathsf{Sign}(\mathsf{pk}_{\mathsf{ot}})},$$

$$\boxed{(\sigma, \bot) \leftarrow \Sigma.\mathsf{Adapt}(\mathsf{cpk}, \mathsf{pk}_{\mathsf{ot}}, \sigma', \mathsf{sk}')}, \text{ and}$$

$$\sigma_{\mathsf{ot}} \leftarrow \Sigma_{\mathsf{ot}}.\mathsf{Sign}(\mathsf{sk}_{\mathsf{ot}}, \pi_\Pi || x || \mathsf{pk} || \sigma).$$

Transition - Game 0 → Game 1: Under adaptability of signatures, this change is only conceptual, i.e., $\mathsf{Pr}[S_0] = \mathsf{Pr}[S_1]$.

Game 2: As Game 1, but we further modify S_2' as follows: we engage with a strong one-time signature challenger in each call and keep a mapping from challengers to keys.

$\mathsf{S}_2'(\mathsf{crs}, x)$: Obtain $\mathsf{sk}' \xleftarrow{R} \mathbb{H}$, $\mathsf{pk} \leftarrow \mathsf{cpk} \cdot \mu(\mathsf{sk}')$. Further, obtain $\boxed{\mathsf{pk}_{\mathsf{ot}} \leftarrow \mathcal{C}_\kappa^{\mathsf{ot}}}$ and return $\pi \leftarrow (\pi_\Pi, \mathsf{pk}, \sigma, \mathsf{pk}_{\mathsf{ot}}, \sigma_{\mathsf{ot}})$, where

$$\pi_\Pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (x, \mathsf{cpk}, \mathsf{pk}), (\bot, -\mathsf{sk}')),\ \sigma' \leftarrow \mathcal{C}_\kappa^{\mathsf{f}}.\mathsf{Sign}(\mathsf{pk}_{\mathsf{ot}}),$$

$$(\sigma, \bot) \leftarrow \Sigma.\mathsf{Adapt}(\mathsf{cpk}, \mathsf{pk}_{\mathsf{ot}}, \sigma', \mathsf{sk}'), \text{ and}$$

$$\boxed{\sigma_{\mathsf{ot}} \leftarrow \mathcal{C}_\kappa^{\mathsf{ot}}.\mathsf{Sign}(\pi_\Pi || x || \mathsf{pk} || \sigma)}.$$

Transition - Game 1 → Game 2: This change is conceptual, i.e., $\mathsf{Pr}[S_1] = \mathsf{Pr}[S_2]$.

Game 3: As Game 2, but we assume that E_2 used inside E does not fail to extract a valid witness with respect to L' (i.e., we abort if it fails).

Transition - Game 2 → Game 3: We bound the probability that the adversary outputs a tuple $(x^\star, \pi^\star)$ in Game 3 so that E_2 fails. We refer to this event as F_1. For the sake of contradiction assume that $\mathsf{Pr}[F_1]$ is non-negligible. Then we could obtain crs_Π from a proof of knowledge challenger and set $\xi \leftarrow \bot$ within S_1'. Whenever the adversary outputs $(x^\star, \pi^\star)$ we output it to the challenger. Now, the probability for our reduction to win the proof of knowledge game is exactly $\mathsf{Pr}[F_1]$. That is, we have that $|\mathsf{Pr}[S_2] - \mathsf{Pr}[S_3]| \leq \varepsilon_{\mathsf{e2}}(\kappa)$.

Game 4: As Game 3, but we assume that for every tuple $(x^\star, \pi^\star)$ output by the adversary, E never fails to output a witness w so that $R(x, w) = 1$ (i.e., we abort if it fails).

Transition Game 3 → Game 4: We bound the probability that the adversary manages to come up with a tuple $(x^\star, \pi^\star)$, where $\pi^\star = (\pi_\Pi^\star, \mathsf{pk}^\star, \sigma^\star, \mathsf{pk}_{\mathsf{ot}}^\star, \sigma_{\mathsf{ot}}^\star)$, so that we extract $(\bot, \mathsf{sk}_{\mathsf{e}}) \leftarrow \mathsf{E}_2(\mathsf{crs}, \xi, x^\star, \pi^\star)$ inside E. We refer to this event as F_2. If F_2 happens, we obtain a signature $(\sigma_{\mathsf{f}}, \mathsf{cpk}) \leftarrow \Sigma.\mathsf{Adapt}(\mathsf{pk}^\star, \mathsf{pk}_{\mathsf{ot}}^\star, \sigma^\star, \mathsf{sk}_{\mathsf{e}})$. By definition of the game we know that $(x^\star, \pi^\star)$ is not a query-answer pair of the simulator. Thus, we have two cases: (1) A signature on $\mathsf{pk}_{\mathsf{ot}}^\star$ was never obtained from the EUF-CMA challenger and we can output $(\mathsf{pk}_{\mathsf{ot}}^\star, \sigma_{\mathsf{f}})$ as a valid EUF-CMA forgery. (2) A signature on $\mathsf{pk}_{\mathsf{ot}}^\star$ was previously obtained. Then we have by definition that either $\pi_\Pi^\star || x^\star || \mathsf{pk}^\star || \sigma^\star$ or $\sigma_{\mathsf{ot}}^\star$ is different from the tuple signed by the strong one-time signature challenger upon simulation and we can output $(\pi_\Pi^\star || x^\star || \mathsf{pk}^\star || \sigma^\star, \sigma_{\mathsf{ot}}^\star)$ as a forgery for the strong one-time signature scheme to the respective challenger. Taking the union bound yields $|\mathsf{Pr}[S_3] - \mathsf{Pr}[S_4]| \leq q \cdot \varepsilon_{\mathsf{ot}}(\kappa) + \varepsilon_{\mathsf{f}}(\kappa)$.

In Game 4, we always extract a witness w such that $R(x^\star, w) = 1$, i.e., $\mathsf{Pr}[S_4] = 0$; Game 0 and Game 4 are computationally indistinguishable. Overall, we obtain $\mathsf{Pr}[S_0] \leq q \cdot \varepsilon_{\mathsf{ot}}(\kappa) + \varepsilon_{\mathsf{f}}(\kappa) + \varepsilon_{\mathsf{e2}}(\kappa)$, which completes the proof. □