Two-Party Adaptor Signatures From Identification Schemes*

Andreas Erwig[1], Sebastian Faust[1], Kristina Hostáková[2,†], Monosij Maitra[1,‡], and Siavash Riahi[1]

[1] Technische Universität Darmstadt, Germany
`firstname.lastname@tu-darmstadt.de`
[2] ETH Zürich, Switzerland
`kristina.hostakova@inf.ethz.ch`

Abstract. Adaptor signatures are a novel cryptographic primitive with important applications for cryptocurrencies. They have been used to construct second layer solutions such as payment channels or cross-currency swaps. The basic idea of an adaptor signature scheme is to tie the signing process to the revelation of a secret value in the sense that, much like a regular signature scheme, an adaptor signature scheme can authenticate messages, but simultaneously leaks a secret to certain parties. Recently, Aumayr et al. provide the first formalization of adaptor signature schemes, and present provably secure constructions from ECDSA and Schnorr signatures. Unfortunately, the formalization and constructions given in this work have two limitations: (1) current schemes are limited to ECDSA and Schnorr signatures, and no generic transformation for constructing adaptor signatures is known; (2) they do not offer support for aggregated two-party signing, which can significantly reduce the blockchain footprint in applications of adaptor signatures.

In this work, we address these two shortcomings. First, we show that signature schemes that are constructed from identification (ID) schemes, which additionally satisfy certain homomorphic properties, can generically be transformed into adaptor signature schemes. We further provide an impossibility result which proves that unique signature schemes (e.g., the BLS scheme) cannot be transformed into an adaptor signature scheme. In addition, we define two-party adaptor signature schemes with aggregatable public keys and show how to instantiate them via a generic transformation from ID-based signature schemes. Finally, we give instantiations of our generic transformations for the Schnorr, Katz-Wang and Guillou-Quisquater signature schemes.

1 Introduction

Blockchain technologies, envisioned first in 2009 [36], have spurred enormous interest by academia and industry. This technology puts forth a decentralized payment paradigm, where financial transactions are stored in a decentralized data structure – often referred to as the blockchain. The main cryptographic primitive used by blockchain systems is the one of digital signature schemes, which allow users to authenticate payment transactions. Various different flavors of digital signature schemes are used by blockchain systems, e.g., ring signatures [41] add privacy-preserving features to cryptocurrencies [42], while threshold signatures and multi-signatures are used for multi-factor authorization of transactions [20].

Adaptor signatures (sometimes also referred to as scriptless scripts) are another important type of digital signature scheme introduced by the cryptocurrency community [39] and recently formalized by Aumayr et al. [2]. In a nutshell, adaptor signatures tie together authorization of a message and the leakage of a secret value. Namely, they allow a *signer* to produce a *pre-signature* under her secret key such that this pre-signature can be *adapted* into a valid signature by a *publisher* knowing a certain secret value. If the completed signature gets published, the signer is able to extract the embedded secret used by the publisher.

∗ This is the full version of the paper accepted at PKC 2021.
† Research partially conducted at Technische Universität Darmstadt, Germany.
‡ Research partially conducted at Indian Institute of Technology Madras, India.

To demonstrate the concept of adaptor signatures, let us discuss the simple example of a preimage sale which serves as an important building block in many blockchain applications such as payment channels [6, 13, 40, 2], payment routing in payment channel networks (PCNs) [32, 16, 35] or atomic swaps [14, 23]. Assume that a seller offers to reveal a preimage of a hash value h in exchange for c coins from a concrete buyer. This is a classical instance of a fair exchange problem, which can be solved using the blockchain as follows. The buyer locks c coins in a transaction which can be spent by another transaction if it is authorized by the seller and contains a preimage of the hash value h.

While this solution implements the preimage sale, it has various drawbacks: (i) The only hash functions that can be used are the ones supported by the underlying blockchain. For example, the most popular blockchain-based cryptocurrency, Bitcoin, supports only SHA-1, SHA-256 and RIPEMD-160 [5]. This makes the above solution unsuitable for applications like privacy-preserving payment routing in PCNs [32, 16] that crucially rely on the preimage sale instantiated with a *homomorphic* hash function. (ii) The hash value has to be fixed at the beginning of the sale and cannot be changed later without a new transaction being posted on the blockchain. This is problematic in, e.g., generalized payment channels [2], where users utilize the ideas from the preimage sale to repeatedly update channel balances without any blockchain interaction. (iii) Finally, the blockchain script is non-standard as, in addition to a signature verification, it contains a hash preimage verification. This does not only make the transaction more expensive but also allows parties who are maintaining the blockchain (also known as *miners*) to censor transactions belonging to a preimage sale.

The concept of adaptor signatures allows us to implement a preimage sale in a way that overcomes most of the aforementioned drawbacks. The protocol works at a high level as follows. The buyer locks c coins in a transaction which can be spent by a transaction authorized by *both* the seller and the buyer. Thereafter, the buyer pre-signs a transaction spending the c coins with respect to the hash value h. If the seller knows a preimage of h, she can adapt the pre-signature of the buyer, attach her own signature and claim the c coins. The buyer can then extract a preimage from the adapted signature. Hence, parties are not restricted to the hash functions supported by the blockchain, i.e., drawback (i) is addressed. Moreover, the buyer can pre-sign the spending transaction with respect to multiple hash values which overcomes drawback (ii). However, the third drawback remains. While the usage of adaptor signatures avoids the hash preimage verification in the script, it adds a signature verification (i.e., there are now 2 signature verifications in total) which makes this type of exchange easily distinguishable from a normal payment transaction. Hence, the sale remains rather expensive and censorship is not prevented.

The idea of *two-party* adaptor signatures is to replace the two signature verifications by one. The transaction implementing a preimage sale then has exactly the same format as a transaction simply transferring coins. As a result the price (in terms of fees paid to the miners) of the preimage sale transaction is the same as the price for a normal payment. Moreover, censorship is prevented as miners cannot distinguish the transactions belonging to the preimage sale from a standard payment transaction. Hence, point (iii) is fully addressed.

The idea of replacing two signatures by one has already appeared in the literature in the context of payment channels. Namely, Malavolta et al. [32] presented protocols for two-party threshold adaptor signatures based on Schnorr and ECDSA digital signatures. However, they did not present a standalone definition for the threshold primitive and hence security for these schemes has not been analyzed. Furthermore, the key generation of the existing threshold adaptor signature schemes is interactive which is undesirable. Last but not least, their constructions are tailored to Schnorr and ECDSA signature schemes and hence is not generic. From the above points, the following natural question arises:

Is it possible to define and instantiate two-party adaptor signature schemes with non-interactive key generation in a generic way?

1.1 Our contribution

Our main goal is to define two-party adaptor signatures and explore from which digital signature we can instantiate this new primitive. We proceed in three steps which we summarize below and depict in Fig. 1.

Step 1: From ID schemes to adaptor signatures. Our first goal is to determine if there exists a specific class of signature schemes which can be generically transformed into adaptor signatures. Given the existing Schnorr-based construction [39, 2], a natural choice is to explore signature schemes constructed in a similar fashion. To this end, we focus on signature schemes built from identification (ID) schemes using the Fiat-Shamir transform [27]. We show that ID-based signature schemes satisfying certain additional properties can be transformed to adaptor signature schemes generically. In addition to Schnorr signatures [43], this class includes Katz-Wang and Guillou-Quisquater signatures [26, 24]. As an additional result, we show that adaptor signatures *cannot* be built from unique signatures, ruling out constructions from, e.g., BLS signatures [10].

Our generic transformation of adaptor signatures from ID schemes has multiple benefits. Firstly, by instantiating it with the Guillou-Quisquater siganture scheme, we obtain the first RSA-based adaptor signature scheme. Secondly, since Katz-Wang signatures offers tight security (under the decisional Diffie-Hellman (DDH) assumption), and our generic transformation also achieves tight security, our result shows how to construct adaptor signatures with a tight reduction to the underlying DDH assumption.

Step 2: From ID schemes to two-party signatures. Our second goal is to generically transform signature schemes built from ID schemes into two-party signature schemes with aggregatable public keys. Unlike threshold signatures, these signatures have non-interactive key generation. This means that parties can independently generate their key pairs and later collaboratively generate signatures that are valid under their *combined* public key. For our transformation, we require the signature scheme to satisfy certain aggregation properties which, as we show, are present in the three aforementioned signature schemes. While this transformation serves as a middle step towards our main goal of constructing two-party adaptor signatures, we believe it is of independent interest.

Step 3: From ID schemes to two-party adaptor signatures. Finally, we define two-party adaptor signature schemes with aggregatable public keys. In order to instantiate this novel cryptographic primitive, we use similar techniques as in step 1 where we "lifted" standard signature schemes to adaptor signature schemes. More precisely, we present a transformation turning a two-party signature scheme based on an ID scheme into a two-party adaptor signature scheme.

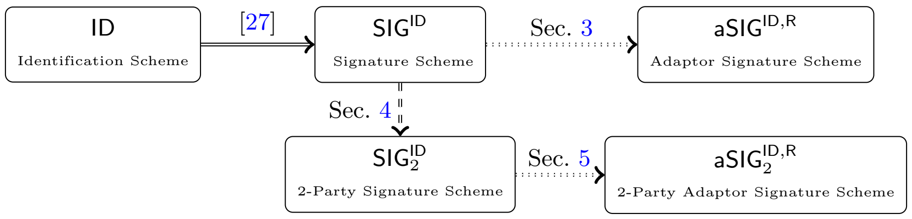

Fig. 1: Overview of our results. Full arrow represents a generic transformation, dotted and dashed arrows represent a generic transformation which requires additional homomorphic or aggregation properties respectively.

Remark 1. Let us point out that Fig. 1 presents our transformation steps from signature schemes based on ID schemes to two-party adaptor signatures. Despite the fact that we generically construct our two-party adaptor signature scheme from two-party signature schemes based on ID schemes, we reduce its security to the strong unforgeability of the underlying single party signature scheme. Therefore, we do not need the two-party signature scheme from ID schemes to be strongly unforgeable. This gives us a more general result than proving security based on strong unforgeability of the two-party signature scheme from ID schemes. We note that any ID scheme can be transformed to a signature scheme with strong unforgeability by Bellare and Shoup [4].

Let us further mention that our security proofs are in the random oracle model. Proving the security of our constructions and the original constructions from [2] in the standard model remains an interesting open problem.

1.2 Related Work

Adaptor Signatures. The notion of adaptor signatures was first introduced by Poelstra [39] and has since been used in many blockchain related applications, such as PCNs [32], payment channel hubs [45] or atomic swaps [14]. However, the adaptor signatures as a standalone primitive were only formalized later by Aumayr et al. [2], where they were used to generalize the concept of payment channels. Concurrently, Fournier [19] attempted to formalize adaptor signatures, however, as pointed out in [2], his definition is weaker than the one given in [2] and not sufficient for certain applications. All the previously mentioned works constructed adaptor signatures only from Schnorr and ECDSA signatures, i.e., they did not show generic transformations for building adaptor signature schemes. As previously mentioned, a two-party threshold variant of adaptor signatures was presented by Malavolta et al. [32]. Their construction requires interactive key generation, thereby differing from our two-party adaptor signature notion. Moreover, no standalone definition of the threshold primitive was provided.

Two works [17, 46] have recently introduced post-quantum secure adaptor signature schemes, i.e., schemes that remain secure even in presence of an adversary having access to a quantum computer. In order to achieve post-quantum security, [17] based its scheme on standard and well-studied lattice assumptions, namely Module-SIS and Module-LWE, while the scheme in [46] is based on lesser known assumptions for isogenies. Both works additionally show how to construct post-quantum secure PCNs from their respective adaptor signature schemes.

Multi-Signatures and ID Schemes. Multi-Signatures have been subject to extensive research in the past (e.g., [38, 37, 25]). In a nutshell, multi-signatures allow a set of signers to collaboratively generate a signature for a common message such that the signature can be verified given the public key of each signer. More recently, the notion of multi-signatures with aggregatable public keys has been introduced [33] and worked on [9, 28], which allows to aggregate the public keys of all signers into one single public key. We use some results from the work of Kiltz et al. [27], which provides a concrete and modular security analysis of signatures schemes from ID schemes obtained via the Fiat-Shamir transformation. Our paper builds up on their work and uses some of their notation.

2 Preliminaries

Notation. We denote by $x \leftarrow_\$ \mathcal{X}$ the uniform sampling of x from the set $\mathcal{X}$. Throughout this paper, n denotes the security parameter. By $x \leftarrow \mathsf{A}(y)$ we denote a *probabilistic polynomial time* (PPT) algorithm A that on input y, outputs x. When A is a *deterministic polynomial time* (DPT) algorithm, we use the notation $x := \mathsf{A}(y)$. A function $\nu\colon \mathbb{N} \to \mathbb{R}$ is *negligible in* n if for every $k \in \mathbb{N}$, there exists $n_0 \in \mathbb{N}$ s.t. for every $n \geq n_0$ it holds that $|\nu(n)| \leq 1/n^k$.

Digital signatures. A digital signature scheme is a triple of algorithms $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$, where $\mathsf{Gen}(1^n)$ is a PPT algorithm that on input the security parameter, outputs a secret and public key pair (sk, pk); $\mathsf{Sign}_{sk}(m)$ is a PPT algorithm that on input a secret key sk and a message $m \in \{0,1\}^*$, outputs a signature σ; and $\mathsf{Vrfy}_{pk}(m; \sigma)$ is a DPT algorithm that on input a public key pk, a message m and a signature σ, outputs a bit b. A signature scheme satisfies correctness if for all messages $m \in \{0,1\}^*$ and valid key pairs $(sk, pk) \leftarrow \mathsf{Gen}(1^n)$ it holds that $\mathsf{Vrfy}_{pk}(m; \mathsf{Sign}_{sk}(m)) = 1$. We say that SIG is *deterministic* if Sign is a DPT algorithm. We say that SIG has *unique signatures* if for all $n \in \mathbb{N}$, key pairs $(sk, pk) \leftarrow \mathsf{Gen}(1^n)$ and messages $m \in \{0,1\}^*$ there exists only one signature σ for which $\mathsf{Vrfy}_{pk}(m, \sigma) = 1$.

In this work, we use signature schemes that satisfy the notion of (strong) existential unforgeability under chosen message attack ($\mathsf{EUF{-}CMA}$ or $\mathsf{SUF{-}CMA}$). At a high level, $\mathsf{EUF{-}CMA}$ guarantees that a PPT adversary on input the public key pk and with access to a signing oracle, cannot produce a valid signature on a fresh message m. The $\mathsf{SUF{-}CMA}$ definition provides a stronger guarantee where the adversary cannot produce a *new* valid signature on any message m. We recall the formal definition of $\mathsf{EUF{-}CMA}$ and $\mathsf{SUF{-}CMA}$ in Appx. A.

Hard relation. Let $\mathsf{R} \subseteq \mathcal{D}_S \times \mathcal{D}_\mathsf{w}$ be a relation with statement/witness pairs $(Y, y) \in \mathcal{D}_S \times \mathcal{D}_\mathsf{w}$ and let the language $L_\mathsf{R} \subseteq \mathcal{D}_S$ associated to R be defined as $L_\mathsf{R} := \{Y \in \mathcal{D}_S \mid \exists y \in \mathcal{D}_\mathsf{w} \text{ s.t. } (Y, y) \in \mathsf{R}\}$. We say that R is a *hard relation* if: (i) There exists a PPT sampling algorithm $\mathsf{GenR}(1^n)$ that on input the security parameter outputs a pair $(Y, y) \in \mathsf{R}$; (ii) The relation R is poly-time decidable; (iii) For all PPT adversaries $\mathcal{A}$, the probability that $\mathcal{A}$ outputs a valid witness $y \in \mathcal{D}_\mathsf{w}$ for $Y \in L_\mathsf{R}$ is negligible.

Non-interactive zero knowledge proof. We now recall the definition of a non-interactive zero-knowledge (NIZK) proof of knowledge which has first been introduced by Blum et al. [7]. A NIZK proof of knowledge with respect to a polynomial-time recognizable binary relation R is given by the following tuple of PPT algorithms $\mathsf{NIZK} := (\mathsf{Setup}_\mathsf{R}, \mathsf{Prove}, \mathsf{Verify})$, where (i) $\mathsf{Setup}_\mathsf{R}(1^n)$ outputs a common reference string crs; (ii) $\mathsf{Prove}(\mathsf{crs}, (Y, y))$ outputs a proof π for $(Y, y) \in \mathsf{R}$; (iii) $\mathsf{Verify}(\mathsf{crs}, Y, \pi)$ outputs a bit $b \in \{0, 1\}$. Further, the NIZK proof of knowledge w.r.t. R should satisfy the following properties: (i) completeness, (ii) soundness and (iii) zero-knowledge. We refer to Appx. A for further details.

Extractable commitments. Extractable commitment schemes have been first introduced by De Santis et al. [12]. A commitment scheme consists of a tuple of three PPT algorithms, $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Dec})$ where Gen gets as input the security parameter n and outputs public parameters pp, Com takes as input pp and a message $m \in \{0, 1\}^*$ and outputs a tuple (c, d) and Dec takes as input pp and a tuple (c, d) and either outputs m or $\perp$. Let n be the security parameter and let $pp \leftarrow \mathsf{Gen}(1^n)$. A commitment scheme must satisfy two properties: (i) hiding and (ii) binding. An extractable commitment scheme additionally satisfies a third property, namely extracability. We refer to Appx. A for further details.

2.1 Adaptor Signatures

We now recall the definition of adaptor signatures, recently put forward in [2].

Definition 1 (Adaptor signature). *An adaptor signature scheme w.r.t. a hard relation* R *and a signature scheme* $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *consists of a tuple of four algorithms* $\mathsf{aSIG}_{\mathsf{R},\mathsf{SIG}} = (\mathsf{pSign}, \mathsf{Adapt}, \mathsf{pVrfy}, \mathsf{Ext})$ *defined as:*

$\mathsf{pSign}_{sk}(m, Y)$**:** *is a PPT algorithm that on input a secret key sk, message $m \in \{0, 1\}^*$ and statement $Y \in L_\mathsf{R}$, outputs a pre-signature $\widetilde{\sigma}$.*

$\mathsf{pVrfy}_{pk}(m, Y; \widetilde{\sigma})$**:** *is a DPT algorithm that on input a public key pk, message $m \in \{0, 1\}^*$, statement $Y \in L_\mathsf{R}$ and pre-signature $\widetilde{\sigma}$, outputs a bit b.*

$\mathsf{Adapt}_{pk}(\widetilde{\sigma}, y)$**:** *is a DPT algorithm that on input a pre-signature $\widetilde{\sigma}$ and witness y, outputs a signature σ.*

$\mathsf{Ext}_{pk}(\sigma, \widetilde{\sigma}, Y)$**:** *is a DPT algorithm that on input a signature σ, pre-signature $\widetilde{\sigma}$ and statement $Y \in L_\mathsf{R}$, outputs a witness y such that $(Y, y) \in \mathsf{R}$, or $\perp$.*

An adaptor signature scheme, besides satisfying plain digital signature correctness, should also satisfy pre-signature correctness that we formalize next.

Definition 2 (Pre-signature correctness). *An adaptor signature* $\mathsf{aSIG}_{\mathsf{R},\mathsf{SIG}}$ *satisfies* pre-signature correctness*, if for all $n \in \mathbb{N}$ and $m \in \{0, 1\}^*$:*

$$\Pr\left[\begin{array}{c} \mathsf{pVrfy}_{pk}(m, Y; \widetilde{\sigma}) = 1 \wedge \\ \mathsf{Vrfy}_{pk}(m; \sigma) = 1 \wedge \\ (Y, y') \in \mathsf{R} \end{array} \middle| \begin{array}{l} (sk, pk) \leftarrow \mathsf{Gen}(1^n), (Y, y) \leftarrow \mathsf{GenR}(1^n) \\ \widetilde{\sigma} \leftarrow \mathsf{pSign}_{sk}(m, Y), \sigma := \mathsf{Adapt}_{pk}(\widetilde{\sigma}, y) \\ y' := \mathsf{Ext}_{pk}(\sigma, \widetilde{\sigma}, Y) \end{array}\right] = 1.$$

An adaptor signature scheme $\mathsf{aSIG}_{\mathsf{R},\mathsf{SIG}}$ is called *secure* if it satisfies three security properties: *existential unforgeablity under chosen message attack for adaptor signatures*, *pre-signature adaptability* and *witness extractability*. Let us recall the formal definition of these properties next.

The notion of unforgeability for adaptor signatures is similar to existential unforgeability under chosen message attacks for standard digital signatures but additionally requires that producing a forgery σ for some message m^* is hard even given a pre-signature on m^* w.r.t. a random statement $Y \in L_{\mathsf{R}}$.

Definition 3 **($\mathsf{aEUF{-}CMA}$ Security).** *An adaptor signature scheme* $\mathsf{aSIG}_{\mathsf{R},\mathsf{SIG}}$ *is unforgeable if for every PPT adversary* $\mathcal{A}$ *there exists a negligible function* ν *such that:* $\Pr[\mathsf{aSigForge}_{\mathcal{A},\mathsf{aSIG}_{\mathsf{R},\mathsf{SIG}}}(n) = 1] \leq \nu(n)$*, where the definition of the experiment* $\mathsf{aSigForge}_{\mathcal{A},\mathsf{aSIG}_{\mathsf{R},\mathsf{SIG}}}$ *is as follows:*

```
$\mathsf{aSigForge}_{\mathcal{A},\mathsf{aSIG}_{\mathsf{R},\mathsf{SIG}}}(n)$
1 : $\mathcal{Q} := \emptyset, (sk, pk) \leftarrow \mathsf{Gen}(1^n)$
2 : $m^* \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{S}},\mathcal{O}_{\mathrm{pS}}}(pk)$
3 : $(Y, y) \leftarrow \mathsf{GenR}(1^n), \widetilde{\sigma} \leftarrow \mathsf{pSign}_{sk}(m^*, Y)$
4 : $\sigma^* \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{S}},\mathcal{O}_{\mathrm{pS}}}(\widetilde{\sigma}, Y)$
5 : return $\big(m^* \notin \mathcal{Q} \wedge \mathsf{Vrfy}_{pk}(m^*; \sigma^*)\big)$

$\mathcal{O}_{\mathrm{S}}(m)$
1 : $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$
2 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
3 : return $\sigma$

$\mathcal{O}_{\mathrm{pS}}(m, Y)$
1 : $\widetilde{\sigma} \leftarrow \mathsf{pSign}_{sk}(m, Y)$
2 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
3 : return $\widetilde{\sigma}$
```

A natural requirement for an adaptor signature scheme is that any valid pre-signature w.r.t. Y (possibly produced by a malicious signer) can be completed into a valid signature using a witness y with $(Y, y) \in \mathsf{R}$.

Definition 4 **(Pre-signature adaptability).** *An adaptor signature scheme* $\mathsf{aSIG}_{\mathsf{SIG},\mathsf{R}}$ *satisfies pre-signature adaptability, if for all* $n \in \mathbb{N}$*, messages* $m \in \{0,1\}^*$*, statement/witness pairs* $(Y, y) \in \mathsf{R}$*, public keys pk and pre-signatures* $\widetilde{\sigma} \leftarrow \{0,1\}^*$ *we have* $\mathsf{pVrfy}_{pk}(m, Y; \widetilde{\sigma}) = 1$*, then* $\mathsf{Vrfy}_{pk}(m; \mathsf{Adapt}_{pk}(\widetilde{\sigma}, y)) = 1$.

The last property that we are interested in is *witness extractability.* Informally, it guarantees that a valid signature/pre-signatue pair $(\sigma, \widetilde{\sigma})$ for message/statement (m, Y) can be used to extract a corresponding witness y.

Definition 5 **(Witness extractability).** *An adaptor signature scheme* $\mathsf{aSIG}_{\mathsf{R}}$ *is* witness extractable *if for every PPT adversary* $\mathcal{A}$*, there exists a negligible function* ν *such that the following holds:* $\Pr[\mathsf{aWitExt}_{\mathcal{A},\mathsf{aSIG}_{\mathsf{R},\mathsf{SIG}}}(n) = 1] \leq \nu(n)$*, where the experiment* $\mathsf{aWitExt}_{\mathcal{A},\mathsf{aSIG}_{\mathsf{R},\mathsf{SIG}}}$ *is defined as follows:*

```
$\mathsf{aWitExt}_{\mathcal{A},\mathsf{aSIG}_{\mathsf{R},\mathsf{SIG}}}(n)$
1 : $\mathcal{Q} := \emptyset, (sk, pk) \leftarrow \mathsf{Gen}(1^n)$
2 : $(m^*, Y^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{S}},\mathcal{O}_{\mathrm{pS}}}(pk)$
3 : $\widetilde{\sigma} \leftarrow \mathsf{pSign}_{sk}(m^*, Y^*)$
4 : $\sigma^* \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{S}},\mathcal{O}_{\mathrm{pS}}}(\widetilde{\sigma})$
5 : $y := \mathsf{Ext}_{pk}(\sigma^*, \widetilde{\sigma}, Y^*)$
6 : return $(m^* \notin \mathcal{Q} \wedge (Y^*, y) \notin \mathsf{R} \wedge \mathsf{Vrfy}_{pk}(m^*; \sigma^*))$

$\mathcal{O}_{\mathrm{S}}(m)$
1 : $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$
2 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
3 : return $\sigma$

$\mathcal{O}_{\mathrm{pS}}(m, Y)$
1 : $\widetilde{\sigma} \leftarrow \mathsf{pSign}_{sk}(m, Y)$
2 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
3 : return $\widetilde{\sigma}$
```

Let us stress that while the witness extractability experiment $\mathsf{aWitExt}$ looks fairly similar to the experiment $\mathsf{aSigForge}$, there is one crucial difference; namely, the adversary is allowed to choose the forgery statement Y^*. Hence, we can assume that it knows a witness for Y^* and can thus generate a valid signature on the forgery message m^*. However, this is not sufficient to win the experiment. The adversary wins *only* if the valid signature does not reveal a witness for Y^*.

2.2 Identification and Signature Schemes

In this section we recall the definition of identification schemes and how they are transformed to signature schemes as described in [27].

Definition 6 (Canonical Identification Scheme [27]). *A canonical identification scheme* ID *is defined as a tuple of four algorithms* $\mathsf{ID} := (\mathsf{IGen}, \mathsf{P}, \mathsf{ChSet}, \mathsf{V})$.

- *The key generation algorithm* IGen *takes the system parameters* par *as input and returns secret and public key* (sk, pk). *We assume that* pk *defines the set of challenges, namely* ChSet.
- *The prover algorithm* P *consists of two algorithms namely* P_1 *and* P_2:
  - P_1 *takes as input the secret key* sk *and returns a commitment* $R \in \mathcal{D}_{\mathsf{rand}}$ *and a state* St.
  - P_2 *takes as input the secret key* sk, *a commitment* $R \in \mathcal{D}_{\mathsf{rand}}$, *a challenge* $h \in \mathsf{ChSet}$, *and a state* St *and returns a response* $s \in \mathcal{D}_{\mathsf{resp}}$.
- *The verifier algorithm* V *is a deterministic algorithm that takes the public key* pk *and the conversation transcript as input and outputs* 1 *(acceptance) or* 0 *(rejection).*

We require that for all $(sk, pk) \in \mathsf{IGen}(\mathsf{par})$, *all* $(R, St) \in \mathsf{P}_1(sk)$, *all* $h \in \mathsf{ChSet}$ *and all* $s \in \mathsf{P}_2(sk, R, h, St)$, *we have* $\mathsf{V}(pk, R, h, s) = 1$.

We recall that an identification scheme ID is called *commitment-recoverable*, if V first internally calls a function V_0 which recomputes $R_0 = \mathsf{V}_0(pk, h, s)$ and then outputs 1, iff $R_0 = R$. Using Fiat-Shamir heuristic one can transform any identification scheme ID of the above form into a digital signature scheme $\mathsf{SIG}^{\mathsf{ID}}$. We recall this transformation in Fig. 2 when ID is commitment-recoverable.

$\mathsf{Gen}(1^n)$	$\mathsf{Sign}_{sk}(m)$	$\mathsf{Vrfy}_{pk}(m; (h, s))$
$1: (sk, pk) \leftarrow \mathsf{IGen}(n)$	$1: (R, St) \leftarrow \mathsf{P}_1(sk)$	$1: R := \mathsf{V}_0(pk, h, s)$
$2: \textbf{return } (sk, pk)$	$2: h := \mathcal{H}(R, m)$	$2: \textbf{return } h = \mathcal{H}(R, m)$
	$3: s \leftarrow \mathsf{P}_2(sk, R, h, St)$	
	$4: \textbf{return } (h, s)$	

Fig. 2: $\mathsf{SIG}^{\mathsf{ID}}$: Digital signature schemes from identification schemes [27]

3 Adaptor Signatures from $\mathsf{SIG}^{\mathsf{ID}}$

Our first goal is to explore and find digital signature schemes which can generically be transformed to adaptor signatures. Interestingly, we observe that both existing adaptor signature schemes, namely the Schnorr-based and the ECDSA-based schemes, utilize the randomness used during signature generation to transform digital signatures to adaptor signatures [2]. We first prove a negative result, namely that it is impossible to construct an adaptor signature scheme from a unique signature scheme [44, 31, 21]. Thereafter, we focus on signature schemes constructed from identification schemes (cf. Fig. 2) and show that if the underlying ID-based signature scheme $\mathsf{SIG}^{\mathsf{ID}}$ satisfies certain additional properties, then we can generically transform it into an adaptor signature scheme. To demonstrate the applicability of our generic transformation, we show in Appx. B that many existing $\mathsf{SIG}^{\mathsf{ID}}$ instantiations satisfy the required properties.

3.1 Impossibility Result for Unique Signatures

An important class of digital signatures are those where the signing algorithm is deterministic and the generated signatures are unique. Given the efficiency of deterministic signature schemes along with numerous other advantages that come from signatures being unique [44, 31, 21], it would be tempting to design adaptor signatures based on unique signatures. However, we show in Thm. 1 that if the signature scheme has unique signatures, then it is impossible to construct a secure adaptor signature scheme from it.

Theorem 1. *Let* R *be a hard relation and* $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *be a signature scheme with unique signatures. Then there does not exist an adaptor signature scheme* $\mathsf{aSIG}_{\mathsf{R},\mathsf{SIG}}$.

Proof. We prove this theorem by contradiction. Assume there exists an adaptor signature scheme where the underlying signature scheme, SIG, has unique signatures. We construct a PPT algorithm $\mathcal{A}$ which internally uses the adaptor signature and breaks the hardness of R. In other words, $\mathcal{A}$ receives $(1^n, Y)$ as input and outputs y, such that $(Y, y) \in \mathsf{R}$. Below, we describe $\mathcal{A}$ formally.

```
On input $(1^n, Y)$, $\mathcal{A}$ proceeds as follows:
1 : Sample a new key pair $(sk, pk) \leftarrow \mathsf{Gen}(1^n)$.
2 : Choose an arbitrary message $m$ from the signing message space.
3 : Generate a pre-signature, $\widetilde{\sigma} \leftarrow \mathsf{preSign}_{sk}(m, Y)$.
4 : Generate a signature, $\sigma := \mathsf{Sign}_{sk}(m)$.
5 : Compute and output $y := \mathsf{Ext}_{pk}(\sigma, \widetilde{\sigma}, Y)$.
```

We now show that y returned by $\mathcal{A}$ is indeed a witness of Y, i.e., $(Y, y) \in \mathsf{R}$. From the correctness of the adaptor signature scheme, we know that for any y' s.t. $(Y, y') \in \mathsf{R}$ the signature $\sigma' := \mathsf{Adapt}(\widetilde{\sigma}, y')$ is a valid signature, i.e., $\mathsf{Vrfy}_{pk}(m, \sigma') = 1$. Moreover, we know that $y'' := \mathsf{Ext}_{pk}(\sigma', \widetilde{\sigma}, Y)$ is such that $(Y, y'') \in \mathsf{R}$. As SIG is a unique signature scheme, this implies that $\sigma' = \sigma$ which in turn implies that the witness y returned by $\mathcal{A}$ is y''. Hence, $\mathcal{A}$ breaks the hardness of R with probability 1.

Let us briefly discuss which signature schemes are affected by our impossibility result. Unique signature schemes (also known as verifiable unpredictable functions (VUF)) have been first introduced in [21]. Furthermore, many follow-up works such as [34, 31] and most recently [44], have shown how to instantiate this primitive in the standard model. Another famous example of a unique signature scheme is BLS [10]. Naturally, due to our impossibility result, an adaptor signature scheme cannot be instantiated from these signature schemes.

3.2 Generic Transformation to Adaptor Signatures

We now describe how to generically transform a randomized digital signature scheme $\mathsf{SIG}^{\mathsf{ID}}$ from Fig. 2 into an adaptor signature scheme w.r.t. a hard relation R. For brevity, we denote the resulting adaptor signature scheme as $\mathsf{aSIG}^{\mathsf{ID},\mathsf{R}}$ instead of $\mathsf{aSIG}_{\mathsf{R},\mathsf{SIG}^{\mathsf{ID}}}$. The main idea behind our transformation is to *shift* the public randomness of the Sign procedure by a statement Y for the relation R in order to generate a modified signature called a *pre-signature.* Using a corresponding witness y (i.e., $(Y, y) \in \mathsf{R}$), the shift of the public randomness in the pre-signature can be reversed (or adapted), in order to obtain a regular (or full) signature. Moreover, it should be possible to extract a witness given both the pre-signature and the full-signature. To this end, let us formalize three new *deterministic* functions which we will use later in our transformation.

1. For the randomness shift, we define a function $f_{\mathsf{shift}} : \mathcal{D}_{\mathsf{rand}} \times L_{\mathsf{R}} \rightarrow \mathcal{D}_{\mathsf{rand}}$ that takes as input a commitment value $R \in \mathcal{D}_{\mathsf{rand}}$ of the identification scheme and a statement $Y \in L_{\mathsf{R}}$ of the hard relation, and outputs a new commitment value $R' \in \mathcal{D}_{\mathsf{rand}}$.
2. For the adapt operation, we define $f_{\mathsf{adapt}} : \mathcal{D}_{\mathsf{resp}} \times \mathcal{D}_{\mathsf{w}} \rightarrow \mathcal{D}_{\mathsf{resp}}$ that takes as input a response value $\tilde{s} \in \mathcal{D}_{\mathsf{resp}}$ of the identification scheme and a witness $y \in \mathcal{D}_{\mathsf{w}}$ of the hard relation, and outputs a new response value $s \in \mathcal{D}_{\mathsf{resp}}$.

3. Finally, for witness extraction, we define $f_{\mathsf{ext}}\colon \mathcal{D}_{\mathsf{resp}} \times \mathcal{D}_{\mathsf{resp}} \to \mathcal{D}_{\mathsf{w}}$ that takes as input two response values $\tilde{s}, s \in \mathcal{D}_{\mathsf{resp}}$ and outputs a witness $y \in \mathcal{D}_{\mathsf{w}}$.

Our transformation from $\mathsf{SIG}^{\mathsf{ID}}$ to $\mathsf{aSIG}^{\mathsf{ID,R}}$ is shown in Fig. 3.

```
pSign_sk(m, Y)
  1 : $(R_{\mathsf{pre}}, St) \leftarrow \mathsf{P}_1(sk)$
  2 : $R_{\mathsf{sign}} := f_{\mathsf{shift}}(R_{\mathsf{pre}}, Y)$
  3 : $h := \mathcal{H}(R_{\mathsf{sign}}, m)$
  4 : $\tilde{s} \leftarrow \mathsf{P}_2(sk, R_{\mathsf{pre}}, h, St)$
  5 : return $(h, \tilde{s})$

pVrfy_pk(m, Y; (h, s̃))
  1 : $\widehat{R}_{\mathsf{pre}} := \mathsf{V}_0(pk, h, \tilde{s})$
  2 : $\widehat{R}_{\mathsf{sign}} := f_{\mathsf{shift}}(\widehat{R}_{\mathsf{pre}}, Y)$
  3 : $b := (h = \mathcal{H}(\widehat{R}_{\mathsf{sign}}, m))$
  4 : return $b$

Adapt_pk((h, s̃), y)
  1 : $s = f_{\mathsf{adapt}}(\tilde{s}, y)$
  2 : return $(h, s)$

Ext_pk((h, s), (h, s̃), Y)
  1 : return $f_{\mathsf{ext}}(s, \tilde{s})$
```

Fig. 3: $\mathsf{aSIG}^{\mathsf{ID,R}}$: Generic transformation from $\mathsf{SIG}^{\mathsf{ID}}$ to a $\mathsf{aSIG}_{\mathsf{R,SIG}}$ scheme

In order for $\mathsf{aSIG}^{\mathsf{ID,R}}$ to be an adaptor signature scheme, we need the functions f_{shift}, f_{adapt} and f_{ext} to satisfy two properties. The first property is a homomorphic one and relates the functions f_{shift} and f_{adapt} to the commitment-recoverable component V_0 and the hard relation R. Informally, for all $(Y, y) \in \mathsf{R}$, we need the following to be equivalent: (i) Extract the public randomness from a response $\tilde{s}$ using V_0 and then apply f_{shift} to shift the public randomness by Y, and (ii) apply f_{adapt} to shift the *secret* randomness in $\tilde{s}$ by y and then extract the public randomness using V_0. Formally, for any public key pk, any challenge $h \in \mathsf{ChSet}$, any response value $\tilde{s} \in \mathcal{D}_{\mathsf{resp}}$ and any statement/witness pair $(Y, y) \in \mathsf{R}$, it must hold that:

$$f_{\mathsf{shift}}(\mathsf{V}_0(pk, h, \tilde{s}), Y) = \mathsf{V}_0(pk, h, f_{\mathsf{adapt}}(\tilde{s}, y)). \tag{1}$$

The second property requires that the function $f_{\mathsf{ext}}(\tilde{s}, \cdot)$ is the inverse function of $f_{\mathsf{adapt}}(\tilde{s}, \cdot)$ for any $\tilde{s} \in \mathcal{D}_{\mathsf{resp}}$. Formally, for any $y \in \mathcal{D}_{\mathsf{w}}$ and $\tilde{s} \in \mathcal{D}_{\mathsf{resp}}$, we have

$$y = f_{\mathsf{ext}}(f_{\mathsf{adapt}}(\tilde{s}, y), \tilde{s}). \tag{2}$$

To give an intuition about the functions f_{shift}, f_{adapt} and f_{ext} and their purpose, let us discuss their concrete instantiations for Schnorr signatures and show that they satisfy Equations (1) and (2). The instantiations for Katz-Wang signatures and Guillou-Quisquater signatures can be found in Appx. B.

Example 1 (Schnorr signatures). Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of prime order p where the discrete logarithm problem in $\mathbb{G}$ is hard. The functions IGen, P_1, P_2 and V_0 for Schnorr's signature scheme are defined in Fig. 4.

Let us consider the hard relation $\mathsf{R} = \{(Y, y) \mid Y = g^y\}$, i.e., group elements and their discrete logarithms, and let us define the functions f_{shift}, f_{adapt}, f_{ext} as:

$$f_{\mathsf{shift}}(Y, R) := Y \cdot R, \quad f_{\mathsf{adapt}}(\tilde{s}, y) := \tilde{s} + y, \quad f_{\mathsf{ext}}(s, \tilde{s}) := s - \tilde{s}.$$

```
IGen(n)
  1 : $sk \leftarrow_\$ \mathbb{Z}_q, pk = g^{sk}$
  2 : return $(sk, pk)$

P_1(sk)
  1 : $r \leftarrow_\$ \mathbb{Z}_q, R = g^r$
  2 : return $(R, r)$

P_2(sk, R, h, r)
  1 : $s = r + h \cdot sk$
  2 : return $s$

V_0(pk, h, s)
  1 : $R = g^s \cdot pk^{-h}$
  2 : return $(R)$
```

Fig. 4: Schnorr signature scheme

Intuitively, the function f_{shift} is *shifting* randomness in the group while the function f_{adapt} *shifts* randomness in the exponent. To prove that Eq. (1) holds, let us fix an arbitrary public key $pk \in \mathbb{G}$, a challenge $h \in \mathbb{Z}_q$, a response value $s \in \mathbb{Z}_q$ and a statement witness pair $(Y, y) \in \mathsf{R}$, i.e, $Y = g^y$. We have

$$\begin{aligned} f_{\mathsf{shift}}(\mathsf{V}_0(pk, h, s), Y) &= f_{\mathsf{shift}}(g^s \cdot pk^{-h}, Y) = g^s \cdot pk^{-h} \cdot Y \\ &= g^{s+y} \cdot pk^{-h} = \mathsf{V}_0(pk, h, s + y) = \mathsf{V}_0(pk, h, f_{\mathsf{adapt}}(s, y)) \end{aligned}$$

which is what we wanted to prove. In order to show that Eq. (2) holds, let us fix an arbitrary witness $y \in \mathbb{Z}_q$ and a response value $s \in \mathbb{Z}_q$. Then we have

$$f_{\mathsf{ext}}(f_{\mathsf{adapt}}(s, y), s) = f_{\mathsf{ext}}(s + y, s) = s + y - s = y$$

and hence Eq. (2) is satisfied as well.

We now show that the transformation from Fig. 3 is a secure adaptor signature scheme if functions $f_{\mathsf{shift}}, f_{\mathsf{adapt}}, f_{\mathsf{ext}}$ satisfying Equations (1) and (2) exist.

Theorem 2. *Assume that* $\mathsf{SIG}^{\mathsf{ID}}$ *is a* $\mathsf{SUF{-}CMA}$*-secure signature scheme transformed using Fig. 2, let* $f_{\mathsf{shift}}, f_{\mathsf{adapt}}$ *and* f_{ext} *be functions satisfying the relations from Equations (1) and (2), and* R *be a hard relation. Then the resulting* $\mathsf{aSIG}^{\mathsf{ID},\mathsf{R}}$ *scheme from the transformation in Fig. 3 is a secure adaptor signature scheme in the random oracle model.*

In order to prove Thm. 2, we must show that $\mathsf{aSIG}^{\mathsf{ID},\mathsf{R}}$ satisfies *pre-signature correctness*, $\mathsf{aEUF{-}CMA}$ *security*, *pre-signature adaptability* and *witness extractability* properties described in Defs. 2 to 5 respectively.

Lemma 1 (Pre-Signature Correctness). *Under the assumptions of Thm. 2,* $\mathsf{aSIG}^{\mathsf{ID},\mathsf{R}}$ *satisfies pre-signature correctness as for Def. 2.*

Proof. Let us fix an arbitrary message m and a statement witness pair $(Y, y) \in \mathsf{R}$. Let $(sk, pk) \leftarrow \mathsf{Gen}(1^n)$, $\widetilde{\sigma} \leftarrow \mathsf{pSign}_{sk}(m, Y)$, $\sigma := \mathsf{Adapt}_{pk}(\widetilde{\sigma}, y)$ and $y' := \mathsf{Ext}_{pk}(\sigma, \widetilde{\sigma}, Y)$. From Fig. 3 we know that $\widetilde{\sigma} = (h, \tilde{s})$, $\sigma = (h, s)$ and $y' = f_{\mathsf{ext}}(s, \tilde{s})$, where we have $s := f_{\mathsf{adapt}}(\tilde{s}, y)$, $\tilde{s} \leftarrow \mathsf{P}_2(sk, R_{\mathsf{pre}}, h, St)$, $h := \mathcal{H}(R_{\mathsf{sign}}, m)$, $R_{\mathsf{sign}} := f_{\mathsf{shift}}(R_{\mathsf{pre}}, Y)$ and $(R_{\mathsf{pre}}, St) \leftarrow \mathsf{P}_1(sk)$. We first show $\mathsf{pVrfy}_{pk}(m, Y; \widetilde{\sigma}) = 1$. From completeness of the ID scheme, we know that $\mathsf{V}_0(pk, h, \tilde{s}) = R_{\mathsf{pre}}$. Hence:

$$\mathcal{H}(f_{\mathsf{shift}}(\mathsf{V}_0(pk, h, \tilde{s}), Y), m) = \mathcal{H}(f_{\mathsf{shift}}(R_{\mathsf{pre}}, Y), m) = \mathcal{H}(R_{\mathsf{sign}}, m) = h \tag{3}$$

which is what we needed to prove. We now show that $\mathsf{Vrfy}_{pk}(m; \sigma) = 1$. By Fig. 2, we need to show that $h = \mathcal{H}(\mathsf{V}_0(pk, h, s), m)$. This follows from the property of f_{shift}, f_{adapt} (cf. Eq. (1)) and Eq. (3) as follows:

$$\begin{aligned} \mathcal{H}(\mathsf{V}_0(pk, h, s), m) =& \mathcal{H}(\mathsf{V}_0(pk, h, f_{\mathsf{adapt}}(\tilde{s}, y)), m) \\ \overset{(1)}{=}& \mathcal{H}(f_{\mathsf{shift}}(\mathsf{V}_0(pk, h, \tilde{s}), Y), m) \overset{(3)}{=} h. \end{aligned}$$

Finally, we need to show that $(Y, y') \in \mathsf{R}$. This follows from Eq. (2) since:

$$y' = f_{\mathsf{ext}}(s, \tilde{s}) = f_{\mathsf{ext}}(f_{\mathsf{adapt}}(\tilde{s}, y), \tilde{s}) \overset{(2)}{=} y.$$

Lemma 2 ($\mathsf{aEUF{-}CMA}$-Security). *Under the assumptions of Thm. 2,* $\mathsf{aSIG}^{\mathsf{ID},\mathsf{R}}$ *satisfies the* $\mathsf{aEUF{-}CMA}$ *security as for Def. 3.*

Let us give first a high level overview of the proof. Our goal is to provide a reduction such that, given an adversary $\mathcal{A}$ who can win the experiment $\mathsf{aSigForge}_{\mathcal{A},\mathsf{aSIG}^{\mathsf{ID},\mathsf{R}}}$, we can build a simulator who can win the $\mathsf{strongSigForge}$ experiment of the underlying signature or can break the hardness of the relation R. In the first case, we check if $\mathcal{A}$'s forgery σ^* is equal to $\mathsf{Adapt}_{pk}(\widetilde{\sigma}, y)$. If so, we use $\mathcal{A}$ to break the hardness of the relation R by extracting the witness $y = \mathsf{Ext}(\sigma^*, \widetilde{\sigma}, Y)$. Otherwise, $\mathcal{A}$ was able to forge a signature "unrelated" to the pre-signature provided to it. In this case, it is used to win the $\mathsf{strongSigForge}$ experiment. All that remains is to answer $\mathcal{A}$'s signing and pre-signing queries using $\mathsf{strongSigForge}$'s signing queries. This is done by programming the random oracle such that the full-signatures generated by the challenger in the $\mathsf{strongSigForge}$ game look like pre-signatures for $\mathcal{A}$.

Proof. We prove the lemma by defining a series of game hops. The modifications for each game hop is presented in code form in Appx. D.

Game $\boldsymbol{G_0}$: This game is the original $\mathsf{aSigForge}$ experiment (see Fig. 14), where the adversary $\mathcal{A}$ outputs a valid forgery σ^* for a message m of its choice, while having access to pre-signing and signing oracles $\mathcal{O}_{\mathrm{pS}}$ and $\mathcal{O}_{\mathrm{S}}$ respectively. Being in the random oracle model, all the algorithms of the scheme and the adversary have access to the random oracle $\mathcal{H}$. Since $\boldsymbol{G_0}$ corresponds to $\mathsf{aSigForge}$, it follows that $\Pr[\mathsf{aSigForge}_{\mathcal{A},\mathsf{aSIG}^{\mathsf{ID},\mathsf{R}}}(n) = 1] = \Pr[\boldsymbol{G_0} = 1]$.
Game $\boldsymbol{G_1}$: This game works as $\boldsymbol{G_0}$ except when the adversary outputs a forgery σ^*, the game checks if adapting the pre-signature $\widetilde{\sigma}$ using the secret witness y results in σ^*. If so, the game aborts.

Claim. Let Bad_1 be the event where $\boldsymbol{G_1}$ aborts. Then $\Pr[\mathsf{Bad}_1] \leq \nu_1(n)$, where ν_1 is a negligible function in n.

Proof: This claim is proven by a reduction to the relation R. We construct a simulator $\mathcal{S}$ which breaks the hardness of R using $\mathcal{A}$ that causes $\boldsymbol{G_1}$ to abort with non-negligible probability. The simulator receives a challenge Y^*, and generates a key pair $(sk, pk) \leftarrow \mathsf{Gen}(1^n)$ in order to simulate $\mathcal{A}$'s queries to the oracles $\mathcal{H}$, $\mathcal{O}_{\mathrm{pS}}$ and $\mathcal{O}_{\mathrm{S}}$. This simulation of the oracles work as described in $\boldsymbol{G_1}$.

Upon receiving the challenge message m^* from $\mathcal{A}$, $\mathcal{S}$ computes a pre-signature $\widetilde{\sigma} \leftarrow \mathsf{pSign}_{sk}(m^*, Y^*)$ and returns the pair $(\widetilde{\sigma}, Y)$ to the adversary. Upon $\mathcal{A}$ outputting a forgery σ^* and assuming that Bad_1 happened (i.e., $\mathsf{Adapt}(\widetilde{\sigma}, y) = \sigma$), pre-signature correctness (Def. 2) implies that the simulator can extract y^* by executing $\mathsf{Ext}(\sigma^*, \widetilde{\sigma}, Y^*)$ in order to obtain $(Y^*, y^*) \in \mathsf{R}$.

We note that the view of $\mathcal{A}$ in this simulation and in $\boldsymbol{G_1}$ are indistinguishable, since the challenge Y^* is an instance of the hard relation R and has the same distribution to the public output of GenR. Therefore, the probability that $\mathcal{S}$ breaks the hardness of R is equal to the probability that the event Bad_1 happens. Hence, we conclude that Bad_1 only happens with negligible probability. ■

Since games $\boldsymbol{G_1}$ and $\boldsymbol{G_0}$ are equivalent except if event Bad_1 occurs, it holds that $\Pr[\boldsymbol{G_0} = 1] \leq \Pr[\boldsymbol{G_1} = 1] + \nu_1(n)$.

Game $\boldsymbol{G_2}$: This game is similar to the previous game except for a modification in the $\mathcal{O}_{\mathrm{pS}}$ oracle. After the execution of $\mathsf{preSign}_{sk}$, the oracle obtains a pre-signature $\widetilde{\sigma}$ from which it extracts the randomness $R_{\mathsf{pre}} \leftarrow \mathsf{V}_0(pk, \widetilde{\sigma})$. The oracle computes $R_{\mathsf{sign}} = f_{\mathsf{shift}}(R_{\mathsf{pre}}, Y)$ and checks if $\mathcal{H}$ was already queried on the inputs $R_{\mathsf{pre}} \| m$ or $R_{\mathsf{sign}} \| m$ *before* the execution of pSign_{sk}. In this case the game aborts.

Claim. Let Bad_2 be the event that $\boldsymbol{G_2}$ aborts in $\mathcal{O}_{\mathrm{pS}}$. Then $\Pr[\mathsf{Bad}_2] \leq \nu_2(n)$, where ν_2 is a negligible function in n.

Proof: We first recall that the output of P_1 (i.e., R_{pre}) is uniformly random from a super-polynomial set of size q in the security parameter. From this it follows that R_{sign} is distributed uniformly at random in the same set. Furthermore, $\mathcal{A}$ being a PPT algorithm, it can only make polynomially many queries to $\mathcal{H}$, $\mathcal{O}_{\mathrm{S}}$ and $\mathcal{O}_{\mathrm{pS}}$ oracles. Denoting ℓ as the total number of queries to $\mathcal{H}$, $\mathcal{O}_{\mathrm{S}}$ and $\mathcal{O}_{\mathrm{pS}}$, we have: $\Pr[\mathsf{Bad}_2] = \Pr[H'[R_{\mathsf{pre}} || m] \neq \bot \vee H'[R_{\mathsf{sign}} || m] \neq \bot] \leq 2\frac{\ell}{q} \leq \nu_2(n)$. This follows from the fact that ℓ is polynomial in the security parameter. ■

Since games $\boldsymbol{G_2}$ and $\boldsymbol{G_1}$ are identical except in the case where Bad_2 occurs, it holds that $\Pr[\boldsymbol{G_1} = 1] \leq \Pr[\boldsymbol{G_2} = 1] + \nu_2(n)$.

Game $\boldsymbol{G_3}$: In this game, upon a query to the $\mathcal{O}_{\mathrm{pS}}$, the game produces a full-signature instead of a pre-signature by executing Sign_{sk} instead of $\mathsf{preSign}_{sk}$. Accordingly, it programs the random oracle $\mathcal{H}$ to make the full-signature "look like" a pre-signature from the point of view of the adversary $\mathcal{A}$. This is done by:

1. It sets $\mathcal{H}(R_{\mathsf{pre}} \| m)$ to the value stored at position $\mathcal{H}(R_{\mathsf{sign}} \| m)$.
2. It sets $\mathcal{H}(R_{\mathsf{sign}} \| m)$ to a fresh value chosen uniformly at random.

The above programming makes sense as our definition of f_{shift} requires it to be deterministic and to possess the same domain and codomain with respect to the commitment set $\mathcal{D}_{\mathsf{rand}}$. Note further that $\mathcal{A}$ can only notice that $\mathcal{H}$ was programmed if it was previously queried on either $R_{\mathsf{pre}}\|m$ or $R_{\mathsf{sign}}\|m$. But as described in the previous game, we abort if such an event happens. Hence, we have that $\Pr[\boldsymbol{G_2} = 1] = \Pr[\boldsymbol{G_3} = 1]$.

Game $\boldsymbol{G_4}$: In this game, we impose new checks during the challenge phase that are same as the ones imposed in $\boldsymbol{G_2}$ during the execution of $\mathcal{O}_{\mathrm{pS}}$.

Claim. Let Bad_3 be the event that $\boldsymbol{G_4}$ aborts in the challenge phase. Then $\Pr[\mathsf{Bad}_3] \leq \nu_3(n)$, where ν_3 is a negligible function in n.

Proof: The proof is identical to the proof in $\boldsymbol{G_2}$. ■

It follows that $\Pr[\boldsymbol{G_4} = 1] \leq \Pr[\boldsymbol{G_3} = 1] + \nu_3(n)$.

Game $\boldsymbol{G_5}$: Similar to game $\boldsymbol{G_3}$, we generate a signature instead of a pre-signature in the challenge phase and program $\mathcal{H}$ such that the full-signature looks like a correct pre-signature from $\mathcal{A}$'s point of view. We get $\Pr[\boldsymbol{G_5} = 1] = \Pr[\boldsymbol{G_4} = 1]$.

The modifications from games $\boldsymbol{G_1}$ - $\boldsymbol{G_5}$ in code form can be found in Fig. 15. Now that the transition from the original $\mathsf{aSigForge}$ experiment (game $\boldsymbol{G_0}$) to game $\boldsymbol{G_5}$ is indistinguishable, it only remains to show the existence of a simulator $\mathcal{S}$ that can perfectly simulate $\boldsymbol{G_5}$ and uses $\mathcal{A}$ to win the $\mathsf{strongSigForge}$ game. In Fig. 16, we describe the simulator's code in a concise way.

We emphasize that the main differences between the simulation as shown in Fig. 16 and Game $\boldsymbol{G_5}$ are syntactical. Namely, instead of generating the public and secret keys and computing the algorithm Sign_{sk} and the random oracle $\mathcal{H}$, $\mathcal{S}$ uses its oracles $\mathsf{SIG}^{\mathsf{ID}}$ and $\mathcal{H}^{\mathsf{ID}}$. Therefore, $\mathcal{S}$ perfectly simulates $\boldsymbol{G_5}$. It remains to show that $\mathcal{S}$ can use the forgery output by $\mathcal{A}$ to win the $\mathsf{strongSigForge}$ game.

Claim. (m^*, σ^*) constitutes a valid forgery in game $\mathsf{strongSigForge}$.

Proof: To prove this claim, we show that the tuple (m^*, σ^*) has not been returned by the oracle $\mathsf{SIG}^{\mathsf{ID}}$ before. First note that $\mathcal{A}$ wins the experiment if it has not queried on the challenge message m^* to $\mathcal{O}_{\mathrm{pS}}$ or $\mathcal{O}_{\mathrm{S}}$. Therefore, $\mathsf{SIG}^{\mathsf{ID}}$ is queried on m^* only during the challenge phase. If $\mathcal{A}$ outputs a forgery σ^* that is equal to the signature σ as output by $\mathsf{SIG}^{\mathsf{ID}}$, it would lose the game since this signature is not valid given the fact that $\mathcal{H}$ is programmed.

Hence, $\mathsf{SIG}^{\mathsf{ID}}$ has never output σ^* when queried on m^* before, thus making (m^*, σ^*) a valid forgery for game $\mathsf{strongSigForge}$. ■

From games $\boldsymbol{G_0} - \boldsymbol{G_5}$, we have that $\Pr[\boldsymbol{G_0} = 1] \leq \Pr[\boldsymbol{G_5} = 1] + \nu(n)$, where $\nu(n) = \nu_1(n) + \nu_2(n) + \nu_3(n)$ is a negligible function in n. Since $\mathcal{S}$ simulates game $\boldsymbol{G_5}$ perfectly, we also have that $\Pr[\boldsymbol{G_5} = 1] = \Pr[\mathsf{strongSigForge}_{\mathcal{S}^{\mathcal{A}},\mathsf{SIG}}(n) = 1]$. Combining this with the probability statement in $\boldsymbol{G_0}$, we obtain the following:

$\Pr[\mathsf{aSigForge}_{\mathcal{A},\mathsf{aSIG}^{\mathsf{ID},\mathsf{R}}}(n) = 1] \leq \Pr[\mathsf{strongSigForge}_{\mathcal{S}^{\mathcal{A}},\mathsf{SIG}^{\mathsf{ID}}}(n) = 1] + \nu(n)$.

Recall that the negligible function $\nu_1(n)$, contained in the sum $\nu(n)$ above, precisely quantifies the adversary's advantage in breaking the hard relation R. Thus, the probability of breaking the unforgeability of the $\mathsf{aSIG}^{\mathsf{ID},\mathsf{R}}$ is clearly bounded above by that of breaking either R or the strong unforgeability of $\mathsf{SIG}^{\mathsf{ID}}$.

Lemma 3 (Pre-Signature Adaptability). *Under the assumptions of Thm. 2,* $\mathsf{aSIG}^{\mathsf{ID},\mathsf{R}}$ *satisfies the pre-signature adaptability as for Def. 4.*

Proof. Assume $\mathsf{pVrfy}_{pk}(m, Y; \widetilde{\sigma}) = 1$, with the notations having their usual meanings from Fig. 3, which means $h = \mathcal{H}(f_{\mathsf{shift}}(\mathsf{V}_0(pk, h, \tilde{s}), Y), m)$. For any valid pair $(Y, y) \in R$, we can use the homomorphic property from Eq. (1). Then, for such a pair $(Y, y) \in R$, plugging $f_{\mathsf{shift}}(\mathsf{V}_0(pk, h, \tilde{s}), Y) = \mathsf{V}_0(pk, h, f_{\mathsf{adapt}}(\tilde{s}, y))$ in the above equation implies $h = \mathcal{H}(\mathsf{V}_0(pk, h, f_{\mathsf{adapt}}(\tilde{s}, y)), m)$. This directly implies $\mathsf{Vrfy}_{pk}(m; \sigma) = 1$, where $s = f_{\mathsf{adapt}}(\tilde{s}, y)$ and $\sigma = (h, s)$. Therefore, adapting the valid pre-signature would also result in a valid full-signature.

Lemma 4 (Witness Extractability). *Under the assumptions of Thm. 2,* $\mathsf{aSIG}^{\mathsf{ID},\mathsf{R}}$ *satisfies the witness extractability as for Def. 5.*

This proof is very similar to the proof of Lemma 2 with the mere difference that we only need to provide a reduction to the $\mathsf{strongSigForge}$ experiment. This is because in the $\mathsf{aWitExt}_{\mathcal{A},\mathsf{aSIG}_{\mathsf{R}_g},\mathsf{SIG}^{\mathsf{ID}}}$ experiment, $\mathcal{A}$ provides the public value Y^* and must forge a valid full-signature σ^* such that $(Y^*, \mathsf{Ext}_{pk}(\sigma^*, \widetilde{\sigma}, Y^*)) \notin \mathsf{R}$. The full proof can be found in Appx. C.

Remark 2. We note that our proofs for the $\mathsf{aEUF{-}CMA}$ security and witness extractability are in its essence reductions to the strong unforgeability of the underlying signature schemes. Yet the Fiat-Shamir transformation does not immediately guarantee the resulting signature scheme to be strongly unforgeable. However, we first note that many such signature schemes are indeed strongly unforgeable, for instance Schnorr [27], Katz-Wang (from Chaum-Pedersen identification scheme) [26] and Guillou-Quisquater [1] signature schemes all satisfy strong unforgeability. Moreover, one can transform any Fiat-Shamir based existentially unforgeable signature scheme into a strongly unforgeable one via the generic transformation using the results of Bellare et.al. [4].

4 Two-party Signatures with Aggregatable Public Keys from Identification Schemes

Before providing our definition and generic transformation for two-party adaptor signatures, we show how to generically transform signature schemes based on identification schemes into two-party signature schemes with aggregatable public keys denoted by SIG_2. In Sec. 5, we then combine the techniques used in this section with the ones from Sec. 3 in order to generically transform identification schemes into two-party adaptor signature schemes.

Informally, a SIG_2 scheme allows two parties to jointly generate a signature which can be verified under their combined public keys. An application of such signature schemes can be found in cryptocurrencies where two parties wish to only allow conditional payments such that both users have to sign a transaction in order to spend some funds. Using SIG_2, instead of submitting two separate signatures, the parties can submit a single signature while enforcing the same condition (i.e., a transaction must have a valid signature under the combined key) and hence reduce the communication necessary with the blockchain. Importantly and unlike threshold signature schemes, the key generation here is non-interactive. In other words, parties generate their public and secret keys independently and anyone who knows both public keys can compute the joint public key of the two parties.

We use the notation $\Pi_{\mathsf{Func}\langle x_i, x_{1-i}\rangle}$ to represent a two-party interactive protocol Func between P_i and P_{1-i} with respective secret inputs x_i, x_{1-i} for $i \in \{0,1\}$. Furthermore, if there are common public inputs e.g., $y_1, \cdots, y_n$ we use the notation $\Pi_{\mathsf{Func}\langle x_i, x_{1-i}\rangle}(y_1, \cdots, y_n)$. We note that the execution of a protocol might not be symmetric, i.e., party $\mathcal{P}_i$ executes the procedures $\Pi_{\mathsf{Func}\langle x_i, x_{1-i}\rangle}$ while party $\mathcal{P}_{1-i}$ executes the procedures $\Pi_{\mathsf{Func}\langle x_{1-i}, x_i\rangle}$.

4.1 Two-party Signatures with Aggregatable Public Keys

We start with defining a two-party signature scheme with aggregatable public keys. Our definition is inspired by the definitions from prior works [9, 28, 8].

Definition 7 (Two-party Signature with Aggregatable Public Keys). *A two-party signature scheme with aggregatable public keys is a tuple of PPT protocols and algorithms* $\mathsf{SIG}_2 = (\mathsf{Setup}, \mathsf{Gen}, \Pi_{\mathsf{Sign}}, \mathsf{KAg}, \mathsf{Vrfy})$*, formally defined as:*

$\mathsf{Setup}(1^n)$**:** *is a PPT algorithm that on input a security parameter n, outputs public parameters pp.*
$\mathsf{Gen}(pp)$**:** *is a PPT algorithm that on input public parameter pp, outputs a key pair (sk, pk).*
$\Pi_{\mathsf{Sign}\langle sk_i, sk_{1-i}\rangle}(pk_0, pk_1, m)$**:** *is an interactive, PPT protocol that on input secret keys sk_i from party $\mathcal{P}_i$ with $i \in \{0,1\}$ and common values $m \in \{0,1\}^*$ and pk_0, pk_1, outputs a signature σ.*
$\mathsf{KAg}(pk_0, pk_1)$**:** *is a DPT algorithm that on input two public keys pk_0, pk_1, outputs an aggregated public key apk.*

$\mathsf{Vrfy}_{apk}(m;\sigma)$**:** *is a DPT algorithm that on input public parameters pp, a public key apk, a message $m \in \{0,1\}^*$ and a signature σ, outputs a bit b.*

The *completeness* property of SIG_2 guarantees that if the protocol Π_{Sign} is executed correctly between the two parties, the resulting signature is a valid signature under the aggregated public key.

Definition 8 (Completeness). *A two-party signature with aggregatable public keys SIG_2 satisfies completeness, if for all key pairs $(sk, pk) \leftarrow \mathsf{Gen}(1^n)$ and messages $m \in \{0,1\}^*$, the protocol $\Pi_{\mathsf{Sign}\langle sk_i, sk_{1-i}\rangle}(pk_0, pk_1, m)$ outputs a signature σ to both parties $\mathcal{P}_0, \mathcal{P}_1$ such that $\mathsf{Vrfy}_{apk}(m;\sigma) = 1$ where $apk := \mathsf{KAg}(pk_0, pk_1)$.*

A two-party signature scheme with aggregatable public keys should satisfy *unforgeability.* At a high level, this property guarantees that if one of the two parties is malicious, this party is not able to produce a valid signature under the aggregated public key without cooperation of the other party. We formalize the property through an experiment $\mathsf{SigForge}^b_{\mathcal{A},\mathsf{SIG}_2}$, where $b \in \{0,1\}$ defines which of the two parties is corrupt.This experiment is initialized by a security parameter n and run between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, which proceeds as follows. The challenger first generates the public parameters pp by running the setup procedure $\mathsf{Setup}(1^n)$ as well as a signing key pair (sk_{1-b}, pk_{1-b}) by executing $\mathsf{Gen}(1^n)$, thereby simulating the honest party $\mathcal{P}_{1-b}$. Thereafter, $\mathcal{C}$ forwards pp_{C} and pk_{1-b} to the adversary $\mathcal{A}$ who generates its own key pair (sk_b, pk_b), thereby emulating the malicious party P_b, and submits (sk_b, pk_b) to $\mathcal{C}$. The adversary $\mathcal{A}$ additionally obtains access to an *interactive* and stateful signing oracle $\mathcal{O}^b_{\Pi_\mathrm{S}}$, which simulates the honest party $\mathcal{P}_{1-b}$ during the execution of $\Pi^{\mathcal{A}}_{\mathsf{Sign}\langle sk_{1-b},\cdot\rangle}$. Furthermore, every queried message m is stored in a query list $\mathcal{Q}$.

Eventually, $\mathcal{A}$ outputs a forgery in form of a $\mathsf{SIG}_2^{\mathsf{ID}}$ signature σ^* and a message m^*. $\mathcal{A}$ wins the experiment if σ^* is a valid signature for m^* under the aggregated public key $apk := \mathsf{KAg}(pk_0, pk_1)$ and m^* was never queried before, i.e., $m^* \notin \mathcal{Q}$. Below, we give a formal definition of the unforgeability game.

Definition 9 (2-$\mathsf{EUF\text{–}CMA}$ Security). *A two-party, public key aggregatable signature scheme SIG_2 is unforgeable if for every PPT adversary $\mathcal{A}$, there exists a negligible function ν such that: for $b \in \{0,1\}$, $\Pr[\mathsf{SigForge}^b_{\mathcal{A},\mathsf{SIG}_2}(n) = 1] \leq \nu(n)$, where the experiment $\mathsf{SigForge}^b_{\mathcal{A},\mathsf{SIG}_2}(n)$ is defined as follows:*

```
$\mathsf{SigForge}^b_{\mathcal{A},\mathsf{SIG}_2}(n)$
1 : $\mathcal{Q} := \emptyset, pp \leftarrow \mathsf{Setup}(1^n)$
2 : $(sk_{1-b}, pk_{1-b}) \leftarrow \mathsf{Gen}(pp)$
3 : $(sk_b, pk_b) \leftarrow \mathcal{A}(pp, pk_{1-b})$
4 : $(\sigma^*, m^*) \leftarrow \mathcal{A}^{\mathcal{O}^b_{\Pi_\mathrm{S}}(\cdot)}(pk_{1-b}, sk_b, pk_b)$
5 : return $\left(m^* \notin \mathcal{Q} \land \mathsf{Vrfy}_{\mathsf{KAg}(pk_0,pk_1)}(m^*;\sigma^*)\right)$

$\mathcal{O}^b_{\Pi_\mathrm{S}}(m)$
1 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
2 : $\sigma \leftarrow \Pi^{\mathcal{A}}_{\mathsf{Sign}\langle sk_{1-b},\cdot\rangle}(pk_0, pk_1, m)$
```

Remark 3 (On security definition.). There are two different approaches for modeling signatures with aggregatable public keys in the literature, namely the plain public-key model [3] (also known as key-verification model [15]) and the knowledge-of-secret-key (KOSK) model [8]. In the plain public-key setting the adversary chooses a key pair (sk_b, pk_b) and only declares the public key pk_b to the challenger in the security game. However, security proofs in this setting typically require rewinding techniques with the forking lemma. This is undesirable for the purpose of this paper, as we aim to construct adaptor signatures and its two-party variant generically as building blocks for further applications such as payment channels [2]. Payment channels are proven secure in the UC framework that does not allow the use of rewinding techniques in order to ensure concurrency. Thus, the plain public-key model does not seem suitable for our purpose. In the KOSK setting, however, the adversary outputs its (possibly maliciously chosen) key pair (sk_b, pk_b) to the challenger. In practice this means that the parties need to exchange zero-knowledge proofs of knowledge of their secret key[3]. Similar to previous works [8, 30], we do not require the forking lemma or rewinding in the KOSK setting and hence follow this approach.

[3] Using techniques from [22, 18] it is possible to obtain NIZKs which allow for witness extraction without rewinding.

4.2 Generic Transformation from $\mathsf{SIG}^{\mathsf{ID}}$ to $\mathsf{SIG}_2^{\mathsf{ID}}$

We now give a generic transformation from $\mathsf{SIG}^{\mathsf{ID}}$ schemes to two-party signature schemes with aggregatable public keys.

At a high level, our transformation turns the signing procedure into an interactive protocol which is executed between the two parties $\mathcal{P}_0, \mathcal{P}_1$. The main idea is to let both parties engage in a randomness exchange protocol in order to generate a joint public randomness which can then be used for the signing procedure. In a bit more detail, to create a joint signature, each party $\mathcal{P}_i$ for $i \in \{0, 1\}$ can individually create a partial signature with respect to the *joint randomness* by using the secret key sk_i and exchange her partial signature with $\mathcal{P}_{1-i}$. The joint randomness ensures that both partial signatures can be combined to one jointly computed signature.

In the following, we describe the randomness exchange protocol that is executed during the signing procedure in more detail, as our transformation heavily relies on it. The protocol, denoted by $\Pi_{\mathsf{Rand\text{-}Exc}}$, makes use of two cryptographic building blocks, namely an extractable commitment scheme $\mathsf{C} = (\mathsf{Gen}, \mathsf{Com}, \mathsf{Dec}, \mathsf{Extract})$ and a NIZK proof system $\mathsf{NIZK} = (\mathsf{Setup_R}, \mathsf{Prove}, \mathsf{Verify})$. Consequently, the common input to both parties $\mathcal{P}_0$ and $\mathcal{P}_1$ are the public parameters pp_{C} of the commitment scheme, while each party P_i takes as secret input her secret key sk_i. In the following, we give description of the $\Pi_{\mathsf{Rand\text{-}Exc}\langle sk_0, sk_1\rangle}(pp_{\mathsf{C}}, \mathsf{crs})$ protocol and present it in a concise way in Fig. 5.

$$
\begin{array}{lcl}
\mathcal{P}_0(pp_{\mathsf{C}}, \mathsf{crs}, sk_0) & & \mathcal{P}_1(pp_{\mathsf{C}}, \mathsf{crs}, sk_1) \\
\hline
(R_0, St_0) \leftarrow \mathsf{P}_1(sk_0) & & \\
\pi_0 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, R_0, sk_0) & & \\
(c, d) \leftarrow \mathsf{C.Com}(pp_{\mathsf{C}}, (R_0, \pi_0)) & \xrightarrow{\quad c \quad} & (R_1, St_1) \leftarrow \mathsf{P}_1(sk_1) \\
 & \xleftarrow{\quad R_1, \pi_1 \quad} & \pi_1 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, R_1, sk_1) \\
 & \xrightarrow{\quad d \quad} & R_0' \leftarrow \mathsf{C.Dec}(pp_{\mathsf{C}}, c, d) \\
\text{If } \mathsf{NIZK.Verify}(\mathsf{crs}, R_1, \pi_1) = 0, \text{ then abort} & & \text{If } \mathsf{NIZK.Verify}(\mathsf{crs}, R_0', \pi_0) = 0, \text{ then abort} \\
\hline
R_0, St_0, R_1 & & R_1, St_1, R_0
\end{array}
$$

Fig. 5: $\Pi_{\mathsf{Rand\text{-}Exc}}$ Protocol

1. Party $\mathcal{P}_0$ generates her public randomness R_0 using algorithm P_1 from the underlying ID scheme alongside a NIZK proof $\pi_0 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, R_0, sk_0)$ that this computation was executed correctly with the corresponding secret value sk_0. $\mathcal{P}_0$ executes $(c, d) \leftarrow \mathsf{C.Com}(pp, (R_0, \pi_0))$ to commit to R_0 and π_0 and sends the commitment c to $\mathcal{P}_1$.
2. Upon receiving the commitment c from $\mathcal{P}_0$, party $\mathcal{P}_1$ generates her public randomness R_1 using algorithm P_1. She also computes a NIZK proof as $\pi_1 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, R_1, sk_1)$, which proves correct computation of R_1, and sends R_1 and π_1 to $\mathcal{P}_0$.
3. Upon receiving R_1 and π_1 from $\mathcal{P}_1$, $\mathcal{P}_0$ sends the opening d to her commitment c to $\mathcal{P}_1$.
4. $\mathcal{P}_1$ opens the commitment in this round. At this stage, both parties check that the received zero-knowledge proofs are valid. If the proofs are valid, each party $\mathcal{P}_i$ for $i \in \{0, 1\}$ outputs R_i, St_i, R_{1-i}.

Our transformation can be found in Fig. 6. Note that we use a deterministic function $f_{\mathsf{com\text{-}rand}}(\cdot, \cdot)$ in step 3 in the signing protocol which combines the two public random values R_0 and R_1. In step 6 of the same protocol, we assume that the partial signatures are exchanged between the parties via the protocol Π_{Exchange} upon which the parties can combine them using a deterministic function $f_{\mathsf{com\text{-}sig}}(\cdot, \cdot)$ in step 7. Further, a combined signature can be verified under a combined public key of the two parties. In more detail, to verify a combined signature $(h, s) := f_{\mathsf{com\text{-}sig}}(h, (s_0, s_1))$, in step 7, there must exist an additional deterministic function $f_{\mathsf{com\text{-}pk}}(\cdot, \cdot)$ (in step 1 of the KAg algorithm) such that:

$$
\Pr\left[\mathsf{Vrfy}_{apk}(m; (h, s)) = 1 \;\middle|\; \begin{array}{l} (pk_0, sk_0) \leftarrow \mathsf{IGen}(n), (pk_1, sk_1) \leftarrow \mathsf{IGen}(n) \\ (h, s) \leftarrow \Pi_{\mathsf{Sign}\langle sk_0, sk_1\rangle}(pk_0, pk_1, m) \\ apk := f_{\mathsf{com\text{-}pk}}(pk_0, pk_1) \end{array}\right] = 1. \quad (4)
$$

```
$\mathsf{Setup}(1^n)$
    1 : $pp_{\mathsf{C}} \leftarrow \mathsf{C.Gen}(1^n)$
    2 : $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup_R}(1^n)$
    3 : return $pp := (1^n, pp_{\mathsf{C}}, \mathsf{crs})$

$\mathsf{Gen}(pp)$
    1 : Parse $pp = (1^n, pp_{\mathsf{C}}, \mathsf{crs})$
    2 : $(sk, pk') \leftarrow \mathsf{IGen}(n)$
    3 : $pk := (pp, pk')$
    4 : return $(sk, pk)$

$\mathsf{KAg}(pk_0, pk_1)$
    1 : $apk := f_{\mathsf{com\text{-}pk}}(pk_0, pk_1)$
    2 : return $apk$

$\Pi_{\mathsf{Sign}\langle sk_i, sk_{1-i}\rangle}(pk_i, pk_{1-i}, m)$
    1 : Parse $pk_i = ((1^n, pp_{\mathsf{C}}, \mathsf{crs}), pk'_i)$
    2 : $(R_i, St_i, R_{1-i}) \leftarrow \Pi_{\mathsf{Rand\text{-}Exc}\langle sk_i, sk_{1-i}\rangle}(pp_{\mathsf{C}}, \mathsf{crs})$
    3 : $R_{\mathsf{sign}} := f_{\mathsf{com\text{-}rand}}(R_0, R_1)$
    4 : $h := \mathcal{H}(R_{\mathsf{sign}}, m)$
    5 : $s_i \leftarrow \mathsf{P}_2(sk_i, R_i, h, St_i)$
    6 : $s_{1-i} \leftarrow \Pi_{\mathsf{Exchange}}\langle s_i, s_{1-i}\rangle$
    7 : $(h, s) := f_{\mathsf{com\text{-}sig}}(h, (s_0, s_1))$
    8 : return $(h, s)$

$\mathsf{Vrfy}_{apk}(m; (h, s))$
    1 : $R_{\mathsf{sign}} := \mathsf{V}_0(apk, h, s)$
    2 : return $h := \mathcal{H}(R_{\mathsf{sign}}, m)$
```

Fig. 6: $\mathsf{SIG}_2^{\mathsf{ID}}$: SIG_2 scheme from identification scheme.

We also require that given a full signature and a secret key sk_i with $i \in \{0, 1\}$, it is possible to extract a valid partial signature under the the public key pk_{1-i} of the other party. In particular, there exists a function $f_{\mathsf{dec\text{-}sig}}(\cdot, \cdot, \cdot)$ such that:

$$\Pr\left[\mathsf{Vrfy}_{pk_{1-i}}(m; (h, s_{1-i})) = 1 \,\middle|\, \begin{array}{l} (pk_0, sk_0) \leftarrow \mathsf{IGen}(n), (pk_1, sk_1) \leftarrow \mathsf{IGen}(n) \\ (h, s) \leftarrow \Pi_{\mathsf{Sign}\langle sk_0, sk_1\rangle}(pk_0, pk_1, m) \\ (h, s_{1-i}) := f_{\mathsf{dec\text{-}sig}}(sk_i, pk_i, (h, s)) \end{array}\right] = 1. \tag{5}$$

Note that equations 4 and 5 implicitly define $f_{\mathsf{com\text{-}sig}}$ through the execution of Π_{Sign} in the conditional probabilities.

The instantiations of these functions for Schnorr, Katz-Wang signatures and Guillou-Quisquater signatures can be found in Appx. B.

We note the similarity between this transformation with that in Fig. 3. In particular, both of them compute the public randomness R_{sign} by shifting the original random values. Note also that running the algorithm V_0 on the inputs (pk_i, h, s_i) would return $R_i, \forall i \in \{0, 1\}$.

Below, we show that the transformation in Fig. 6 provides a secure two-party signature with aggregatable public keys. To this end, we show that $\mathsf{SIG}_2^{\mathsf{ID}}$ satisfies SIG_2 *completeness* and *unforgeability* from Def. 8 and Def. 9, respectively.

Theorem 3. *Assume that $\mathsf{SIG}^{\mathsf{ID}}$ is a signature scheme based on the transformation from an identification scheme as per Fig. 2. Further, assume that the functions $f_{\mathsf{com\text{-}sig}}$, $f_{\mathsf{com\text{-}pk}}$ and $f_{\mathsf{dec\text{-}sig}}$ satisfy the relations, Equations (4) and (5) respectively. Then the resulting $\mathsf{SIG}_2^{\mathsf{ID}}$ scheme from the transformation in Fig. 6 is a secure two-party signature scheme with aggregatable public keys in the random oracle model.*

Lemma 5. *Under the assumptions of Thm. 3, $\mathsf{SIG}_2^{\mathsf{ID}}$ satisfies Def. 8.*

Proof. The proof follows directly from Eq. 4 and the construction of KAg algorithm in Fig. 6.

Lemma 6. *Under the assumptions of Thm. 3, $\mathsf{SIG}_2^{\mathsf{ID}}$ satisfies Def. 9.*

Proof. We prove this lemma by exhibiting a simulator $\mathcal{S}$ that breaks the unforgeability of the $\mathsf{SIG}^{\mathsf{ID}}$ scheme if it has access to an adversary that can break the unforgeability of the $\mathsf{SIG}_2^{\mathsf{ID}}$ scheme. More precisely, we show a series of games, starting with the $\mathsf{SigForge}^{b}_{\mathcal{A},\mathsf{SIG}_2}$ experiment, such that each game is computationally indistinguishable from the previous one. The last game is modified in such a way that the simulator can use the adversary's forgery to create its own forgery for the unforgeability game against the $\mathsf{SIG}^{\mathsf{ID}}$ scheme.

To construct this simulator, we note that the $\Pi_{\mathsf{Rand\text{-}Exc}}$ protocol in Fig. 6 must satisfy two properties (similar to [29]). First, the commitment scheme must be extractable for the simulator, and second, the NIZK proof used must be simulatable. The reasons for these two properties become evident in the proof.

We prove Lemma 6 by separately considering the cases of the adversary corrupting party $\mathcal{P}_0$ or party $\mathcal{P}_1$, respectively.

Adversary corrupts $\mathcal{P}_0$. In the following we give the security proof in case the adversary corrupts party $\mathcal{P}_0$.
Game G_0: This is the regular $\mathsf{SigForge}^{0}_{\mathcal{A},\mathsf{SIG}_2}(n)$ experiment, in which the adversary plays the role of party $\mathcal{P}_0$. In the beginning of the game, the simulator generates the public parameters as $pp \leftarrow \mathsf{Setup}(1^n)$. Note that the Setup procedure, apart from computing $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}_{\mathsf{R}}(1^n)$, includes the execution of $\mathsf{C.Gen}$ through which the simulator learns the trapdoor tr for the commitment scheme C. Further, $\mathcal{S}$ generates a fresh signing key pair $(sk_1, pk_1) \leftarrow \mathsf{Gen}(1^n)$, sends pp and pk_1 to $\mathcal{A}$ and receives the adversary's key pair (pk_0, sk_0). The simulator simulates the experiment honestly. In particular, it simulates the interactive signing oracle $\mathcal{O}^{0}_{\Pi_{\mathrm{S}}}$ honestly by playing the role of party $\mathcal{P}_1$.

Game G_1: This game proceeds exactly like the previous game, with a modification in the simulation of the signing oracle. Upon $\mathcal{A}$ initiating the signing protocol by calling the interactive signing oracle, $\mathcal{S}$ receives the commitment c to its public randomness R_0 from $\mathcal{A}$. The simulator, using the trapdoor tr, then extracts a randomness $R_0' \leftarrow \mathsf{C.Extract}(pp, tr, c)$ and computes the joint randomness as $R_{\mathsf{sign}} \leftarrow f_{\mathsf{com\text{-}rand}}(R_0', R_1)$. $\mathcal{S}$ honestly computes the zero-knowledge proof to its own randomness R_1 and sends it to $\mathcal{A}$. Upon receiving the opening d to c from the adversary, $\mathcal{S}$ checks if $R_0' = \mathsf{C.Dec}(pp, c, d)$. If this does not hold, $\mathcal{S}$ aborts, otherwise $\mathcal{S}$ continues to simulate the rest of the experiment honestly.

Claim. Let Bad_1 be the event that G_1 aborts in the signing oracle. Then, we have $\Pr[\mathsf{Bad}_1] \leq \nu_1(n)$, where ν_1 is a negligible function in n.

Proof: Note that game G_1 aborts only if the extracted value R_0' from commitment c is not equal to the actual committed value R_0 in c, i.e., if $\mathsf{C.Extract}(pp, tr, c) \neq \mathsf{C.Dec}(pp, c, d)$. By the extractability property of C this happens only with negligible probability. In other words, it holds that $\Pr[\mathsf{Bad}_1] \leq \nu_1(n)$, where ν_1 is a negligible function in n. ■

Game G_2: This game proceeds as game G_1, with a modification to the signing oracle. Upon input message m, instead of generating its signature (h, s_0) with respect to the joint public randomness R_{sign}, the simulator generates it only with respect to its own randomness R_0. Further, the simulator programs the random oracle in the following way: as in the previous game, it computes the joint randomness R_{sign} and then programs the random oracle in a way such that on input (R_{sign}, m) the random oracle returns h.

It is easy to see that this game is indistinguishable from G_1 if the adversary has not queried the random oracle on input (R_{sign}, m) before the signing query. If, however, the adversary has issued this random oracle query before the signing query (i.e., $\mathcal{H}(R_{\mathsf{sign}}, m) \neq \bot$)), then the simulation aborts.

Claim. Let Bad_2 be the event that G_2 aborts in the signing oracle. Then, we have $\Pr[\mathsf{Bad}_2] \leq \nu_2(n)$, where ν_2 is a negligible function in n.

Proof: We first recall that the output of P_1 (i.e., R_{pre}) is uniformly random from a super-polynomial set of size q in the security parameter. From this it follows that R_{sign} is distributed uniformly at random in the same set. Furthermore, $\mathcal{A}$ being a PPT algorithm, can only make polynomially many queries to $\mathcal{H}$ and $\mathcal{O}_{\mathrm{pS}}$ oracles. Denoting ℓ as the total number of queries to $\mathcal{H}$ and $\mathcal{O}_{\mathrm{S}}$, we have: $\Pr[\mathsf{Bad}_2] = \Pr[\mathcal{H}(R_{\mathsf{sign}}, m) \neq \bot] \leq \frac{\ell}{q} \leq \nu_2(n)$. This follows from the fact that ℓ is polynomial in the security parameter. ■

Game G_3: In this game, the only modification as compared to the previous game is that during the Setup procedure, the simulator executes the algorithm $(\widetilde{\mathsf{crs}}, \tau) \leftarrow \mathsf{NIZK.Setup}'_{\mathsf{R}}(1^n)$ instead of $\mathsf{crs} \leftarrow \mathsf{Setup}_{\mathsf{R}}(1^n)$, which allows the simulator to learn the trapdoor τ. Since the two distributions $\{\mathsf{crs} : \mathsf{crs} \leftarrow \mathsf{Setup}_{\mathsf{R}}(1^n)\}$ and $\{\widetilde{\mathsf{crs}} : (\widetilde{\mathsf{crs}}, \tau) \leftarrow \mathsf{Setup}'_{\mathsf{R}}(1^n)\}$ are indistinguishable to $\mathcal{A}$ except with negligible probability, we have that $\Pr[\boldsymbol{G_2} = 1] \leq \Pr[\boldsymbol{G_3} = 1] + \nu_3(n)$ where ν_3 is a negligible function in n.

Game $\boldsymbol{G_4}$: This game proceeds exactly like the previous game except that the simulator does not choose its own key pair, but rather uses its signing oracle from the EUF–CMA game to simulate the adversary's interactive signing oracle $\mathcal{O}^0_{\Pi_{\mathsf{S}}}$. More concretely, upon the adversary calling $\mathcal{O}^0_{\Pi_{\mathsf{S}}}$ on message m, the simulator calls its own signing oracle which provides a signature (h, s_1) for m under secret key sk_1. Note that the simulator does not know sk_1 or the secret randomness r_1 used in s_1. Therefore, the simulator has to additionally simulate the NIZK proof that proves knowledge of r_1 in s_1. More concretely, the simulator executes $\pi_{\mathsf{S}} \leftarrow \mathsf{S}(\widetilde{\mathsf{crs}}, \tau, R_1)$, where R_1 is the public randomness used in s_1. Due to the fact that the distributions $\{\pi : \pi \leftarrow \mathsf{Prove}(\widetilde{\mathsf{crs}}, R_1, r_1)\}$ and $\{\pi_{\mathsf{S}} : \pi_{\mathsf{S}} \leftarrow \mathsf{S}(\widetilde{\mathsf{crs}}, \tau, R_1)\}$ are indistinguishable to $\mathcal{A}$ except with negligible probability, it holds that $\Pr[\boldsymbol{G_3} = 1] \leq \Pr[\boldsymbol{G_4} = 1] + \nu_4(n)$ where ν_4 is a negligible function in n.

It remains to show that the simulator can use a valid forgery output by $\mathcal{A}$ to break unforgeability of the $\mathsf{SIG}^{\mathsf{ID}}$ scheme.

Claim. A valid forgery $(m^*, (h^*, s^*))$ output by $\mathcal{A}$ in game $\mathsf{SigForge}_{\mathcal{A},\mathsf{SIG}_2^{\mathsf{ID}}}$ can be transformed into a valid forgery $(m^*, (h^*, s_1^*))$ in game $\mathsf{SigForge}_{\mathcal{S},\mathsf{SIG}^{\mathsf{ID}}}$.

Proof: When $\mathcal{A}$ outputs a valid forgery $(m^*, (h^*, s^*))$, $\mathcal{S}$ extracts the partial signature (h^*, s_1^*) by executing $f_{\mathsf{dec\text{-}sig}}(sk_0, pk_0, (h^*, s^*))$ (from Eq. 5). Note that the simulator knows the adversary's key pair (sk_0, pk_0). The simulator then submits $(m^*, (h^*, s_1^*))$ as its own forgery to the EUF–CMA challenger. By definition, $\mathcal{A}$ wins this game if it has not queried a signature on m^* before. Thus, $\mathcal{S}$ has also not queried the EUF–CMA signing oracle on m^* before. Further, Eq. (5) implies that $(m^*, (h^*, s_1^*))$ is a valid forgery under the public key pk_1. ■

From games $\boldsymbol{G_0} - \boldsymbol{G_4}$, we have that $\Pr[\boldsymbol{G_0} = 1] \leq \Pr[\boldsymbol{G_4} = 1] + \nu(n)$, where $\nu(n) = \nu_1(n) + \nu_2(n) + \nu_3(n) + \nu_4(n)$ is a negligible function in n. Thus, we have $\Pr[\mathsf{SigForge}_{\mathcal{A},\mathsf{SIG}_2^{\mathsf{ID}}}(n) = 1] \leq \Pr[\mathsf{SigForge}_{\mathcal{S},\mathsf{SIG}^{\mathsf{ID}}}(n) = 1] + \nu(n)$.

Adversary corrupts $\mathcal{P}_1$. In case the adversary corrupts $\mathcal{P}_1$, the simulator has to simulate $\mathcal{P}_0$. The proof for this case follows exactly the same steps as above with the exception that game $\boldsymbol{G_1}$ is not required. This is due to the reason that the simulator now plays the role of the committing party in the randomness exchange and hence does not have to extract $\mathcal{A}$'s randomness from the commitment c.

5 Two-party Aggregatable Adaptor Signatures

We are now ready to formally introduce the notion of two-party adaptor signatures with aggregatable public keys which we denote by aSIG_2. Our definition can be seen as a combination of the definition of adaptor signatures from Sec. 3 and the definition of two-party signatures with aggregatable public keys from Sec. 4. Unlike the single party adaptor signatures, in aSIG_2 both parties have the role of the signer and generate pre-signatures cooperatively. Furthermore, both parties can adapt the pre-signature given a witness value y. We note that both the pre-signature and the full-signature are valid under the aggregated public keys of the two parties. We formally define an aSIG_2 scheme w.r.t. a SIG_2 scheme (which is in turn defined w.r.t. a SIG scheme) and a hard relation R.

Afterwards, we show how to instantiate our new definition. Concretely, we present a generic transformation that turns a $\mathsf{SIG}_2^{\mathsf{ID}}$ scheme with certain homomorphic properties into a two-party adaptor signatures scheme. As a $\mathsf{SIG}_2^{\mathsf{ID}}$ scheme is constructed w.r.t. a $\mathsf{SIG}^{\mathsf{ID}}$ scheme (cf. Sec. 4), the construction presented in this section can implicitly transform digital signatures based on ID schemes to two-party adaptor signatures.

The definition of a two-party adaptor signature scheme aSIG_2 is similar to the definition of a standard adaptor signature scheme as for Def. 1. The main difference lies in the pre-signature generation. Namely, the algorithm pSign is replaced by a *protocol* Π_{pSign} which is executed between two parties.

Definition 10 (Two-Party Adaptor Signature Scheme with Aggregatable Public Keys). *A two-party adaptor signature scheme with aggregatable public keys is defined w.r.t. a hard relation* R *and a two-party signature scheme with aggregatable public keys* $\mathsf{SIG}_2 = (\mathsf{Setup}, \mathsf{Gen}, \Pi_{\mathsf{Sign}}, \mathsf{KAg}, \mathsf{Vrfy})$. *It is run between parties* $\mathcal{P}_0, \mathcal{P}_1$ *and consists of a tuple* $\mathsf{aSIG}_2 = (\Pi_{\mathsf{pSign}}, \mathsf{Adapt}, \mathsf{pVrfy}, \mathsf{Ext})$ *of efficient protocols and algorithms which are defined as follows:*

$\Pi_{\mathsf{pSign}\langle sk_{\mathsf{i}}, sk_{1-\mathsf{i}}\rangle}(pk_0, pk_1, m, Y)$**:** *is an interactive protocol that on input secret keys* sk_i *from party* $\mathcal{P}_i$ *with* $i \in \{0,1\}$ *and common values public keys* pk_i*, message* $m \in \{0,1\}^*$ *and statement* $Y \in L_{\mathsf{R}}$*, outputs a pre-signature* $\widetilde{\sigma}$.

$\mathsf{pVrfy}_{apk}(m, Y; \widetilde{\sigma})$**:** *is a DPT algorithm that on input an aggregated public key apk, a message* $m \in \{0,1\}^*$*, a statement* $Y \in L_{\mathsf{R}}$ *and a pre-signature* $\widetilde{\sigma}$*, outputs a bit b.*

$\mathsf{Adapt}_{apk}(\widetilde{\sigma}, y)$**:** *is a DPT algorithm that on input an aggregated public key apk, a pre-signature* $\widetilde{\sigma}$ *and witness y, outputs a signature* σ.

$\mathsf{Ext}_{apk}(\sigma, \widetilde{\sigma}, Y)$**:** *is a DPT algorithm that on input an aggregated public key apk, a signature* σ*, pre-signature* $\widetilde{\sigma}$ *and statement* $Y \in L_{\mathsf{R}}$*, outputs a witness y such that* $(Y, y) \in \mathsf{R}$*, or* $\perp$.

We note that in aSIG_2, the pVrfy algorithm enables public verifiability of the pre-signatures, e.g., aSIG_2 can be used in a three-party protocol where the third party needs to verify the validity of the generated pre-signatrue.

In the following, we formally define properties that a two-party adaptor signature scheme with aggregatable public keys aSIG_2 has to satisfy. These properties are similar to the ones for single party adaptor signature schemes. We start by defining two-party pre-signature correctness which, similarly to Def. 2 states that an honestly generated pre-signature and signature are valid, and it is possible to extract a valid witness from them.

Definition 11 (Two-Party Pre-Signature Correctness). *A two-party adaptor signature with aggregatable public keys* aSIG_2 *satisfies two-party pre-signature correctness, if for all* $n \in \mathbb{N}$*, messages* $m \in \{0,1\}^*$*, it holds that:*

$$\Pr\left[\begin{array}{c|l} \mathsf{pVrfy}_{apk}(m, Y; \widetilde{\sigma}) = 1 & pp \leftarrow \mathsf{Setup}(1^n), (sk_0, pk_0) \leftarrow \mathsf{Gen}(pp) \\ \wedge & (sk_1, pk_1) \leftarrow \mathsf{Gen}(pp), (Y, y) \leftarrow \mathsf{GenR}(1^n) \\ \mathsf{Vrfy}_{apk}(m; \sigma) = 1 & \widetilde{\sigma} \leftarrow \Pi_{\mathsf{pSign}\langle sk_0, sk_1\rangle}(pk_0, pk_1, m, Y) \\ \wedge & apk := \mathsf{KAg}(pk_0, pk_1) \\ (Y, y') \in \mathsf{R} & \sigma := \mathsf{Adapt}_{apk}(\widetilde{\sigma}, y), y' := \mathsf{Ext}_{apk}(\sigma, \widetilde{\sigma}, Y) \end{array}\right] = 1.$$

The unforgeability security definition is similar to Def. 9, except the adversary interacts with two oracles $\mathcal{O}^b_{\Pi_{\mathrm{S}}}, \mathcal{O}^b_{\Pi_{\mathrm{pS}}}$ in order to generate signatures and pre-signatures, as in Def. 3. More precisely, in the $\mathsf{aSigForge}^b_{\mathcal{A},\mathsf{aSIG}_2}(n)$ experiment defined below, $\mathcal{A}$ obtains access to *interactive*, stateful signing and pre-signing oracles $\mathcal{O}^b_{\Pi_{\mathrm{S}}}$ and $\mathcal{O}^b_{\Pi_{\mathrm{pS}}}$ respectively. Oracles $\mathcal{O}^b_{\Pi_{\mathrm{S}}}$ and $\mathcal{O}^b_{\Pi_{\mathrm{pS}}}$ simulate the honest party $\mathcal{P}_{1-b}$ during an execution of the protocols $\Pi^{\mathcal{A}}_{\mathsf{Sign}\langle sk_{1-\mathsf{b}}, \cdot\rangle}$ and $\Pi^{\mathcal{A}}_{\mathsf{pSign}\langle sk_{1-\mathsf{b}}, \cdot\rangle}$ respectively. Similar to Def. 9, both the protocols $\Pi^{\mathcal{A}}_{\mathsf{Sign}\langle sk_{1-\mathsf{b}}, \cdot\rangle}, \Pi^{\mathcal{A}}_{\mathsf{pSign}\langle sk_{1-\mathsf{b}}, \cdot\rangle}$ employed by the respective oracles $\mathcal{O}^b_{\Pi_{\mathrm{S}}}, \mathcal{O}^b_{\Pi_{\mathrm{pS}}}$ gets an oracle access to $\mathcal{A}$ as well.

Definition 12 (2-$\mathsf{aEUF{-}CMA}$ Security). *A two-party adaptor signature with aggregatable public keys* aSIG_2 *is unforgeable if for every PPT adversary* $\mathcal{A}$ *there exists a negligible function* ν *such that:* $\Pr[\mathsf{aSigForge}_{\mathcal{A},\mathsf{aSIG}_2}(n) = 1] \leq \nu(n)$*, where the experiment* $\mathsf{aSigForge}_{\mathcal{A},\mathsf{aSIG}_2}(n)$ *is defined as follows:*

The definition of two-party pre-signature adaptability follows Def. 4 closely. The only difference is that in this setting the pre-signature must be valid under the aggregated public keys.

Definition 13 (Two-Party Pre-Signature Adaptability). *A two-party adaptor signature scheme with aggregatable public keys* aSIG_2 *satisfies two-party pre-signature adaptability, if for all* $n \in \mathbb{N}$*, messages* $m \in \{0,1\}^*$*, statement and witness pairs* $(Y, y) \in \mathsf{R}$*, public keys* pk_0 *and* pk_1*, and pre-signatures* $\widetilde{\sigma} \in \{0,1\}^*$ *satisfying* $\mathsf{pVrfy}_{apk}(m, Y; \widetilde{\sigma}) = 1$ *where* $apk := \mathsf{KAg}(pk_0, pk_1)$*, we have* $\Pr[\mathsf{Vrfy}_{apk}(m; \mathsf{Adapt}_{apk}(\widetilde{\sigma}, y)) = 1] = 1$.

```
$\mathsf{aSigForge}^b_{\mathcal{A},\mathsf{aSIG}_2}(n)$
1 : $\mathcal{Q} := \emptyset, pp \leftarrow \mathsf{Setup}(1^n)$
2 : $(sk_{1-b}, pk_{1-b}) \leftarrow \mathsf{Gen}(pp)$
3 : $(sk_b, pk_b) \leftarrow \mathcal{A}(pp, pk_{1-b})$
4 : $m^* \leftarrow \mathcal{A}^{\mathcal{O}^b_{\Pi_{\mathrm{S}}}, \mathcal{O}^b_{\Pi_{\mathrm{pS}}}}(pk_{1-b}, sk_b, pk_b)$
5 : $(Y, y) \leftarrow \mathsf{GenR}(1^n)$
6 : $\widetilde{\sigma} \leftarrow \Pi^{\mathcal{A}}_{\mathsf{pSign}\langle sk_{1-b}, \cdot \rangle}(m^*, Y)$
7 : $\sigma^* \leftarrow \mathcal{A}^{\mathcal{O}^b_{\Pi_{\mathrm{S}}}, \mathcal{O}^b_{\Pi_{\mathrm{pS}}}}(\widetilde{\sigma}, Y)$
8 : return $\left(m^* \notin \mathcal{Q} \wedge \mathsf{Vrfy}_{\mathsf{KAg}(pk_0, pk_1)}(m^*; \sigma^*)\right)$

$\mathcal{O}^b_{\Pi_{\mathrm{S}}}(m)$
1 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
2 : $\sigma \leftarrow \Pi^{\mathcal{A}}_{\mathsf{Sign}\langle sk_{1-b}, \cdot \rangle}(pk_0, pk_1, m)$

$\mathcal{O}^b_{\Pi_{\mathrm{pS}}}(m, Y)$
1 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
2 : $\widetilde{\sigma} \leftarrow \Pi^{\mathcal{A}}_{\mathsf{pSign}\langle sk_{1-b}, \cdot \rangle}(pk_0, pk_1, m, Y)$
```

Finally, we define two-party witness extractability.

Definition 14 (Two-Party Witness Extractability). *A two-party public key aggregatable adaptor signature scheme* aSIG_2 *is* witness extractable *if for every PPT adversary $\mathcal{A}$, there exists a negligible function ν such that the following holds:* $\Pr[\mathsf{aWitExt}_{\mathcal{A},\mathsf{aSIG}_2}(n) = 1] \leq \nu(n)$*, where the experiment* $\mathsf{aWitExt}_{\mathcal{A},\mathsf{aSIG}_2}$ *is defined as follows:*

```
$\mathsf{aWitExt}^b_{\mathcal{A},\mathsf{aSIG}_2}(n)$
1 : $\mathcal{Q} := \emptyset, pp \leftarrow \mathsf{Setup}(1^n)$
2 : $(sk_{1-b}, pk_{1-b}) \leftarrow \mathsf{Gen}(pp)$
3 : $(sk_b, pk_b) \leftarrow \mathcal{A}(pp, pk_{1-b})$
4 : $(m^*, Y^*) \leftarrow \mathcal{A}^{\mathcal{O}^b_{\Pi_{\mathrm{S}}}, \mathcal{O}^b_{\Pi_{\mathrm{pS}}}}(pk_{1-b}, sk_b, pk_b)$
5 : $\widetilde{\sigma} \leftarrow \Pi^{\mathcal{A}}_{\mathsf{pSign}\langle sk_{1-b}, \cdot \rangle}(m^*, Y^*)$
6 : $\sigma^* \leftarrow \mathcal{A}^{\mathcal{O}^b_{\Pi_{\mathrm{S}}}, \mathcal{O}^b_{\Pi_{\mathrm{pS}}}}(\widetilde{\sigma})$
7 : $apk := \mathsf{KAg}(pk_0, pk_1), y' := \mathsf{Ext}_{apk}(\sigma^*, \widetilde{\sigma}, Y^*)$
8 : return $(m^* \notin \mathcal{Q} \wedge (Y^*, y') \notin \mathsf{R} \wedge \mathsf{Vrfy}_{apk}(m^*; \sigma^*))$

$\mathcal{O}^b_{\Pi_{\mathrm{S}}}(m)$
1 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
2 : $\sigma \leftarrow \Pi^{\mathcal{A}}_{\mathsf{Sign}\langle sk_{1-b}, \cdot \rangle}(pk_0, pk_1, m)$

$\mathcal{O}^b_{\Pi_{\mathrm{pS}}}(m, Y)$
1 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
2 : $\widetilde{\sigma} \leftarrow \Pi^{\mathcal{A}}_{\mathsf{pSign}\langle sk_{1-b}, \cdot \rangle}(pk_0, pk_1, m, Y)$
```

Note that the only difference between this experiment and the $\mathsf{aSigForge}_{\mathcal{A},\mathsf{aSIG}_2}$ experiment is that here the adversary is allowed to choose the statement/witness pair (Y^*, y^*) and that the winning condition additionally requires that for the extracted witness $y' \leftarrow \mathsf{Ext}_{apk}(\sigma^*, \widetilde{\sigma}, Y^*)$ it holds that $(Y^*, y') \notin \mathsf{R}$.

A two-party adaptor signature scheme with aggregatable public keys aSIG_2 is called *secure* if it satisfies the properties 2-aEUF−CMA security, *two-party pre-signature adaptability* and *two-party witness extractability.*

5.1 Generic transformation from $\mathsf{SIG}_2^{\mathsf{ID}}$ to $\mathsf{aSIG}_2^{\mathsf{ID},\mathsf{R}}$

We now present our generic transformation to achieve two-party adaptor signature schemes with aggregatable public keys from identification schemes. In its essence, this transformation is a combination of the transformations presented in Figs. 3 and 6. More precisely, similar to the transformation from $\mathsf{SIG}^{\mathsf{ID}}$ to $\mathsf{aSIG}^{\mathsf{ID},\mathsf{R}}$ presented in Fig. 3, we assume the existence of functions f_{shift}, f_{adapt} and f_{ext} with respect to the relation R. We then make use of the $\Pi_{\mathsf{Rand\text{-}Exc}}$ protocol from the transformation in Fig. 6 to let parties agree on the randomness that is going to be used during the pre-signing process. However, unlike the transformation in

Fig. 6, the resulting randomness is shifted by a statement Y for relation R using the function f_{shift}. The transformation can be found in Fig. 7.

```
$\Pi_{\mathsf{pSign}\langle sk_0, sk_1 \rangle}(pk_0, pk_1, m, Y)$
1 : Parse $pk_i = ((1^n, pp_{\mathsf{C}}, \mathsf{crs}), pk'_i), i \in \{0, 1\}$
2 : $(R_i, St_i, R_{1-i}) \leftarrow \Pi_{\mathsf{Rand\text{-}Exc}\langle sk_i, sk_{1-i} \rangle}(pp_{\mathsf{C}}, \mathsf{crs})$
3 : $R_{\mathsf{pre}} := f_{\mathsf{com\text{-}rand}}(R_0, R_1)$
4 : $R_{\mathsf{sign}} := f_{\mathsf{shift}}(R_{\mathsf{pre}}, Y), h := \mathcal{H}(R_{\mathsf{sign}}, m)$
5 : $\tilde{s}_i \leftarrow \mathsf{P}_2(sk_i, R_i, h, St_i)$
6 : $\tilde{s}_{1-i} \leftarrow \Pi_{\mathsf{Exchange}}\langle \tilde{s}_i, \tilde{s}_{1-i} \rangle$
7 : $(h, \tilde{s}) := f_{\mathsf{com\text{-}sig}}(h, (\tilde{s}_i, \tilde{s}_{1-i}))$
8 : return $(h, \tilde{s})$
```

```
$\mathsf{pVrfy}_{apk}(m, Y; (h, \tilde{s}))$
1 : $\widehat{R}_{\mathsf{pre}} := \mathsf{V}_0(apk, h, \tilde{s})$
2 : return $h = \mathcal{H}(f_{\mathsf{shift}}(\widehat{R}_{\mathsf{pre}}, Y), m)$

$\mathsf{Adapt}_{pk}((h, \tilde{s}), y)$
1 : return $(h, f_{\mathsf{adapt}}(\tilde{s}, y))$

$\mathsf{Ext}_{pk}((h, s), (h, \tilde{s}), Y)$
1 : return $f_{\mathsf{ext}}(s, \tilde{s})$
```

Fig. 7: A two-party adaptor signature scheme with aggregatable public keys $\mathsf{aSIG}_2^{\mathsf{ID},\mathsf{R}}$ defined with respect to a $\mathsf{SIG}_2^{\mathsf{ID}}$ scheme and a hard relation R.

Theorem 4. *Assume that $\mathsf{SIG}^{\mathsf{ID}}$ is an $\mathsf{SUF{-}CMA}$-secure signature scheme transformed using Fig. 2. Let $f_{\mathsf{shift}}, f_{\mathsf{adapt}}$ and f_{ext} be functions satisfying the relations from Equations (1) and (2), and R be a hard relation. Further, assume that $f_{\mathsf{com\text{-}sig}}$, $f_{\mathsf{com\text{-}pk}}$ and $f_{\mathsf{dec\text{-}sig}}$ satisfy the relation from Equations (4) and (5). Then the resulting $\mathsf{aSIG}_2^{\mathsf{ID},\mathsf{R}}$ scheme from the transformation in Fig. 7 is a secure two-party adaptor signature scheme with aggregatable public keys in the random oracle model.*

In order to prove Thm. 4, we must show that $\mathsf{aSIG}_2^{\mathsf{ID},\mathsf{R}}$ satisfies the *pre-signature correctness*, 2-$\mathsf{aEUF{-}CMA}$ *security*, *pre-signature adaptability* and *witness extractability* properties as described in Defs. 11 to 14 respectively. We provide the full proofs of the following lemmas in Appx. C and only mention the intuition behind the proofs here. As mentioned in the introduction of this work, despite the fact that $\mathsf{aSIG}_2^{\mathsf{ID},\mathsf{R}}$ is constructed from $\mathsf{SIG}_2^{\mathsf{ID}}$, we require only $\mathsf{SIG}^{\mathsf{ID}}$ to be $\mathsf{SUF{-}CMA}$-secure in order to prove 2-$\mathsf{aEUF{-}CMA}$ security for $\mathsf{aSIG}_2^{\mathsf{ID},\mathsf{R}}$.

Lemma 7 (Two-Party Pre-Signature Correctness). *Under the assumptions of Thm. 4, $\mathsf{aSIG}_2^{\mathsf{ID},\mathsf{R}}$ satisfies Def. 11.*

The proof of Lemma 7 follows directly from Equations (1) to (3) and the correctness of SIG_2 from Lemma 5.

Lemma 8 (2-$\mathsf{aEUF{-}CMA}$-Security). *Under the assumptions of Thm. 4, $\mathsf{aSIG}_2^{\mathsf{ID},\mathsf{R}}$ satisfies Def. 12.*

Proof Sketch: In a nutshell the proof of this lemma is a combination of the proofs of Lemmas 2 and 6, i.e., the proof is done by a reduction to the hardness of the relation R and the $\mathsf{SUF{-}CMA}$ of the underlying signature scheme. During the signing process, the challenger queries its $\mathsf{SUF{-}CMA}$ signing oracle and receives a signature σ. As in the proof of Lemma 6, the challenger programs the random oracle such that σ appears like a signature generated with the combined randomness of the challenger and the adversary. Simulating the pre-signing process is similar with the exception that before programming the random oracle, the randomness must be shifted using the function f_{shift}. Finally, the challenger and the adversary generate a pre-signature $\widetilde{\sigma}^* = (h, \tilde{s})$ on the challenge message m^* and the adversary outputs the forgery $\sigma^* = (h, s)$. If $f_{\mathsf{ext}}(s, \tilde{s})$ returns the y generated by the challenger, as in the proof of Lemma 2, the hardness of the relation R can be broken. Otherwise, using $f_{\mathsf{dec\text{-}sig}}$, it is possible to use the forgery provided by the adversary to extract a forgery for the $\mathsf{SUF{-}CMA}$ game.

Lemma 9 (Two-Party Pre-Signature Adaptability). *Under the assumptions of Thm. 4, $\mathsf{aSIG}_2^{\mathsf{ID},\mathsf{R}}$ satisfies Def. 13.*

Proof Sketch: The proof of Lemma 9 is analogous to the proof of Lemma 3.

Lemma 10 (Two-party Witness Extractability). *Under the assumptions of Thm. 4,* $\mathsf{aSIG}_2^{\mathsf{ID},\mathsf{R}}$ *satisfies Def. 14.*

Proof Sketch: The proof of Lemma 10 is very similar to the proof of Lemma 8 except that the adversary chooses Y now and thus, no reduction to the hardness of the relation R is needed.

Acknowledgments

This work was partly supported by the German Research Foundation (DFG) Emmy Noether Program *FA 1320/1-1*, by the *DFG CRC 1119 CROSSING* (project S7), by the German Federal Ministry of Education and Research (BMBF) *iBlockchain project* (grant nr. 16KIS0902), by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts within their joint support of the *National Research Center for Applied Cybersecurity ATHENE* and by *NSCX project* (project number CS1920241NSCX008801) on *Design and Development of Blockchain based Technologies* in the *Department of Computer Science and Engineering, IIT Madras.*

References

[1] M. Abdalla et al. "Tighter Reductions for Forward-Secure Signature Schemes". In: *PKC 2013*. 2013.

[2] L. Aumayr et al. *Generalized Bitcoin-Compatible Channels*. Cryptology ePrint Archive, Report 2020/476. https://tinyurl.com/y52ggoj6. 2020.

[3] M. Bellare and G. Neven. "Multi-signatures in the plain public-Key model and a general forking lemma". In: *ACM CCS 2006*. 2006.

[4] M. Bellare and S. Shoup. "Two-Tier Signatures, Strongly Unforgeable Signatures, and Fiat-Shamir Without Random Oracles". In: *PKC 2007*. 2007.

[5] *Bitcoin Scripts*. https://en.bitcoin.it/wiki/Script#Crypto.

[6] *Bitcoin Wiki: Payment Channels*. https://tinyurl.com/y6msnk7u.

[7] M. Blum et al. "Non-Interactive Zero-Knowledge and Its Applications (Extended Abstract)". In: *20th ACM STOC*. 1988.

[8] A. Boldyreva. "Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme". In: *PKC 2003*. 2003.

[9] D. Boneh et al. "Compact Multi-signatures for Smaller Blockchains". In: *ASIACRYPT 2018, Part II*. 2018.

[10] D. Boneh et al. "Short Signatures from the Weil Pairing". In: *ASIACRYPT 2001*. 2001.

[11] D. Chaum and T. P. Pedersen. "Wallet Databases with Observers". In: *CRYPTO'92*. 1993.

[12] A. De Santis et al. "Necessary and Sufficient Assumptions for Non-iterative Zero-Knowledge Proofs of Knowledge for All NP Relations". In: *ICALP 2000*. 2000.

[13] C. Decker and R. Wattenhofer. "A Fast and Scalable Payment Network with Bitcoin Duplex Micropayment Channels". In: *Stabilization, Safety, and Security of Distributed Systems 2015*. 2015.

[14] A. Deshpande and M. Herlihy. "Privacy-Preserving Cross-Chain Atomic Swaps". In: *FC 2020*. 2020.

[15] M. Drijvers et al. "On the Security of Two-Round Multi-Signatures". In: *2019 IEEE Symposium on Security and Privacy*. 2019.

[16] L. Eckey et al. *Splitting Payments Locally While Routing Interdimensionally*. Cryptology ePrint Archive, Report 2020/555. https://eprint.iacr.org/2020/555. 2020.

[17] M. F. Esgin et al. "Post-Quantum Adaptor Signatures and Payment Channel Networks". In: *ESORICS 2020*. 2020.

[18] M. Fischlin. "Communication-Efficient Non-interactive Proofs of Knowledge with Online Extractors". In: *CRYPTO 2005*. 2005.

[19] L. Fournier. *One-Time Verifiably Encrypted Signatures A.K.A. Adaptor Signatures*. `https://tinyurl.com/y4qxopxp`. 2019.

[20] R. Gennaro et al. "Threshold-Optimal DSA/ECDSA Signatures and an Application to Bitcoin Wallet Security". In: *ACNS 16*. 2016.

[21] S. Goldwasser and R. Ostrovsky. "Invariant signatures and non-interactive zero-knowledge proofs are equivalent". In: *Annual International Cryptology Conference*. Springer. 1992.

[22] J. Groth et al. "Perfect Non-interactive Zero Knowledge for NP". In: *EUROCRYPT 2006*. 2006.

[23] J. Gugger. *Bitcoin–Monero Cross-chain Atomic Swap*. Cryptology ePrint Archive, Report 2020/1126. `https://eprint.iacr.org/2020/1126`. 2020.

[24] L. C. Guillou and J.-J. Quisquater. "A "Paradoxical" Indentity-Based Signature Scheme Resulting from Zero-Knowledge". In: *CRYPTO'88*. 1990.

[25] T. Hardjono and Y. Zheng. "A practical digital multisignature scheme based on discrete logarithms (extended abstract)". In: *Advances in Cryptology — AUSCRYPT '92*. 1993.

[26] J. Katz and N. Wang. "Efficiency Improvements for Signature Schemes with Tight Security Reductions". In: *ACM CCS 2003*. 2003.

[27] E. Kiltz et al. "Optimal Security Proofs for Signatures from Identification Schemes". In: *CRYPTO 2016, Part II*. 2016.

[28] D.-P. Le et al. *DDH-based Multisignatures with Public Key Aggregation*. Cryptology ePrint Archive, Report 2019/771. `https://eprint.iacr.org/2019/771`. 2019.

[29] Y. Lindell. "Fast Secure Two-Party ECDSA Signing". In: *CRYPTO 2017, Part II*. 2017.

[30] S. Lu et al. "Sequential Aggregate Signatures and Multisignatures Without Random Oracles". In: *EUROCRYPT 2006*. 2006.

[31] A. Lysyanskaya. "Unique signatures and verifiable random functions from the DH-DDH separation". In: *Annual International Cryptology Conference*. Springer. 2002.

[32] G. Malavolta et al. "Anonymous Multi-Hop Locks for Blockchain Scalability and Interoperability". In: *NDSS 2019*. 2019.

[33] G. Maxwell et al. "Simple Schnorr multi-signatures with applications to Bitcoin". In: *Designs, Codes and Cryptography 2019* (2019).

[34] S. Micali et al. "Verifiable Random Functions". In: *40th FOCS*. 1999.

[35] A. Miller et al. "Sprites and State Channels: Payment Networks that Go Faster Than Lightning". In: *FC 2019*. 2019.

[36] S. Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. `http://bitcoin.org/bitcoin.pdf`. 2009.

[37] K. Ohta and T. Okamoto. "A digital multisignature scheme based on the Fiat-Shamir scheme". In: *Advances in Cryptology — ASIACRYPT '91*. 1993.

[38] T. Okamoto. "A Digital Multisignature Scheme Using Bijective Public-Key Cryptosystems". In: *ACM Trans. Comput. Syst.* 4 (1988).

[39] A. Poelstra. *Scriptless scripts*. `https://tinyurl.com/ludcxyz`. 2017.

[40] J. Poon and T. Dryja. *The Bitcoin Lightning Network: Scalable Off-chain Instant Payments*. `https://tinyurl.com/q54gnb4`. 2016.

[41] R. L. Rivest et al. "How to Leak a Secret". In: *ASIACRYPT 2001*. 2001.

[42] N. van Saberhagen. *CryptoNote v 2.0*. `https://tinyurl.com/lmtylgo`.

[43] C.-P. Schnorr. "Efficient Signature Generation by Smart Cards". In: *Journal of Cryptology* 3 (1991).

[44] S.-T. Shen et al. "Unique signature with short output from cdh assumption". In: *International Conference on Provable Security*. Springer. 2015.

[45] E. Tairi et al. *A^2L: Anonymous Atomic Locks for Scalability in Payment Channel Hubs*. Cryptology ePrint Archive, Report 2019/589. `https://eprint.iacr.org/2019/589`. 2019.

[46] E. Tairi et al. *Post-Quantum Adaptor Signature for Privacy-Preserving Off-Chain Payments*. Cryptology ePrint Archive, Report 2020/1345. `https://eprint.iacr.org/2020/1345`. 2020.

Supplementary Material

A Additional material: Preliminaries

Definition 15 (Digital signatures). *A digital signature scheme* SIG *is a triple of algorithms* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *defined as:*

$\mathsf{Gen}(1^n)$**:** *is a PPT algorithm that on input a security parameter n, outputs a key pair (sk, pk);*
$\mathsf{Sign}_{sk}(m)$**:** *is a PPT algorithm that on input a secret key sk and message $m \in \{0,1\}^*$, outputs a signature σ;*
$\mathsf{Vrfy}_{pk}(m;\sigma)$**:** *is a DPT algorithm that on input a public key pk, message $m \in \{0,1\}^*$ and signature σ, outputs a bit b.*

A signature scheme must satisfy that for all messages $m \in \{0,1\}^*$ it holds that:

$$\forall m \in \{0,1\}^*, \quad \Pr\left[\mathsf{Vrfy}_{pk}(m; \mathsf{Sign}_{sk}(m)) = 1 \mid (sk, pk) \leftarrow \mathsf{Gen}(1^n)\right] = 1,$$

where the probability is taken over the randomness of Gen *and* Sign*.*

Definition 16 (SUF–CMA security). *A signature scheme* SIG *is* SUF–CMA *secure if for every PPT adversary $\mathcal{A}$ there exists a negligible function ν such that* $\Pr[\mathsf{strongSigForge}_{\mathcal{A},\mathsf{SIG}}(n) = 1] \leq \nu(n)$, *where the experiment* $\mathsf{strongSigForge}_{\mathcal{A},\mathsf{SIG}}$ *is defined as follows:*

```
$\mathsf{strongSigForge}_{\mathcal{A},\mathsf{SIG}}(n)$
$\mathcal{Q} := \emptyset$
$(sk, pk) \leftarrow \mathsf{Gen}(1^n)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_S}(pk)$
return $((m^*, \sigma^*) \notin \mathcal{Q} \wedge \mathsf{Vrfy}_{pk}(m^*; \sigma^*))$

$\mathcal{O}_S(m)$
$\sigma \leftarrow \mathsf{Sign}_{sk}(m)$
$\mathcal{Q} := \mathcal{Q} \cup \{m, \sigma\}$
return $\sigma$
```

Definition 17 (EUF–CMA security). *A signature scheme* SIG *is* EUF–CMA *secure if for every PPT adversary $\mathcal{A}$ there exists a negligible function ν such that* $\Pr[\mathsf{SigForge}_{\mathcal{A},\mathsf{SIG}}(n) = 1] \leq \nu(n)$, *where the experiment* $\mathsf{SigForge}_{\mathcal{A},\mathsf{SIG}}$ *is defined as follows:*

```
$\mathsf{SigForge}_{\mathcal{A},\mathsf{SIG}}(n)$
$\mathcal{Q} := \emptyset$
$(sk, pk) \leftarrow \mathsf{Gen}(1^n)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_S}(pk)$
return $(m^* \notin \mathcal{Q} \wedge \mathsf{Vrfy}_{pk}(m^*; \sigma^*))$

$\mathcal{O}_S(m)$
$\sigma \leftarrow \mathsf{Sign}_{sk}(m)$
$\mathcal{Q} := \mathcal{Q} \cup \{m\}$
return $\sigma$
```

Non-interactive zero knowledge proof. We now recall the definition of a non-interactive zero-knowledge (NIZK) proof of knowledge which has first been introduced in [7]. A NIZK proof of knowledge with respect to a polynomial-time recognizable binary relation R is given by the following tuple of PPT algorithms $\mathsf{NIZK} := (\mathsf{Setup}_\mathsf{R}, \mathsf{Prove}, \mathsf{Verify})$, where (i) $\mathsf{Setup}_\mathsf{R}(1^n)$ outputs a common reference string crs; (ii) $\mathsf{Prove}(\mathsf{crs}, (Y, y))$ outputs a proof π for $(Y, y) \in \mathsf{R}$; (iii) $\mathsf{Verify}(\mathsf{crs}, Y, \pi)$ outputs a bit $b \in \{0,1\}$. Further, the NIZK proof of knowledge w.r.t. R should satisfy the following properties: (i) *Completeness*: For all $(Y, y) \in \mathsf{R}$ and $\mathsf{crs} \leftarrow \mathsf{Setup}_\mathsf{R}(1^n)$, it holds that $\mathsf{Verify}(\mathsf{crs}, Y, \mathsf{Prove}(\mathsf{crs}, (Y, y))) = 1$ except with negligible probability; (ii) *Soundness*: For any $(Y, y) \notin \mathsf{R}$ and $\mathsf{crs} \leftarrow \mathsf{Setup}_\mathsf{R}(1^n)$, it holds that $\mathsf{Verify}(\mathsf{crs}, Y, \mathsf{Prove}(\mathsf{crs}, (Y, y))) = 0$ except with negligible probability; (iii) *Zero knowledge*: For any PPT adversary $\mathcal{A}$, there exist PPT algorithms $\mathsf{Setup}'_\mathsf{R}$ and S, where $\mathsf{Setup}'_\mathsf{R}(1^n)$ on input the security parameter, outputs a pair $(\widetilde{\mathsf{crs}}, \tau)$ with τ being a trapdoor and $\mathsf{S}(\widetilde{\mathsf{crs}}, \tau, Y)$ which on input $\widetilde{\mathsf{crs}}, \tau$ and a statement Y, outputs a simulated proof π_S for any $(Y, y) \in R$. It must hold that (1) the distributions $\{\mathsf{crs} : \mathsf{crs} \leftarrow \mathsf{Setup}_\mathsf{R}(1^n)\}$ and $\{\widetilde{\mathsf{crs}} : (\widetilde{\mathsf{crs}}, \tau) \leftarrow \mathsf{Setup}'_\mathsf{R}(1^n)\}$ are indistinguishable to $\mathcal{A}$ except with negligible probability; (2) for any $(\widetilde{\mathsf{crs}}, \tau) \leftarrow \mathsf{Setup}'_\mathsf{R}(1^n)$ and any $(Y, y) \in \mathsf{R}$, the distributions $\{\pi : \pi \leftarrow \mathsf{Prove}(\widetilde{\mathsf{crs}}, Y, y)\}$ and $\{\pi_\mathsf{S} : \pi_\mathsf{S} \leftarrow \mathsf{S}(\widetilde{\mathsf{crs}}, \tau, Y)\}$ are indistinguishable to $\mathcal{A}$ except with negligible probability.

Extractable commitments. Extractable commitment schemes have been first introduced in [12]. A commitment scheme consists of a tuple of three PPT algorithms, $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Dec})$ where Gen gets as input the security parameter n and outputs public parameters pp, Com takes as input pp and a message $m \in \{0,1\}^*$ and outputs a tuple (c, d) and Dec takes as input pp and a tuple (c, d) and either outputs m or $\perp$. Let n be the security parameter and let $pp \leftarrow \mathsf{Gen}(1^n)$. A commitment scheme is computationally hiding if for any two messages m, m' and $(c, d) \leftarrow \mathsf{Com}(pp, m)$ and $(c', d') \leftarrow \mathsf{Com}(pp, m')$, there exists no PPT adversary $\mathcal{A}$ who can distinguish the tuples (m, m', c) and (m, m', c') except with negligible probability. A commitment scheme is computationally binding if for any two messages $m \neq m'$ and $(c, d) \leftarrow \mathsf{Com}(pp, m)$, there exists no PPT adversary $\mathcal{A}$ who can generate an opening d' such that $m' \leftarrow \mathsf{Dec}(pp, c, d')$ except with negligible probability. Finally, a commitment scheme is extractable if Gen outputs an additional secret trapdoor, denoted by tr, and there exists an efficient PPT algorithm $\mathsf{Extract}$ such that for any message m and any tuple $(c, d) \leftarrow \mathsf{Com}(pp, m)$ it holds that $\mathsf{Extract}(pp, tr, c) = m$ except with negligible probability.

B Instantiations

In this section, we show how to instantiate the functions f_{shift}, f_{adapt}, f_{ext} used in our transformations in order to transform Schnorr [43], Katz-Wang[26, 11] and Guillou-Quisquater[24] signature scheme. We note all these signature schemes are obtained from canonical identification schemes and are strongly unforgeable [27]. For simplicity, we ignore the fact that the public parameter pp, is appended to the public keys.

B.1 Schnorr Instantiation

First we recall how Schnorr signature scheme is constructed. Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of prime order q where the discrete logarithm problem in $\mathbb{G}$ is hard. The functions IGen, P_1, P_2 and V_0 for Schnorr's signature scheme are defined in Fig. 8.

```
IGen(n)
1 : sk ←$ Z_q
2 : pk = g^sk
3 : return (sk, pk)
```

```
P_1(sk)
1 : r ←$ Z_q
2 : R = g^r
3 : return (R, r)
```

```
P_2(sk, R, h, r)
1 : s = r + h · sk
2 : return s
```

```
V_0(pk, h, s)
1 : R = g^s · pk^{-h}
2 : return (R)
```

Fig. 8: Schnorr signature scheme

Let us consider the hard relation $\mathsf{R} = \{(Y, y) \mid Y = g^y\}$, i.e., group elements and their discrete logarithms, and let define the functions f_{shift}, f_{adapt}, f_{ext} as:

$$f_{\mathsf{shift}}(Y, R) := Y \cdot R, \quad f_{\mathsf{adapt}}(\tilde{s}, y) := \tilde{s} + y, \quad f_{\mathsf{ext}}(s, \tilde{s}) := s - \tilde{s}.$$

Intuitively, the function f_{shift} is *shifting* randomness in the group while the function f_{adapt} is *shifts* randomness in the exponent. To prove that Eq. (1) holds, let us fix an arbitrary public key $pk \in \mathbb{G}$, a challenge $h \in \mathbb{Z}_q$, a response value $s \in \mathbb{Z}_q$ and a statement witness pair $(Y, y) \in \mathsf{R}$, i.e, $Y = g^y$. We have:

$$\begin{aligned} f_{\mathsf{shift}}(\mathsf{V}_0(pk, h, s), Y) &= f_{\mathsf{shift}}(g^s \cdot pk^{-h}, Y) = g^s \cdot pk^{-h} \cdot Y \\ &= g^{s+y} \cdot pk^{-h} = \mathsf{V}_0(pk, h, s + y) = \mathsf{V}_0(pk, h, f_{\mathsf{adapt}}(s, y)) \end{aligned}$$

which is what we wanted to prove. In order to show that Eq. (2) holds, let us fix an arbitrary witness $y \in \mathbb{Z}_q$ and a response value $s \in \mathbb{Z}_q$. Then we have

$$f_{\mathsf{ext}}(f_{\mathsf{adapt}}(s, y), s) = f_{\mathsf{ext}}(s + y, s) = s + y - s = y$$

and hence Eq. (2) is satisfied as well.

We now can give the description of the functions $f_{\mathsf{com\text{-}pk}}$, $f_{\mathsf{com\text{-}rand}}$, $f_{\mathsf{com\text{-}sig}}$ and $f_{\mathsf{dec\text{-}sig}}$:

$$f_{\mathsf{com\text{-}pk}}(pk_0, pk_1) \coloneqq pk_0 \cdot pk_1, \qquad f_{\mathsf{com\text{-}rand}}(R_0, R_1) \coloneqq R_0 \cdot R_1$$
$$f_{\mathsf{com\text{-}sig}}(h, (s_0, s_1)) \coloneqq (h, (s_0 + s_1)), f_{\mathsf{dec\text{-}sig}}(sk_i, pk_i, (h, s)) \coloneqq (h, s - sk_i \cdot h)$$

Let us now show that the Eq. 4 and 5 holds. It is easy to see that a combined signature is valid under the combined public key. Let us fix two arbitrary secret key $sk_0, sk_1 \in \mathbb{Z}_q$, a challenge $h \in \mathbb{Z}_q$, two random values $r_0, r_1 \in \mathbb{Z}_q$ and a message $m \in \{0,1\}^*$. We have:

$$\begin{aligned} s \coloneqq& s_0 + s_1 = (sk_0 \cdot h + r_0) + (sk_1 \cdot h + r_1) = (sk_0 + sk_1) \cdot h + (r_0 + r_1) \\ apk \coloneqq& pk_0 \cdot pk_1 = g^{sk_0 + sk_1} \end{aligned}$$

As such according to the definition of V_0 we have:

$$\mathsf{V}_0(apk, h, s) = g^s \cdot g^{-(sk_0 + sk_1) \cdot h} = g^{r_0 + r_1}$$

Therefore, the verification algorithm will return 1 as $h = \mathcal{H}(g^{r_0 + r_1}, m)$. Hence, Eq. 4 holds.

Now let us see why $f_{\mathsf{dec\text{-}sig}}$ returns a valid signature under the public key pk_{1-i}. According to $f_{\mathsf{dec\text{-}sig}}$ definition given above we have:

$$s_{1-i} = s - sk_i \cdot h = (sk_i + sk_{1-i}) \cdot h + (r_i + r_{1-i}) - sk_i \cdot h = (sk_{i-i}) \cdot h + (r_i + r_{1-i})$$

As $h = \mathcal{H}(g^{r_i + r_{1-i}}, m)$, we can conclude that the tuple (h, s_{1-i}) is a valid signature under the public key pk_{1-i}.

B.2 Katz-Wang Instantiation

We now recall how the Katz-Wang signature scheme is constructed. In a nutshell, this construction is very similar to Schnorr except it uses two generators g_1 and g_2. Let $\mathbb{G} = \langle g_1 \rangle = \langle g_2 \rangle$ be a cyclic group of prime order q where the discrete logarithm problem in $\mathbb{G}$ is hard. The functions IGen, P_1, P_2 and V_0 for Katz-Wang's signature scheme are defined in Fig. 9.

$\mathsf{IGen}(n)$	$\mathsf{V}_0(pk, h, s)$	$\mathsf{P}_1(sk)$	$\mathsf{P}_2(sk, R, h, r)$
$1: sk \leftarrow_\$ \mathbb{Z}_q$	$1: R_1 = g_1^s \cdot pk^{-h}$	$1: r \leftarrow_\$ \mathbb{Z}_q$	$1: s = r + h \cdot sk$
$2: pk = (g_1^{sk}, g_2^{sk})$	$2: R_2 = g_2^s \cdot pk^{-h}$	$2: R_1 = g_1^r$	$2:$ **return** s
$3:$ **return** (sk, pk)	$3:$ **return** (R_1, R_2)	$3: R_2 = g_2^r$	
		$4:$ **return** $((R_1, R_2), r)$	

Fig. 9: Katz-Wang signature scheme

For this construction, a tuple $((Y_1, Y_2), y)$ is in the relation R if $Y_1 = g_1^y$ and $Y_2 = g_2^y$ i.e., $\mathsf{R} = \{((Y_1, Y_2), y) \mid Y_1 = g_1^y \land Y_2 = g_2^y\}$. The functions f_{shift}, f_{adapt}, f_{ext} are defined as:

$$f_{\mathsf{shift}}((Y_1, Y_2), (R_1, R_2)) \coloneqq ((Y_1 \cdot R_1), (Y_2 \cdot R_2)),$$
$$f_{\mathsf{adapt}}(\tilde{s}, y) \coloneqq \tilde{s} + y, f_{\mathsf{ext}}(s, \tilde{s}) \coloneqq s - \tilde{s}$$

To prove that Eq. (1) holds, let us fix an arbitrary public key $pk = (pk', pk'') \in \mathbb{G} \times \mathbb{G}$, a challenge $h \in \mathbb{Z}_q$, a response value $s \in \mathbb{Z}_q$ and a statement witness pair $(Y_1, Y_2, y) \in \mathsf{R}$. We have:

$$\begin{aligned} f_{\mathsf{shift}}((Y_1, Y_2), \mathsf{V}_0(pk, h, \tilde{s})) &= f_{\mathsf{shift}}((Y_1, Y_2), ((g_1^{\tilde{s}} \cdot pk'^{-h}), (g_2^{\tilde{s}} \cdot pk''^{-h}))) \\ &= (Y_1 \cdot (g_1^{\tilde{s}} \cdot pk'^{-h}), Y_2 \cdot (g_2^{\tilde{s}} \cdot pk''^{-h})) \\ &= ((g_1^{\tilde{s}+y} \cdot pk'^{-h}), (g_2^{\tilde{s}+y} \cdot pk''^{-h})) = \mathsf{V}_0(pk, h, f_{\mathsf{adapt}}(\tilde{s}, y)) \end{aligned}$$

In order to show that Eq. (2) holds, let us fix an arbitrary witness $y \in \mathbb{Z}_q$ and a response value $s \in \mathbb{Z}_q$. Then we have

$$f_{\mathsf{ext}}(f_{\mathsf{adapt}}(s, y), s) = f_{\mathsf{ext}}(s + y, s) = s + y - s = y$$

and hence Eq. (2) is satisfied as well.

We now can give the description of the functions $f_{\mathsf{com\text{-}pk}}$, $f_{\mathsf{com\text{-}rand}}$, $f_{\mathsf{com\text{-}sig}}$ and $f_{\mathsf{dec\text{-}sig}}$:

$$\begin{aligned} f_{\mathsf{com\text{-}pk}}(pk_0, pk_1) &:= pk_0 \cdot pk_1 \\ f_{\mathsf{com\text{-}rand}}((R_{0,1}, R_{0,2}), (R_{1,1}, R_{1,2})) &:= ((R_{0,1} \cdot R_{1,1}), (R_{0,2} \cdot R_{1,2})) \\ f_{\mathsf{com\text{-}sig}}(h, (s_0, s_1)) &:= (h, (s_0 + s_1)) \\ f_{\mathsf{dec\text{-}sig}}(sk_i, pk_i, (h, s)) &:= (h, s - sk_i \cdot h) \end{aligned}$$

Let us now show that the Eq. 4 and 5 holds. It is easy to see that a combined signature is valid under the combined public key. Let us fix two arbitrary secret keys $sk_0, sk_1 \in \mathbb{Z}_q$, a challenge $h \in \mathbb{Z}_q$, two random values $r_0, r_1 \in \mathbb{Z}_q$ and a message $m \in \{0,1\}^*$. We have:

$$\begin{aligned} s &:= s_0 + s_1 = (sk_0 \cdot h + r_0) + (sk_1 \cdot h + r_1) = (sk_0 + sk_1) \cdot h + (r_0 + r_1) \\ apk &:= pk_0 \cdot pk_1 = (g_1^{sk_0 + sk_1}, g_2^{sk_0 + sk_1}) \end{aligned}$$

As such according to the definition of V_0 we have:

$$\mathsf{V}_0(apk, h, s) = ((g_1^s \cdot g_1^{-(sk_0+sk_1)\cdot h}), (g_2^s \cdot g_2^{-(sk_0+sk_1)\cdot h})) = ((g_1^{r_0+r_1}), (g_2^{r_0+r_1}))$$

Therefore, the verification algorithm will return 1 as $h = \mathcal{H}(g_1^{r_0+r_1}, g_2^{r_0+r_1}, m)$. Hence, Eq. 4 holds.

Now let us see why $f_{\mathsf{dec\text{-}sig}}$ returns a valid signature under the public key pk_{1-i}. According to the definition of $f_{\mathsf{dec\text{-}sig}}$ given above, we have:

$$s_{1-i} := s - sk_i \cdot h = (sk_i + sk_{1-i}) \cdot h + (r_i + r_{1-i}) - sk_i \cdot h = (sk_{i-i}) \cdot h + (r_i + r_{1-i})$$

As $h = \mathcal{H}(g_1^{r_i + r_{1-i}}, g_2^{r_i + r_{1-i}}, m)$, we can conclude that the tuple (h, s_{1-i}) is a valid signature under the public key pk_{1-i}.

B.3 Guillou-Quisquater Instantiation

Unlike the previous two constructions Guillou-Quisquater signature scheme is an RSA based scheme. Let us first recall the RSA assumption. Let p and q be two $n/2$-bit prime numbers and $N = p \cdot q$. Furthermore, let e be a $n \cdot c$-bit long prime number where $0 < c < \frac{1}{4}$ such that the greatest common divider of e and $\phi(N) = (p-1) \cdot (q-1)$ is 1. The functions IGen, P_1, P_2 and V_0 for Guillou-Quisquater's signature scheme are defined in Fig. 10.

```
$\mathsf{IGen}(n)$
1 : $sk \leftarrow_\$ \mathbb{Z}_N^*$
2 : $pk = (e^{sk})$
3 : return $(sk, pk)$

$\mathsf{P}_1(sk)$
1 : $r \leftarrow_\$ \mathbb{Z}_N^*$
2 : $R = r^e \mod N$
3 : return $(R, r)$

$\mathsf{P}_2(sk, R, h, r)$
1 : $s = sk^h \cdot r \mod N$
2 : return $s$

$\mathsf{V}_0(pk, h, s)$
1 : $R = s^e \cdot pk^{-h} \mod N$
2 : if $(R, s) \in \mathbb{Z}_N^* \times \mathbb{Z}_N^*$
3 :     return $R$
4 : else return $\bot$
```

Fig. 10: Guillou-Quisquater signature scheme

For this construction, a tuple (Y, y) is in the relation if $Y = y^e$ i.e., $\mathsf{R} = \{(Y, y) \mid y^e \mod N\}$. The functions f_{shift}, f_{adapt}, f_{ext} are instantiated as follows:

$$f_{\mathsf{shift}}(Y, R) := Y \cdot R,\ \ f_{\mathsf{adapt}}(\tilde{s}, y) := \tilde{s} \cdot y,\ \ f_{\mathsf{ext}}(s, \tilde{s}) := s/\tilde{s}$$

To prove that Eq. (1) holds, let us fix an arbitrary public key $pk \in \mathbb{Z}_N^*$, a challenge $h \in \mathbb{Z}_e$, a response value s and a statement witness pair $(Y, y) \in \mathsf{R}$, i.e, $Y = y^e \mod N$. We have:

$$\begin{aligned} f_{\mathsf{shift}}(\mathsf{V}_0(pk, h, \tilde{s}), Y) =& f_{\mathsf{shift}}(s^e \cdot pk^{-h}, Y) = s^e \cdot pk^{-h} \cdot y^e = (s + y)^e \cdot pk^{-h} \\ =& \mathsf{V}_0(pk, h, f_{\mathsf{adapt}}(\tilde{s}, y)) \end{aligned}$$

Equation Eq. (2) is satisfied since:

$$f_{\mathsf{ext}}(f_{\mathsf{adapt}}(s, y), s) = f_{\mathsf{ext}}(s \cdot y, s) = s \cdot y/s = y$$

We now can give the description of the functions $f_{\mathsf{com\text{-}pk}}$, $f_{\mathsf{com\text{-}rand}}$, $f_{\mathsf{com\text{-}sig}}$ and $f_{\mathsf{dec\text{-}sig}}$:

$$\begin{aligned} f_{\mathsf{com\text{-}pk}}(pk_0, pk_1) := pk_0 \cdot pk_1, \qquad & f_{\mathsf{com\text{-}rand}}(R_0, R_1) := (R_0 \cdot R_1), \\ f_{\mathsf{com\text{-}sig}}(h, (s_0, s_1)) := (h, (s_0 \cdot s_1)),\ & f_{\mathsf{dec\text{-}sig}}(sk_i, pk_i, (h, s)) := (h, s \cdot sk_i^{-h}) \end{aligned}$$

Let us now show that the Eq. 4 and 5 holds. It is easy to see that a combined signature is valid under the combined public key. Let us fix two arbitrary secret key $sk_0, sk_1 \in \mathbb{Z}_N^*$, a challenge $h \in \mathbb{Z}_e$, two random values $r_0, r_1 \in \mathbb{Z}_N^*$ and a message $m \in \{0, 1\}^*$. We have:

$$\begin{aligned} s :=& s_0 \cdot s_1 = (sk_0 \cdot sk_1)^h \cdot (r_0 \cdot r_1) \\ apk :=& pk_0 \cdot pk_1 = e^{sk_0 + sk_1} \end{aligned}$$

As such according to the definition of $\mathsf{V}_0(pk, h, s)$ we have:

$$(s_0 \cdot s_1)^e \cdot apk^{-h} \mod N = (r_0 \cdot r_1)^e \mod N = R_0 \cdot R_1$$

Therefore, the verification algorithm will return 1 as $h = \mathcal{H}(R_0 \cdot R_1, m)$. Hence, Eq. 4 holds.

Now let us see why $f_{\mathsf{dec\text{-}sig}}$ returns a valid signature under the public key pk_{1-i}. According to its definition we have:

$$s_{1-i} = sk_{1-i}^h \cdot (r_0 \cdot r_1)$$

As $h = \mathcal{H}((r_i \cdot r_{1-i})^e, m)$, we can conclude that the tuple (h, s_{1-i}) is a valid signature under the public key pk_{1-i}.

C Proofs

We now mention the proofs that were left out of the main body of the paper.

C.1 Proof of Lemma 4

Proof (Lemma 4). We prove the lemma by defining a series of game hops. The overall structure of the proof is similar to that of aEUF–CMA security proof in Lemma 2.

Game G_0: This game is equivalent to the original aWitExt, where the adversary $\mathcal{A}$ must output a valid forgery σ^* for a message m^* of his choice on input a pre-signature $\widetilde{\sigma}$ on m^*. For the forgery it must hold that the witness y, extracted from σ^* and $\widetilde{\sigma}$, is not in relation with the corresponding public statement Y^*, i.e, that $(Y^*, y) \notin \mathsf{R}$. The adversary has access to a pre-sign oracle $\mathcal{O}_{\mathrm{pS}}$ and a sign oracle $\mathcal{O}_{\mathrm{S}}$. Being in the random oracle model, all the algorithms of the scheme and the adversary have access to the random oracle $\mathcal{H}$.

Since G_0 corresponds to aSigForge, it follows that

$$\Pr[G_0 = 1] = \Pr[\mathsf{aWitExt}_{\mathcal{A},\mathsf{aSIG}^{\mathsf{ID},\mathsf{R}}}(n) = 1]. \tag{6}$$

```
Main
    $Q := \emptyset$
    $H := [\bot]$
    $(sk, pk) \leftarrow \mathsf{Gen}(1^n)$
    $(m^*, Y^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{S}}, \mathcal{O}_{\mathrm{pS}}}(pk)$
    $\widetilde{\sigma} \leftarrow \mathsf{pSign}_{sk}(m^*, Y^*)$
    $\sigma^* \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{S}}, \mathcal{O}_{\mathrm{pS}}}(\widetilde{\sigma})$
    $y := \mathsf{Ext}_{pk}(\sigma^*, \widetilde{\sigma}, Y^*)$
    $b_1 := \mathsf{Vrfy}_{pk}(m^*; \sigma^*)$
    $b_2 := m^* \notin Q$
    $b_3 := (Y^*, y) \notin \mathsf{R}$
    return $(b_1 \wedge b_2 \wedge b_3)$

$\mathcal{O}_{\mathrm{S}}(m)$
    $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$
    $Q := Q \cup \{m\}$
    return $\sigma$

$\mathcal{O}_{\mathrm{pS}}(m, Y)$
    $\widetilde{\sigma} \leftarrow \mathsf{pSign}_{sk}(m, Y)$
    $Q := Q \cup \{m\}$
    return $\widetilde{\sigma}$

$\mathcal{H}(x)$
    if $H[x] = \bot$
        $H[x] \leftarrow_{\$} \mathsf{ChSet}$
    return $H[x]$
```

Fig. 11: Formal definition of the game G_0.

Game G_1: This game is similar to the previous game except in the $\mathcal{O}_{\mathrm{pS}}$ oracle. The $\mathcal{O}_{\mathrm{pS}}$ oracle stores a copy H' of the list H before pSign_{sk} is executed. After this execution, the oracle obtains a pre-signature $\widetilde{\sigma}$ from which it extracts the randomness $R_{\mathsf{pre}} \leftarrow \mathsf{V}_0(pk, \widetilde{\sigma})$. The oracle further computes $R_{\mathsf{sign}} = f_{\mathsf{shift}}(R_{\mathsf{pre}}, Y)$ and checks if the random oracle $\mathcal{H}$ was already queried on the inputs $R_{\mathsf{pre}} \| m$ or $R_{\mathsf{sign}} \| m$ *before* the execution of pSign_{sk}, i.e., if $H'[R_{\mathsf{pre}} \| m] \neq \bot$ or $H'[R_{\mathsf{sign}} \| m] \neq \bot$ respectively. In this case the game aborts.

Claim. Let Bad_1 be the event that G_1 aborts in $\mathcal{O}_{\mathrm{pS}}$. Then $\Pr[\mathsf{Bad}_1] \leq \nu_1(n)$, where ν_1 is a negligible function in n.

Proof: We first recall that the output of P_1 (i.e., R_{pre}) is uniformly random from a super-polynomial set of size q in the security parameter. From this it follows that R_{sign} is distributed uniformly at random in the

$\mathcal{O}_{\mathrm{pS}}(m, Y)$ in $\boldsymbol{G_1}$

```
1 : $H' := H$
2 : $\widetilde{\sigma} \leftarrow \mathsf{pSign}_{sk}(m, Y)$
3 : $R_{\mathsf{pre}} \leftarrow \mathsf{V}_0(pk, \widetilde{\sigma})$
4 : $R_{\mathsf{sign}} = f_{\mathsf{shift}}(R_{\mathsf{pre}}, Y)$
5 : **if** $(H'[R_{\mathsf{pre}}||m] \neq \bot$
6 :     $\vee H'[R_{\mathsf{sign}}||m] \neq \bot)$
7 :     Abort
8 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
9 : **return** $\widetilde{\sigma}$
```

$\mathcal{O}_{\mathrm{pS}}(m, Y)$ in $\boldsymbol{G_2}$

```
1 : $H' := H$
2 : $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$
3 : $R \leftarrow \mathsf{V}_0(pk, \sigma)$
4 : $R' = f_{\mathsf{shift}}(R, Y)$
5 : **if** $(H'[R_{\mathsf{sign}}||m] \neq \bot$
6 :     $\vee H'[R_{\mathsf{pre}}||m] \neq \bot)$
7 :     Abort
8 : $x := R_{\mathsf{sign}}||m$
9 : $H[R_{\mathsf{pre}}||m] := H[x]$
10 : $H[x] \leftarrow_{\$} \mathsf{ChSet}$
11 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
12 : **return** $\sigma$
```

Main in $\boldsymbol{G_3}$

```
1 : $\mathcal{Q} := \emptyset$
2 : $H := [\bot]$
3 : $(sk, pk) \leftarrow \mathsf{Gen}(1^n)$
4 : $(m^*, Y^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{S}}, \mathcal{O}_{\mathrm{pS}}}(pk)$
5 : $H' := H$
6 : $\widetilde{\sigma} \leftarrow \mathsf{pSign}_{sk}(m^*, Y^*)$
7 : $R_{\mathsf{sign}} \leftarrow \mathsf{V}_0(pk, \widetilde{\sigma})$
8 : $R_{\mathsf{pre}} = f_{\mathsf{shift}}(R_{\mathsf{sign}}, Y^*)$
9 : **if** $(H'[R_{\mathsf{sign}}||m^*] \neq \bot$
10 :     $\vee H'[R_{\mathsf{pre}}||m^*] \neq \bot)$
11 :     Abort
12 : $\sigma^* \leftarrow \mathcal{A}(\widetilde{\sigma})$
13 : $y := \mathsf{Ext}_{pk}(\widetilde{\sigma}, \sigma^*, Y^*)$
14 : $b_1 := \mathsf{Vrfy}_{pk}(m^*; \sigma^*)$
15 : $b_2 := m^* \notin \mathcal{Q}$
16 : $b_3 := (Y^*, y) \notin \mathsf{R}$
17 : **return** $(b_1 \wedge b_2 \wedge b_3)$
```

Main in $\boldsymbol{G_4}$

```
1 : $\mathcal{Q} := \emptyset$
2 : $H := [\bot]$
3 : $(sk, pk) \leftarrow \mathsf{Gen}(1^n)$
4 : $(m^*, Y^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{S}}, \mathcal{O}_{\mathrm{pS}}}(pk)$
5 : $H' := H$
6 : $\sigma \leftarrow \mathsf{Sign}_{sk}(m^*)$
7 : $R_{\mathsf{sign}} \leftarrow \mathsf{V}_0(pk, \sigma)$
8 : $R_{\mathsf{pre}} = f_{\mathsf{shift}}(R_{\mathsf{sign}}, Y^*)$
9 : **if** $(H'[R_{\mathsf{sign}}||m^*] \neq \bot$
10 :     $\vee H'[R_{\mathsf{pre}}||m^*] \neq \bot)$
11 :     Abort
12 : $x := R_{\mathsf{sign}}||m^*$
13 : $H[R_{\mathsf{pre}}||m^*] := H[x]$
14 : $H[x] \leftarrow_{\$} \mathsf{ChSet}$
15 : $\sigma^* \leftarrow \mathcal{A}(\sigma)$
16 : $y := \mathsf{Ext}_{pk}(\widetilde{\sigma}, \sigma^*, Y^*)$
17 : $b_1 := \mathsf{Vrfy}_{pk}(m^*; \sigma^*)$
18 : $b_2 := m^* \notin \mathcal{Q}$
19 : $b_3 := (Y^*, y) \notin \mathsf{R}$
20 : **return** $(b_1 \wedge b_2 \wedge b_3)$
```

Fig. 12: Changes made in the game hops $\boldsymbol{G_1}$ to $\boldsymbol{G_4}$.

same set. Furthermore, $\mathcal{A}$ being a PPT algorithm, it can only make polynomially many queries to $\mathcal{H}$, $\mathcal{O}_{\mathrm{S}}$ and $\mathcal{O}_{\mathrm{pS}}$ oracles. Denoting ℓ as the total number of queries to $\mathcal{H}$, $\mathcal{O}_{\mathrm{S}}$ and $\mathcal{O}_{\mathrm{pS}}$, we have:

$$\begin{aligned}\Pr[\mathsf{Bad}_1] &= \Pr[H'[R_{\mathsf{pre}}||m] \neq \bot \vee H'[R_{\mathsf{sign}}||m] \neq \bot] \\ &\leq 2\frac{\ell}{q} \leq \nu_1(n)\end{aligned}$$

This follows from the fact that ℓ is polynomial in the security parameter. ■

Since games $\boldsymbol{G_1}$ and $\boldsymbol{G_0}$ are identical except in the case where Bad_1 occurs, it holds that $\Pr[\boldsymbol{G_0} = 1] \leq \Pr[\boldsymbol{G_1} = 1] + \nu_1(n)$.

Game $\boldsymbol{G_2}$: In this game, upon a query to the oracle $\mathcal{O}_{\mathrm{pS}}$, the game produces a full-signature instead of a pre-signature by executing Sign_{sk}, Accordingly, it adjusts the global list H to make the resulting full-signature "look like" a pre-signature from the point of view of the adversary $\mathcal{A}$. This is done as follows:

1. It sets $H[R_{\mathsf{pre}}\|m]$ to the value stored at position $H[R_{\mathsf{sign}}\|m]$.
2. It sets $H[R_{\mathsf{sign}}\|m]$ to a fresh value chosen uniformly at random.

The above programming makes sense as it complies with our definition of f_{shift} (as already argued in Lemma 2). Note that $\mathcal{A}$ can only notice that $\mathcal{H}$ was programmed if it was previously queried on either $R_{\mathsf{pre}}\|m$ or $R_{\mathsf{sign}}\|m$. But as described in the previous game, we abort if such an event happens. Hence, we have that $\Pr[\boldsymbol{G_1} = 1] = \Pr[\boldsymbol{G_2} = 1]$.

Game $\boldsymbol{G_3}$: In this game, we impose the same checks as in $\boldsymbol{G_1}$, but upon generating the pre-signature on the challenge message. Therefore, it follows that $\Pr[\boldsymbol{G_3} = 1] \leq \Pr[\boldsymbol{G_2} = 1] + \nu_2(n)$.

Game $\boldsymbol{G_4}$: Similar to game $\boldsymbol{G_2}$, we generate a signature instead of a pre-signature in the challenge phase of this game and program $\mathcal{H}$ such that the full-signature would look like a correct pre-signature from $\mathcal{A}$'s point of view. Hence, we have $\Pr[\boldsymbol{G_4} = 1] = \Pr[\boldsymbol{G_3} = 1]$.

```
$\mathcal{S}^{\mathsf{SIG}^{\mathsf{ID}},\mathcal{H}^{\mathsf{ID}}}(pk)$
$\mathcal{Q} := \emptyset$
$H := [\perp]$
$(sk, pk) \leftarrow \mathsf{Gen}(1^n)$
$(m^*, Y^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{S}},\mathcal{O}_{\mathrm{pS}}}(pk)$
$H' := H$
$\sigma \leftarrow \mathsf{SIG}^{\mathsf{ID}}(m^*)$
$R_{\mathsf{sign}} \leftarrow \mathsf{V}_0(pk, \sigma)$
$R_{\mathsf{pre}} = f_{\mathsf{shift}}(R_{\mathsf{sign}}, Y^*)$
**if** $(H'[R_{\mathsf{sign}}||m^*] \neq \perp$
    $\vee\, H'[R_{\mathsf{pre}}||m^*] \neq \perp)$
    Abort
$x := R_{\mathsf{sign}}||m^*$
$H[R_{\mathsf{pre}}||m^*] := \mathcal{H}^{\mathsf{ID}}[x]$
$H[x] \leftarrow_{\$} \mathcal{H}^{\mathsf{ID}}(R_{\mathsf{pre}}||m^*)$
$\sigma^* \leftarrow \mathcal{A}(\sigma)$
**return** $(m^*, \sigma^*)$
```

```
$\mathcal{O}_{\mathrm{pS}}(m, Y)$
$H' := H$
$\sigma \leftarrow \mathsf{SIG}^{\mathsf{ID}}(m)$
$R_{\mathsf{sign}} \leftarrow \mathsf{V}_0(pk, \sigma)$
$R_{\mathsf{pre}} = f_{\mathsf{shift}}(R_{\mathsf{sign}}, Y)$
**if** $(H'[R_{\mathsf{sign}}||m] \neq \perp$
    $\vee\, H'[R_{\mathsf{pre}}||m] \neq \perp)$
    Abort
$x := R_{\mathsf{sign}}||m$
$H[R_{\mathsf{pre}}||m] := \mathcal{H}^{\mathsf{ID}}[x]$
$H[x] \leftarrow_{\$} \mathcal{H}^{\mathsf{ID}}(R_{\mathsf{pre}}||m)$
$\mathcal{Q} := \mathcal{Q} \cup \{m\}$
**return** $\sigma$

$\mathcal{H}(x)$
**if** $H[x] = \perp$
    $H[x] := \mathcal{H}^{\mathsf{ID}}(x)$
**return** $H[x]$
```

```
$\mathcal{O}_{\mathrm{S}}(m)$
$\sigma \leftarrow \mathsf{SIG}^{\mathsf{ID}}(m)$
$R_{\mathsf{sign}} \leftarrow \mathsf{V}_0(pk, \sigma)$
$x := R_{\mathsf{sign}}||m$
$H[x] := \mathcal{H}^{\mathsf{ID}}(x)$
$\mathcal{Q} := \mathcal{Q} \cup \{m\}$
**return** $\sigma$
```

Fig. 13: The final simulation.

Now that the transition from the original $\mathsf{aWitExt}$ experiment (game $\boldsymbol{G_0}$) to game $\boldsymbol{G_4}$ is indistinguishable, it only remains to show the existence of a simulator $\mathcal{S}$ that can perfectly simulate $\boldsymbol{G_4}$ and uses $\mathcal{A}$ to win the $\mathsf{strongSigForge}$ game. In Fig. 13, we describe the simulator's code in a concise way.

We emphasize that the main differences between the simulation and $\boldsymbol{G_4}$ are syntactical. Namely, instead of generating the public and secret keys and computing the algorithm Sign_{sk} and the random oracle $\mathcal{H}$, $\mathcal{S}$ uses its oracles $\mathsf{SIG}^{\mathsf{ID}}$ and $\mathcal{H}^{\mathsf{ID}}$. Therefore $\mathcal{S}$ perfectly simulates $\boldsymbol{G_4}$. It remains to show that $\mathcal{S}$ can use the forgery output by $\mathcal{A}$ to win the $\mathsf{strongSigForge}$ game.

Claim. (m^*, σ^*) constitutes a valid forgery in game $\mathsf{strongSigForge}$.

Proof: To prove this claim, we show that the tuple (m^*, σ^*) has not been returned by the oracle $\mathsf{SIG}^{\mathsf{ID}}$ before. First note that $\mathcal{A}$ wins the experiment if it has not queried on the challenge message m^* to $\mathcal{O}_{\mathrm{pS}}$ or $\mathcal{O}_{\mathrm{S}}$. Therefore, $\mathsf{SIG}^{\mathsf{ID}}$ is queried on m^* only during the challenge phase. If $\mathcal{A}$ outputs a forgery σ^* that is equal to the signature σ output by $\mathsf{SIG}^{\mathsf{ID}}$, it would lose the game since this signature would not be valid given the fact that random oracle is programmed. Hence, $\mathsf{SIG}^{\mathsf{ID}}$ has never output σ^* when queried on m^* before, thus making (m^*, σ^*) a valid forgery for game $\mathsf{strongSigForge}$. ■

From games $\boldsymbol{G_0} - \boldsymbol{G_4}$ we have that $\Pr[\boldsymbol{G_0} = 1] \leq \Pr[\boldsymbol{G_4} = 1] + \nu(n)$, where $\nu(n) = \nu_1(n) + \nu_2(n)$ is a negligible function in n. Since $\mathcal{S}$ simulates game $\boldsymbol{G_4}$ perfectly, we also have that $\Pr[\boldsymbol{G_4} = 1] = \Pr[\mathsf{strongSigForge}_{\mathcal{S}^{\mathcal{A}},\mathsf{SIG}}(n) = 1]$ Combining this with Eq. (6) in game $\boldsymbol{G_0}$, we obtain the following:

$$\Pr[\mathsf{aWitExt}_{\mathcal{A},\mathsf{aSIG}^{\mathsf{ID},R}}(n) = 1] \leq \Pr[\mathsf{strongSigForge}_{\mathcal{S}^{\mathcal{A}},\mathsf{SIG}^{\mathsf{ID}}}(n) = 1] + \nu(n).$$

C.2 Proof of Lemma 7

Proof (Lemma 7). This proof is similar to the proof of Lemmas 1 and 5.

Fix an arbitrary message m and a statement/witness pair $(Y, y) \in \mathsf{R}$. Let $(sk_0, pk_0) \leftarrow \mathsf{Gen}(1^n)$, $(sk_1, pk_1) \leftarrow \mathsf{Gen}(1^n)$, $apk \leftarrow \mathsf{KAg}(pk_0, pk_1)$, $(h, \widetilde{\sigma}) \leftarrow \mathsf{pSign}_{\langle sk_0, sk_1 \rangle}(m, Y)$, $(h, \sigma) := \mathsf{Adapt}_{apk}(\widetilde{\sigma}, y)$ and $y' := \mathsf{Ext}_{apk}(\sigma, \widetilde{\sigma}, Y)$. From Figure 7 we know that $y' = f_{\mathsf{ext}}(\sigma, \widetilde{\sigma})$ where:

$$\begin{aligned}
&\sigma := f_{\mathsf{adapt}}(\widetilde{\sigma}, y), && \widetilde{\sigma} \leftarrow f_{\mathsf{com\text{-}sig}}(h, (\widetilde{\sigma}_0, \widetilde{\sigma}_1)),\\
&\widetilde{\sigma}_0 \leftarrow \mathsf{P}_2(sk_0, R_0, h, St_0), && \widetilde{\sigma}_1 \leftarrow \mathsf{P}_2(sk_1, R_1, h, St_1),\\
&h := \mathcal{H}(R_{\mathsf{sign}}, m), &&\\
&R_{\mathsf{sign}} := f_{\mathsf{shift}}(R_{\mathsf{pre}}, Y), && R_{\mathsf{pre}} = f_{\mathsf{com\text{-}rand}}(R_0, R_1),\\
&(R_0, St_0, R_1) \leftarrow \Pi_{\mathsf{Rand\text{-}Exc}\langle sk_0, sk_1 \rangle} \text{ and } (R_1, St_1, R_0) \leftarrow \Pi_{\mathsf{Rand\text{-}Exc}\langle sk_1, sk_0 \rangle}.
\end{aligned}$$

Let us first prove that $\mathsf{pVrfy}_{apk}(m, Y; \widetilde{\sigma}) = 1$. From Eq. (4) and the completeness of the ID scheme, we know that $\mathsf{V}_0(apk, h, \widetilde{\sigma}) = R_{\mathsf{pre}}$. Hence,

$$\mathcal{H}(f_{\mathsf{shift}}(\mathsf{V}_0(apk, h, \widetilde{\sigma}), Y), m) = \mathcal{H}(f_{\mathsf{shift}}(R_{\mathsf{pre}}, Y), m) = \mathcal{H}(R_{\mathsf{sign}}, m) = h \tag{7}$$

which is what we needed to prove.

Let us now show that $\mathsf{Vrfy}_{apk}(m; \sigma) = 1$. By Fig. 2, we need to show that $h = \mathcal{H}(\mathsf{V}_0(apk, h, \sigma), m)$. This follows from the property of f_{shift}, f_{adapt} (c.f. Eq. (1)) and Eq. (3) as follows:

$$\begin{aligned}
\mathcal{H}(\mathsf{V}_0(apk, h, \sigma), m) =& \mathcal{H}(\mathsf{V}_0(apk, h, f_{\mathsf{adapt}}(\widetilde{\sigma}, y)), m)\\
&\overset{(1)}{=} \mathcal{H}(f_{\mathsf{shift}}(\mathsf{V}_0(apk, h, \widetilde{\sigma}), Y), m) \overset{(7)}{=} h.
\end{aligned}$$

Finally, we need to show that $(Y, y') \in \mathsf{R}$. This follows from Eq. (2) since:

$$y' = f_{\mathsf{ext}}(\sigma, \widetilde{\sigma}) = f_{\mathsf{ext}}(f_{\mathsf{adapt}}(\widetilde{\sigma}, y), \widetilde{\sigma}) \overset{(2)}{=} y.$$

C.3 Proof of Lemma 8

Proof (Lemma 8).

The proof of this Lemma is similar to the proof of Lemmas 2 and 6. We prove this Lemma by a reduction to the $\mathsf{SUF\text{-}CMA}$ security of the underlying scheme SIG. More precisely we use the adversary who can win the unforgeability of two-party adaptor signature scheme with aggregatable public keys against our scheme to break the $\mathsf{SUF\text{-}CMA}$ security of the SIG scheme. In particular, we construct a simulator who simulates the unforgeability game $\mathsf{SigForge}^b_{\mathcal{A},\mathsf{aSIG}_2}(n)$ while having access to the oracles from the $\mathsf{SUF\text{-}CMA}$ game. We note that the $\Pi_{\mathsf{Rand\text{-}Exc}}$ protocol must satisfy two properties (similar to [29]). First, the commitment must be extractable for the simulator who is playing the $\mathsf{SUF\text{-}CMA}$ game. This is necessary in order to program the random oracle in time before the adversary can compute the combined randomness. Second, the zero-knowledge proof must be simulatable.

Game G_0: This game represents the original $\mathsf{aSigForge}$ experiment, where the adversary $\mathcal{A}$ must output a valid forgery for a message m of his choice, while having access to the interactive pre-signature and signature oracles $\mathcal{O}_{\mathrm{pS}}$ and $\mathcal{O}_{\mathrm{S}}$. As we are in the random oracle model, all the algorithms of the scheme and the adversary have access to the random oracle $\mathcal{H}$.

$$\Pr[G_0 = 1] = \Pr[\mathsf{aSigForge}^b_{\mathcal{A},\mathsf{aSIG}_2}(n) = 1]$$

Game G_1: This game is similar to G_0 except upon the adversary outputting a forgery σ^*, the game aborts if:

$$\mathsf{Adapt}_{apk}(\widetilde{\sigma}, y) = \sigma^*$$

As in Lemma 2 in this case our simulator can break the hardness of the relation R.

Claim. Let Bad_1 be the event where $\boldsymbol{G_1}$ aborts. Then $\Pr[\mathsf{Bad}_1] \leq \nu_1(n)$, where ν_1 is a negligible function in n.

Proof: this proof is analogous to the proof of claim 3.2. ■

Game G_2**:** This game is similar to G_1 except in the $\mathcal{O}_{\mathrm{S}}$ oracle. All signing queries are simulated as follows:

1. Generate a signature on message m.
2. Upon generating the signature $\sigma_{1-b} \coloneqq (h, \sigma_{1-b})$, extract the randomness $R_{1-b} \coloneqq \mathsf{V}_0(pk_{1-b}, \sigma_{1-b})$.
3. Program the random oracle $\mathcal{H}$ such that a query on (R_{1-b}, m), instead of h, returns a fresh, randomly chosen value.
4. Start executing the $\mathit{\Pi}_{\mathsf{Rand\text{-}Exc}}$ procedure using R_{1-b} and by simulating the zero-knowledge proof.
5. Extract the public randomness of the adversary R_b.
6. Compute the combined randomness, $R_{\mathsf{sign}} = f_{\mathsf{com\text{-}rand}}(R_b, R_{1-b})$.
7. Program the random oracle $\mathcal{H}$ such that a query on $(R_{\mathsf{sign}} \| m)$ returns h.
8. Continue executing the $\mathit{\Pi}_{\mathsf{Rand\text{-}Exc}}$ protocol.
9. Return (h, σ_{1-b}) to $\mathcal{A}$.

We refer the reader to the proof of Lemma 6 for the simulation of the different cases i.e., $b = 0$ or $b = 1$. As mentioned in the proof of Lemma 2 and 6 the simulation of the signing oracle is indistinguishable from the original execution of the signing procedure for the adversary except with negligible probability that the programming of the random oracle fails, or the commitment extraction fails, or the simulation of the zero-knowledge proof fails. A union bound on the probability of all these events ensure that $\mathcal{A}$'s distinguishing advantage remains negligible. Hence, we have:

$$\Pr[G_1 = 1] \leq \Pr[G_2 = 1] + \nu_2(n).$$

Game G_3**:** This game is similar to G_2 except in the $\mathcal{O}_{\mathrm{pS}}$ oracle. All pre-signing queries are simulated similar to the steps mentioned in Game G_2 except step 6 is replaced with the following step:

6. Compute the combined randomness, $R_{\mathsf{pre}} = f_{\mathsf{com\text{-}rand}}(R_b, R_{1-b})$ and $R_{\mathsf{sign}} = f_{\mathsf{shift}}(R_{\mathsf{pre}}, Y)$.

As in Game G_2 The adversary can only distinguish this game and the previous game only with negligible probability and hence, we have:

$$\Pr[G_2 = 1] \leq \Pr[G_3 = 1] + \nu_3(n).$$

Game G_4**:** This game is similar to G_3 except during the generation of the pre-signature on the challenge message m^*. The generation of the pre-signature is modifies as in game G_3. Therefore, we can conclude that:

$$\Pr[G_3 = 1] \leq \Pr[G_4 = 1] + \nu_4(n).$$

Breaking the $\mathsf{SUF\text{–}CMA}$**:** We can now build our simulator which can break the $\mathsf{SUF\text{–}CMA}$ of the underlying scheme. Our simulator does not generate the secret and public keys but receives the public key from its oracle. Therefore, instead of generating the signatures during the signing and pre-signing queries, $\mathcal{S}$ queries its $\mathsf{SUF\text{–}CMA}$ oracle in order to receive the corresponding signature.

Now upon the adversary producing a valid forgery (h^*, σ^*) on message m^*, the simulator executes $f_{\mathsf{dec\text{-}sig}}(sk_b, pk_b, \sigma^*)$ (from Eq. 5) to extract the partial signature σ^*_{1-b} and submits the output (m^*, σ^*_{1-b}) as its own forgery. The adversary wins this game if she has not queried m^* before. Therefore, according to the simulation of the signing queries, $\mathcal{S}$ has also not queried its oracle on the message m^*. Eq. 5 implies that the extracted signature σ^*_{1-b} is a valid signature for message m^* under the public key pk_{1-b}. Furthermore, σ^*_{1-b} is not previously outputted by the $\mathsf{SUF\text{–}CMA}$ oracle when queried on the message m^* (since in this case the signature will not be valid due to the programming of the random oracle). Since $\mathcal{A}$ makes only a polynomial number of queries to the signing oracle and $\mathcal{H}$, $\mathcal{S}$ can win the $\mathsf{SUF\text{–}CMA}$ experiment with the probability $\Pr[\mathsf{aSigForge}^b_{\mathcal{A},\mathsf{aSIG}_2}(n) = 1] - \nu_1(n)\nu_3(n) - \nu_4(n)$. Thus, the $\mathsf{SUF\text{–}CMA}$ security of the $\mathsf{SIG}^{\mathsf{ID}}$ implies that $\Pr[\mathsf{aSigForge}^b_{\mathcal{A},\mathsf{aSIG}_2}(n) = 1] \leq \nu(n)$, where $\nu(n)$ is a negligible function.

C.4 Proof of Lemma 9

Proof (Lemma 9). This proof is analogous to the proof of Lemma 3. Assume $\mathsf{pVrfy}_{apk}(m, Y; \widetilde{\sigma}) = 1$. This means that $h = \mathcal{H}(f_{\mathsf{shift}}(\mathsf{V}_0(apk, h, \tilde{s}), Y), m)$. For any valid pair $(Y, y) \in R$, we can now use the homomorphic property from Eq. (1). Specifically, for such a pair $(Y, y) \in R$, plugging $f_{\mathsf{shift}}(\mathsf{V}_0(apk, h, \tilde{s}), Y) = \mathsf{V}_0(apk, h, f_{\mathsf{adapt}}(\tilde{s}, y))$ in the above equation implies that:

$$h = \mathcal{H}(\mathsf{V}_0(apk, h, f_{\mathsf{adapt}}(\tilde{s}, y)), m)$$

This directly implies $\mathsf{Vrfy}_{apk}(m; \sigma) = 1$, where $s = f_{\mathsf{adapt}}(\tilde{s}, y)$ and $\sigma = (h, s)$. Therefore, adapting the valid pre-signature would also result in a valid full-signature.

C.5 Proof of Lemma 10

Proof (Lemma 10).

The proof of this Lemma is similar to the proof of Lemmas 4 and 6 and follows the same approach as in the proof of Lemma 8. The only difference between this proof and the proof of Lemma 8 is that no reduction is made to the hardness of the relation R. As explained in the proof of Lemmas 4, this is because in the $\mathsf{aWitExt}^{b}_{\mathcal{A},\mathsf{aSIG}_2}(n)$ experiment, Y^* is given by the adversary $\mathcal{A}$ who must forge a signature such that $(Y^*, \mathsf{Ext}_{apk}(\sigma^*, \widetilde{\sigma}, Y^*)) \notin \mathsf{R}$.

More precisely, we prove this lemma by a reduction to the SUF–CMA security of the underlying schemes. More precisely we use the adversary who can win the unforgeability of two-party adaptor signature scheme with aggregatable public keys against our scheme to break the SUF–CMA security of the signature schemes. In particular, we construct a simulator who simulates the unforgeability game $\mathsf{aWitExt}^{b}_{\mathcal{A},\mathsf{aSIG}_2}(n)$ while having access to the SUF–CMA game oracles. We note that the $\Pi_{\mathsf{Rand\text{-}Exc}}$ protocol must satisfy two properties (similar to [29]). First, the commitment must be extractable for the simulator who is playing the SUF–CMA game. This is necessary in order to program the random oracle in time before the adversary can compute the combined randomness. Second, the zero-knowledge proof used must be simulatable.

Game G_0: This game represents the original aWitExt experiment, where the adversary $\mathcal{A}$ must output a valid forgery for a message m of his choice, while having access to the interactive pre-signature and signature oracles $\mathcal{O}_{\mathrm{pS}}$ and $\mathcal{O}_{\mathrm{S}}$. As we are in the random oracle model, all the algorithms of the scheme and the adversary have access to the random oracle $\mathcal{H}$.

$$\Pr[G_0 = 1] = \Pr[\mathsf{aWitExt}^{b}_{\mathcal{A},\mathsf{aSIG}_2}(n) = 1]$$

Game G_1: This game is similar to G_0 except in the $\mathcal{O}_{\mathrm{S}}$ oracle. All signing queries are simulated as follows:

1. Generate a signature on message m.
2. Upon generating the signature $\sigma_{1-b} := (h, \sigma_{1-b})$, extract the randomness $R_{1-b} := \mathsf{V}_0(pk_{1-b}, \sigma_{1-b})$.
3. Program the random oracle $\mathcal{H}$ such that a query on (R_{1-b}, m), instead of h, returns a fresh, randomly chosen value.
4. Start executing the $\Pi_{\mathsf{Rand\text{-}Exc}}$ procedure using R_{1-b} and by simulating the zero-knowledge proof.
5. Extract the public randomness of the adversary R_b.
6. Compute the combined randomness, $R_{\mathsf{sign}} = f_{\mathsf{com\text{-}rand}}(R_b, R_{1-b})$.
7. Program the random oracle $\mathcal{H}$ such that a query on $(R_{\mathsf{sign}} \| m)$ returns h.
8. Continue executing the $\Pi_{\mathsf{Rand\text{-}Exc}}$ protocol.
9. Return (h, σ_{1-b}) to $\mathcal{A}$.

We refer the reader to the proof of Lemma 6 for the simulation of the different cases i.e., $b = 0$ or $b = 1$. As mentioned in the proof of Lemma 4 and 6 the simulation of the signing oracle is indistinguishable from the original execution of the signing procedure for the adversary except with negligible probability that the random oracle programming fails, or the commitment extraction fails, or the simulation of the zero-knowledge proof fails. Hence, we have:

$$\Pr[G_0 = 1] \leq \Pr[G_1 = 1] + \nu_1(n).$$

Game G_2: This game is similar to G_1 except in the $\mathcal{O}_{\mathrm{pS}}$ oracle. All pre-signing queries are simulated similar to the steps mentioned in Game G_1 except step 6 is replaced with the following step:

6. Compute the combined randomness, $R_{\mathsf{pre}} = f_{\mathsf{com\text{-}rand}}(R_b, R_{1-b})$ and $R_{\mathsf{sign}} = f_{\mathsf{shift}}(R_{\mathsf{pre}}, Y)$.

As in Game G_1 The adversary can only distinguish this game and the previous game only with the negligible probability that the programming of the random oracle fails. Hence we have:

$$\Pr[G_1 = 1] \leq \Pr[G_2 = 1] + \nu_2(n).$$

Game G_3: This game is similar to G_2 except during the generation of the pre-signature on the challenge message m^*. The generation of the pre-signature is modifies as in game G_3. Therefore, we can conclude that:

$$\Pr[G_2 = 1] \leq \Pr[G_3 = 1] + \nu_3(n).$$

Breaking the SUF–CMA**:** We can now build our simulator which can break the SUF–CMA of the underlying scheme. Our simulator does not generate the secret and public keys but receives the public key from its oracle. Therefore, instead of generating the signatures during the signing and pre-signing queries, $\mathcal{S}$ queries its SUF–CMA oracle in order to receive the corresponding signature.

Now upon the adversary producing a valid forgery (h^*, σ^*) on message m^*, the simulator executes $f_{\mathsf{dec\text{-}sig}}(sk_b, pk_b, \sigma^*)$ (from Eq. 5) to extract the partial signature σ^*_{1-b} and submits the output (m^*, σ^*_{1-b}) as its own forgery. The adversary wins this game if she has not queried m^* before. Therefore, according to the simulation of the signing queries, $\mathcal{S}$ has also not queried its oracle on the message m^*. Eq. 5 implies that the extracted signature σ^*_{1-b} is a valid signature for message m^* under the public key pk_{1-b}. Furthermore, σ^*_{1-b} is not previously outputted by the SUF–CMA oracle when queried on the message m^* (since in this case the signature will not be valid due to the programming of the random oracle). Since $\mathcal{A}$ makes only a polynomial number of queries to the signing oracle and $\mathcal{H}$, $\mathcal{S}$ can win the SUF–CMA experiment with the probability $\Pr[\mathsf{aWitExt}^b_{\mathcal{A},\mathsf{aSIG}_2}(n)(n) = 1] - \nu_2(n) - \nu_1(n)$. Thus, the SUF–CMA security of the $\mathsf{SIG}^{\mathsf{ID}}$ implies $\Pr[\mathsf{aWitExt}^b_{\mathcal{A},\mathsf{aSIG}_2}(n)(n)(n) = 1] \leq \nu(n)$, where $\nu(n)$ is a negligible function.

D Additional material: Proof of Lemma 2

The original aEUF–CMA game from Lemma 2 is depicted in Fig. 14, the changes made in Games 1 to Game 5 are described in Fig. 15 and the final simulation is in Fig. 16

```
Main
1 : $\mathcal{Q} := \emptyset$
2 : $H := [\perp]$
3 : $(sk, pk) \leftarrow \mathsf{Gen}(1^n)$
4 : $m^* \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{S}}, \mathcal{O}_{\mathrm{pS}}}(pk)$
5 : $(Y, y) \leftarrow \mathsf{GenR}(1^n)$
6 : $\widetilde{\sigma} \leftarrow \mathsf{pSign}_{sk}(m^*, Y)$
7 : $\sigma^* \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{S}}, \mathcal{O}_{\mathrm{pS}}}(\widetilde{\sigma}, Y)$
8 : $b := \mathsf{Vrfy}_{pk}(m^*; \sigma^*)$
9 : return $(m^* \notin \mathcal{Q} \land b)$

$\mathcal{O}_{\mathrm{S}}(m)$
1 : $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$
2 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
3 : return $\sigma$

$\mathcal{O}_{\mathrm{pS}}(m, Y)$
1 : $\widetilde{\sigma} \leftarrow \mathsf{pSign}_{sk}(m, Y)$
2 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
3 : return $\widetilde{\sigma}$

$\mathcal{H}(x)$
1 : if $H[x] = \perp$
2 :   $H[x] \leftarrow_{\$} \mathsf{ChSet}$
3 : return $H[x]$
```

Fig. 14: Formal definition of the game $\boldsymbol{G_0}$.

Main in $\boldsymbol{G_1}$

```
1 : $\mathcal{Q} := \emptyset$
2 : $H := [\bot]$
3 : $(sk, pk) \leftarrow \mathsf{Gen}(1^n)$
4 : $m^* \leftarrow \mathcal{A}^{\mathcal{O}_\mathrm{S}, \mathcal{O}_\mathrm{pS}}(pk)$
5 : $(Y, y) \leftarrow \mathsf{GenR}(1^n)$
6 : $\widetilde{\sigma} \leftarrow \mathsf{pSign}_{sk}(m^*, Y)$
7 : $\sigma^* \leftarrow \mathcal{A}^{\mathcal{O}_\mathrm{S}, \mathcal{O}_\mathrm{pS}}(\widetilde{\sigma}, Y)$
8 : **if** $\mathsf{Adapt}_{pk}(\widetilde{\sigma}, y) = \sigma^*$
9 :    Abort
10 : $b := \mathsf{Vrfy}_{pk}(m^*; \sigma^*)$
11 : **return** $(m^* \notin \mathcal{Q} \wedge b)$
```

Main in $\boldsymbol{G_4}$

```
1 : $\mathcal{Q} := \emptyset$
2 : $H := [\bot]$
3 : $(sk, pk) \leftarrow \mathsf{Gen}(1^n)$
4 : $m^* \leftarrow \mathcal{A}^{\mathcal{O}_\mathrm{S}, \mathcal{O}_\mathrm{pS}}(pk)$
5 : $(Y, y) \leftarrow \mathsf{GenR}(1^n)$
6 : $H' := H$
7 : $\widetilde{\sigma} \leftarrow \mathsf{pSign}_{sk}(m^*, Y)$
8 : $R_\mathsf{sign} \leftarrow \mathsf{V}_0(pk, \widetilde{\sigma})$
9 : $R_\mathsf{pre} = f_\mathsf{shift}(R_\mathsf{sign}, Y)$
10 : **if** $(H'[R_\mathsf{sign}||m^*] \neq \bot$
11 :    $\vee H'[R_\mathsf{pre}||m^*] \neq \bot)$
12 :    Abort
13 : $\sigma^* \leftarrow \mathcal{A}^{\mathcal{O}_\mathrm{S}, \mathcal{O}_\mathrm{pS}}(\widetilde{\sigma}, Y)$
14 : **if** $\mathsf{Adapt}(\widetilde{\sigma}, y) = \sigma^*$
15 :    Abort
16 : $b := \mathsf{Vrfy}_{pk}(m^*; \sigma^*)$
17 : **return** $(m^* \notin \mathcal{Q} \wedge b)$
```

$\mathcal{O}_\mathrm{pS}(m, Y)$ in $\boldsymbol{G_2}$

```
1 : $H' := H$
2 : $\widetilde{\sigma} \leftarrow \mathsf{pSign}_{sk}(m, Y)$
3 : $R_\mathsf{pre} \leftarrow \mathsf{V}_0(pk, \widetilde{\sigma})$
4 : $R_\mathsf{sign} = f_\mathsf{shift}(R_\mathsf{pre}, Y)$
5 : **if** $(H'[R_\mathsf{pre}||m] \neq \bot$
6 :    $\vee H'[R_\mathsf{sign}||m] \neq \bot)$
7 :    Abort
8 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
9 : **return** $\widetilde{\sigma}$
```

Main in $\boldsymbol{G_5}$

```
1 : $\mathcal{Q} := \emptyset$
2 : $H := [\bot]$
3 : $(sk, pk) \leftarrow \mathsf{Gen}(1^n)$
4 : $m^* \leftarrow \mathcal{A}^{\mathcal{O}_\mathrm{S}, \mathcal{O}_\mathrm{pS}}(pk)$
5 : $(Y, y) \leftarrow \mathsf{GenR}(1^n)$
6 : $H' := H$
7 : $\sigma \leftarrow \mathsf{Sign}_{sk}(m^*)$
8 : $R_\mathsf{sign} \leftarrow \mathsf{V}_0(pk, \sigma')$
9 : $R_\mathsf{pre} = f_\mathsf{shift}(R_\mathsf{sign}, Y)$
10 : **if** $(H'[R_\mathsf{sign}||m^*] \neq \bot$
11 :    $\vee H'[R_\mathsf{pre}||m^*] \neq \bot)$
12 :    Abort
13 : $x := R_\mathsf{sign}||m^*$
14 : $H[R_\mathsf{pre}||m^*] := H[x]$
15 : $H[x] \leftarrow_\$ \mathsf{ChSet}$
16 : $\sigma^* \leftarrow \mathcal{A}^{\mathcal{O}_\mathrm{S}, \mathcal{O}_\mathrm{pS}}(\sigma, Y)$
17 : **if** $\mathsf{Adapt}(\sigma, y) = \sigma^*$
18 :    Abort
19 : $b := \mathsf{Vrfy}_{pk}(m^*; \sigma^*)$
20 : **return** $(m^* \notin \mathcal{Q} \wedge b)$
```

$\mathcal{O}_\mathrm{pS}(m, Y)$ in $\boldsymbol{G_3}$

```
1 : $H' := H$
2 : $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$
3 : $R_\mathsf{pre} \leftarrow \mathsf{V}_0(pk, \sigma)$
4 : $R_\mathsf{sign} = f_\mathsf{shift}(R, Y)$
5 : **if** $(H'[R_\mathsf{sign}||m] \neq \bot$
6 :    $\vee H'[R_\mathsf{pre}||m] \neq \bot)$
7 :    Abort
8 : $x := R_\mathsf{sign}||m$
9 : $H[R_\mathsf{pre}||m] := H[x]$
10 : $H[x] \leftarrow_\$ \mathsf{ChSet}$
11 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
12 : **return** $\sigma$
```

Fig. 15: Changes made in the game hops $\boldsymbol{G_1}$ to $\boldsymbol{G_5}$.

```
$\mathcal{S}^{\mathsf{SIG}^{\mathsf{ID}},\mathcal{H}^{\mathsf{ID}}}(pk)$
1 : $\mathcal{Q} := \emptyset$
2 : $H := [\perp]$
3 : $(sk, pk) \leftarrow \mathsf{Gen}(1^n)$
4 : $m^* \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{S}},\mathcal{O}_{\mathrm{pS}}}(pk)$
5 : $(Y, y) \leftarrow \mathsf{GenR}(1^n)$
6 : $H' := H$
7 : $\sigma \leftarrow \mathsf{SIG}^{\mathsf{ID}}(m^*)$
8 : $R_{\mathsf{sign}} \leftarrow \mathsf{V}_0(pk, \sigma)$
9 : $R_{\mathsf{pre}} = f_{\mathsf{shift}}(R_{\mathsf{sign}}, Y)$
10 : if $(H'[R_{\mathsf{sign}}||m^*] \neq \perp$
11 :   $\vee\, H'[R_{\mathsf{pre}}||m^*] \neq \perp)$
12 :   Abort
13 : $x := R_{\mathsf{sign}}||m^*$
14 : $H[R_{\mathsf{pre}}||m^*] := \mathcal{H}^{\mathsf{ID}}[x]$
15 : $H[x] \leftarrow_{\$} \mathcal{H}^{\mathsf{ID}}(R_{\mathsf{pre}}||m^*)$
16 : $\sigma^* \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{S}},\mathcal{O}_{\mathrm{pS}}}(\sigma, Y)$
17 : if $\mathsf{Adapt}(\widetilde{\sigma}, y) = \sigma^*$
18 :   Abort
19 : return $(m^*, \sigma^*)$
```

```
$\mathcal{O}_{\mathrm{S}}(m)$
1 : $\sigma \leftarrow \mathsf{SIG}^{\mathsf{ID}}(m)$
2 : $R_{\mathsf{sign}} \leftarrow \mathsf{V}_0(pk, \sigma)$
3 : $x := R_{\mathsf{sign}}||m$
4 : $H[x] := \mathcal{H}^{\mathsf{ID}}(x)$
5 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
6 : return $\sigma$
```

```
$\mathcal{O}_{\mathrm{pS}}(m, Y)$
1 : $H' := H$
2 : $\sigma \leftarrow \mathsf{SIG}^{\mathsf{ID}}(m)$
3 : $R_{\mathsf{sign}} \leftarrow \mathsf{V}_0(pk, \sigma)$
4 : $R_{\mathsf{pre}} = f_{\mathsf{shift}}(R_{\mathsf{sign}}, Y)$
5 : if $(H'[R_{\mathsf{sign}}||m] \neq \perp$
6 :   $\vee\, H'[R_{\mathsf{pre}}||m] \neq \perp)$
7 :   Abort
8 : $x := R_{\mathsf{sign}}||m$
9 : $H[R_{\mathsf{pre}}||m] := \mathcal{H}^{\mathsf{ID}}[x]$
10 : $H[x] \leftarrow_{\$} \mathcal{H}^{\mathsf{ID}}(R_{\mathsf{pre}}||m)$
11 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
12 : return $\sigma$
```

```
$\mathcal{H}(x)$
1 : if $H[x] = \perp$
2 :   $H[x] := \mathcal{H}^{\mathsf{ID}}(x)$
3 : return $H[x]$
```

Fig. 16: The final simulation.