# Adaptor Signatures Meet BLS: Enabling Efficient Blockchain Applications with Unique Adaptor Signatures

Javier Gomez-Martinez[1], Erkan Tairi[2⋆], Pedro Moreno-Sanchez[1,3], and Clara Schneidewind[3]

[1] IMDEA Software Institute
{javier.gomez,pedro.moreno}@imdea.org

[2] University of California, Berkeley
erkan.tairi@berkeley.edu

[3] MPI-SP
clara.schneidewind@mpi-sp.org

**Abstract.** Blockchain-based cryptocurrencies give rise to a plenitude of advanced applications (such as cross-currency transfers or privacy-preserving payments) through *blockchain protocols* – cryptographic protocols that orchestrate the processing of financial transactions on the blockchain. To enable a modular design and to enhance reusability across different cryptocurrencies, many blockchain protocols are built upon adaptor signatures (AS), a well-studied cryptographic building block, which is natively supported by most digital signature schemes used for authorizing cryptocurrency transactions. An inherent limitation of AS-based blockchain protocols is the known impossibility to realize AS for unique signature schemes, such as BLS signatures. As a consequence, existing AS-based protocols cannot be executed on cryptocurrencies that base transaction authorization on BLS signatures (such as the Chia Network). For such cryptocurrencies, instead, new custom blockchain protocols need to be created, as recently done for the case of coin mixing (S&P'24 by Hanzlik et al.) or atomic swaps between two cryptocurrencies with BLS-based transaction authorization (ESORICS'24 by Gokay et al.). To avoid such complex and error-prone redesigns, in this work, we develop a novel notion of AS called *Two-Party Asymmetric-Input Solitary-Output Adaptor Signature (2P-AISO-AS)* that sidesteps the known impossibility result and that can be realized from both randomized (i.e., plain) and unique signature schemes. We provide an efficient BLS-based instantiation of 2P-AISO-AS and show that we can obtain performant, BLS-compatible blockchain protocols by replacing AS with 2P-AISO-AS in known AS-based protocols, including zero-knowledge contingent payments, atomic swaps, and coin mixing.

## 1 Introduction

Blockchain-based cryptocurrencies such as Bitcoin enable mutually distrusting users to agree on an evolving ledger of financial transactions without relying on a trusted third party. Through this core functionality, cryptocurrencies give rise to a larger ecosystem of applications that go beyond simple financial transfers. Examples of such applications are privacy-preserving payments, cross-currency transfers, offchain transaction aggregation, or atomic data exchange. Those applications are realized by cryptographic protocols (henceforth referred to as *blockchain protocols*), which enable cryptocurrency users to securely orchestrate the publication of their transactions with the help of cryptography.

**Adaptor-signature-based protocols.** One cryptographic concept that emerged as a key building block of blockchain protocols is *adaptor signatures* [1] (AS). AS are digital signature schemes that, in addition to standard signature generation, also support the creation of so-called *presignatures*, signatures which can be completed (we also say *adapted*) to a full signature with the knowledge of a cryptographic secret. What makes AS particularly powerful is that in the context of cryptocurrencies, AS can be used to cryptographically enforce the trustless exchange between a (financial) transaction and a cryptographic secret. Such a transaction-secret-exchange then enables complex transaction synchronization patterns as needed for

⋆ Part of the work done while the author was affiliated with ENS Paris.

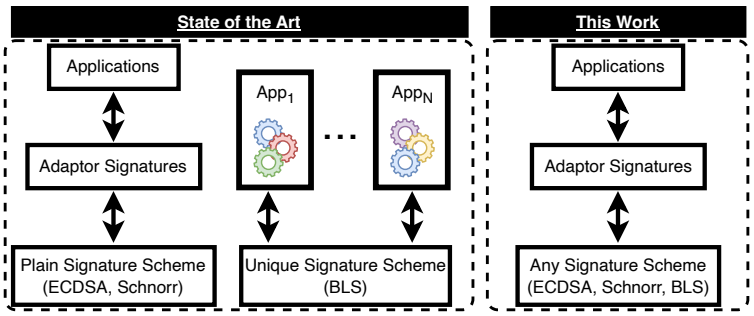



Fig. 1: **Overview of our approach**. This work (right) removes the requirement in state-of-the-art (left) of designing custom protocols to support unique signature schemes.

blockchain protocols implementing atomic swaps [2], coin-mixing [3, 4, 5, 6, 7, 2], payment channels [1, 8, 9], or atomic data exchange [10, 11, 12, 13].

All these protocols rely on AS in a black box manner and are not tied to a specific AS implementation. As a consequence, AS-based blockchain protocols are *cryptocurrency-agnostic* as long as the cryptocurrency supports transaction authorization via a signature scheme that is compatible with AS. Indeed, AS can be constructed for large classes of randomized signature schemes [14], including Schnorr and ECDSA, which are supported by cryptocurrencies like Bitcoin and Ethereum.

**The lack of adaptor signatures for BLS.** Unfortunately, there is one class of signature schemes, which, despite its practical significance, is inherently incompatible with AS, namely, unique signature schemes, with BLS as its most prominent representative. BLS signatures are particularly valued for supporting non-interactive signature aggregation, which promises on-chain space and computation savings by allowing for validating multiple authorizations at the cost of a single signature check. Individual cryptocurrencies, such as the Chia Network [15], already solely rely on BLS signatures for transaction validation and many other cryptocurrency projects started adding dedicated support for BLS verification: Ethereum has offered BLS precompiles since May'25, SUI smart contracts support BLS, and new projects (e.g., ZetaChain) are adopting BLS for transaction validation.

This development challenges the approach of building blockchain protocols upon AS since those protocols cannot be instantiated from BLS signatures. Indeed, the incompatibility between BLS and adaptor signatures has been formally proven through an impossibility result that rules out the existence of adaptor signatures for unique signatures [16]. This result implies that (i) known AS-based blockchain protocols cannot be realized upon cryptocurrencies that base their transaction authorization on BLS, such as the Chia Network [15]; (ii) AS-based blockchain protocols (such as atomic swaps) that target the interoperability between different cryptocurrencies cannot bridge between non-BLS-based and BLS-based cryptocurrencies; and (iii) existing AS-based blockchain protocols cannot benefit from possible performance improvements stemming from BLS's favorable aggregation properties.

This state of affairs poses a fundamental question: *Do BLS-based cryptocurrencies require the design of fundamentally novel blockchain protocols for known applications such as contingent payments, atomic swaps, or anonymous payments?*

Indeed, recent research developments hint at an affirmative answer to this question: For coin-mixing protocols, the efficient realizability in the presence of AS has been extensively studied [3, 4, 5, 6]. Still, to ensure compatibility with BLS signatures, [4] (S&P'24) presents a non-trivial redesign of an existing AS-based protocol, which not only results in an increase in protocol complexity but also in a substantial overhead in computation and communication costs (as opposed to known AS-based constructions). Another example is atomic swaps, where the only known BLS-based protocol [17] is custom-designed for the restricted case of swapping coins in two cryptocurrencies that rely on BLS for transaction authorization. These results are discouraging in that they suggest that (1) existing blockchain protocol designs cannot be reused; and (2) newly developed protocols will need to be customized to a concrete application setting, preventing reuse and

interoperability across different settings. E.g., the atomic swap protocol in [17] cannot be used for atomic swaps between a BLS-based and a non-BLS-based cryptocurrency.

**This work.** In this work, we show that, against the common belief, many AS-based blockchain protocols can be deployed on BLS-based cryptocurrencies (cf. Figure 1). To this end, we contribute a *novel, generalized notion of AS* that can be instantiated for all signature schemes (in particular for BLS) and still provides sufficiently strong guarantees to serve as a building block for many known blockchain protocols. Thus, our new AS notion is a versatile building block, which can facilitate the design of new efficient and truly cryptocurrency-agnostic blockchain protocols in the future.

### 1.1 Our Contributions

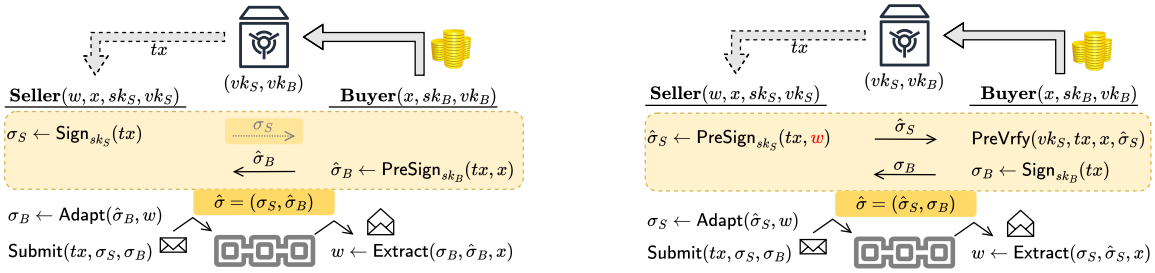
We make the following contributions:

**New AS primitive (Section 4).** We revisit the existing impossibility result from [16] and show how it can be side-stepped when performing small adjustments to the AS primitive and its security notions. The core of these adjustments relies on the insight that AS are deployed as two-party protocols where the two involved parties generate a signature w.r.t. a shared private key. Within the context of practical applications, both the *initial knowledge* as well as the *security needs* of the two parties are *asymmetric*. We reflect this asymmetry in a new primitive that we call *Two-Party Asymmetric-Input Solitary-Output Adaptor Signature (2P-AISO-AS)*.

**Primitive validation (Sections H and K).** To validate that the 2P-AISO-AS primitive indeed overcomes the impossibility result from [16], we present a generic instantiation of 2P-AISO-AS from any signature scheme (including unique signature schemes). This contribution constitutes a feasibility result, and as such is of mainly theoretical interest. Moreover, we show that existing constructions of AS from randomized signatures (e.g., Schnorr and ECDSA) are an instance of 2P-AISO-AS, ensuring that 2P-AISO-AS is compatible with standard AS constructions.

**New efficient constructions for BLS (Section 5).** We show that 2P-AISO-AS can indeed be efficiently instantiated from BLS signatures by providing two concrete and efficient constructions for BLS signatures that avoid the usage of expensive primitives, such as generic two-party computation, and that effectively benefit from BLS's support for non-interactive signature aggregation. The first construction (called BLS-BLS) supports the generation of presignatures that can be completed with the knowledge of a valid BLS signature (for a fixed public key). The second, more general construction (called BLS-DLog) enables the generation of presignatures w.r.t. secrets $w$ that are the discrete logarithm of a public statement $x$. We measure both the execution time and the communication costs incurred by these constructions, demonstrating their practicality.

**Usage in blockchain applications (Section 6).** We show how to build efficient BLS-based blockchain protocols by replacing AS with 2P-AISO-AS in existing AS-based protocols. First, we present the *first* instantiation of a zero-knowledge contingent payment based on BLS signatures. Second, we show a 2P-AISO-AS-based atomic swap protocol that (i) subsumes the protocol in [17] (ESORICS'24) for the concrete case of a swap between two BLS-based cryptocurrencies (instantiating 2P-AISO-AS with BLS-BLS); and (ii) constitutes the *first* atomic swap protocol for the more generic case of a swap between one BLS-based and one randomized-signature-based cryptocurrency (instantiating 2P-AISO-AS with BLS-DLog), thereby covering the interesting case in practice where users want to swap coins between e.g., Bitcoin and the Chia Network. Third, we show that a 2P-AISO-AS-based coin mixing protocol outperforms the custom BLS-compatible coin-mixing protocol Sweep-UC (S&P'24). Here, our approach reduces the running time from over 6 seconds to less than 300ms and the communication from 43 KB to 1832 bytes, thereby providing a speedup of more than 22× in running time and a reduction of more than 24× in communication overhead. This testifies how our framework can arrive at efficient BLS-based protocols through a principled transformation from existing AS-based protocols.




(a) **Using randomized AS**. The buyer extracts the secret $w$ from the attached signature $\sigma_B$.

(b) **Using unique AS**. The buyer extracts the secret $w$ from the attached signature $\sigma_S$.

Fig. 2: **Preimage sale protocols**. The protocols use unique and randomized AS respectively. In both cases the seller claims the payment transaction $tx$ on the ledger and the buyer extracts the secret $w$.

## 2 Key Ideas

In this section, we overview the key ideas behind 2P-AISO-AS. In particular, we explain how 2P-AISO-AS sidesteps the impossibility result from [16] and how they can be realized using a BLS-based construction.

**Adaptor signatures.** Adaptor signatures (AS) are constructed upon an existing additional digital signature scheme $\mathsf{DS} := (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrfy})$, providing a user holding signing key $\mathsf{sk}$ with an additional algorithm $\mathsf{PreSign}$ to compute *presignatures* $\hat{\sigma}$ with respect to a message $m$ and a public statement $x$ from a hard relation $\mathsf{R}$: $\hat{\sigma} \leftarrow \mathsf{PreSign}_{\mathsf{sk}}(m, x)$. Presignatures serve a two-fold purpose: (1) Any user learning a witness $w$ s.t. $(x, w) \in \mathsf{R}$ is guaranteed that they can locally *adapt* $\hat{\sigma}$ into a valid signature $\sigma$ using an algorithm $\mathsf{Adapt}$: $\sigma \leftarrow \mathsf{Adapt}(\hat{\sigma}, w)$. (2) Any user learning the adapted signature can locally *extract* $w$ from $\hat{\sigma}$ and $\sigma$ using an algorithm $\mathsf{Extract}$: $w \leftarrow \mathsf{Extract}(\hat{\sigma}, \sigma, x)$. In the latter case, successful extraction is only guaranteed for signatures $\sigma$ that were obtained by adapting $\hat{\sigma}$ using witness $w$.

**Preimage sale from randomized AS.** We illustrate the usage of adaptor signatures in the blockchain setting with the example of a simple preimage sale, a core mechanism that lies at the heart of many blockchain applications (including atomic swaps [2], coin-mixing [3, 4, 5, 6], payment channels [1], and atomic data exchange [10]). We overview such a preimage sale in Figure 2a. The goal of the preimage sale protocol is to ensure that a protocol user (the seller $S$) who knows a cryptographic secret ($w$) can exchange this secret with a transaction $tx$ corresponding to a payment from another user (the buyer $B$) to $S$. In this exchange, it should be ensured that (i) whenever $tx$ is executed, $B$ learns $w$ and (ii) $B$ cannot learn $w$ without executing $tx$. Towards this goal, $B$ first locks the payment amount into an account that is jointly controlled by $B$ and $S$, meaning that transactions spending from this account need to be authorized with respect to verification keys $\mathsf{vk}_S$ and $\mathsf{vk}_B$ of both $S$ and $B$, respectively. The buyer $B$ then can prepare a transaction $tx$ that pays $S$ from this account along with a presignature $\hat{\sigma}_B \leftarrow \mathsf{PreSign}_{\mathsf{sk}_B}(tx, x)$, which acts as a conditional authorization of $tx$ on behalf of $B$. With the knowledge of the secret $w$ satisfying $(x, w) \in \mathsf{R}$, $S$ can then complete $\hat{\sigma}_B$ to a full signature $\sigma_B$ using the $\mathsf{Adapt}$ algorithm and create their own signature $\sigma_S$ to submit $tx$ for publication on the blockchain. Observing $\sigma_B$, $B$ can use the $\mathsf{Extract}$ algorithm to obtain $w$. Intuitively, this protocol is secure since (i) $B$ can be assured that the publication of $tx$ will reveal $w$ to them and (ii) once $S$ reveals information ($\sigma_B$) that enables $B$ to infer $w$, also $tx$ will get published on the blockchain. Note that for the second point, it is crucial that the funds to be transferred were originally locked in a shared account (and are not solely controlled by $B$). Otherwise, a malicious $B$ could perform a front-running attack: upon observing the submission of $tx$ with $\sigma_B$ to the blockchain, $B$ could immediately extract $w$ and then authorize and submit an alternative transaction $tx'$ that spends the same funds as $tx$ (e.g., transferring them to another account of $B$). Due to the asynchronous blockchain execution model, even when submitted after $tx$, $tx'$ can be included into the blockchain before $tx$, and hence, causing $tx$ to fail when finally processed. If instead $tx$ transfers funds from a jointly controlled account, $S$ can prevent the submission of any such conflicting

$\mathcal{A}(x)$

$(\mathsf{vk}_{\mathcal{A}}, \mathsf{sk}_{\mathcal{A}}) \leftarrow \mathsf{KeyGen}(1^n)$; $m \leftarrow_{\$} \{0,1\}^*$

$\hat{\sigma}_{\mathcal{A}} \leftarrow \mathsf{PreSign}_{\mathsf{sk}_{\mathcal{A}}}(x, m)$; $\sigma_{\mathcal{A}} \leftarrow \mathsf{Sign}_{\mathsf{sk}_{\mathcal{A}}}(m)$

**return** $\mathsf{Extract}(\hat{\sigma}_{\mathcal{A}}, \sigma_{\mathcal{A}}, x)$

Fig. 3: **Impossibility of building AS from unique signatures.** Adversary breaking hardness of a relation if an adaptor signature scheme with underlying unique signatures existed.

transaction $tx'$. It is hence ensured that $tx$ will have the desired effect when eventually processed by the blockchain consensus.[4]

**Impossibility of unique AS.** AS, as used in the preimage sale example, can only be constructed from randomized signature schemes. In such schemes, the $\mathsf{Sign}$ algorithm uses an internal randomness, which ensures that the algorithm produces different signatures even when signing the same message $m$ twice (with the same signing key $\mathsf{sk}$). This property has been shown to be essential to construct adaptor AS schemes in the first place. The corresponding impossibility result presented in [16] builds upon the observation that any AS scheme based on unique signatures would easily allow us to build an adversary $\mathcal{A}$ that breaks the hardness of the relation $\mathsf{R}$, as illustrated in Figure 3. When provided with a statement $x$, $\mathcal{A}$ computes $w$ as follows: $\mathcal{A}$ creates a presignature $\hat{\sigma}_{\mathcal{A}}$ on some self-chosen message $m$ and self-sampled key pair $(\mathsf{vk}_{\mathcal{A}}, \mathsf{sk}_{\mathcal{A}})$, as well as a corresponding signature $\sigma_{\mathcal{A}}$. Provided that the underlying signature scheme has unique signatures, the resulting signature $\sigma_{\mathcal{A}}$ needs to coincide with the signature $\sigma'_{\mathcal{A}} \leftarrow \mathsf{Adapt}(\hat{\sigma}_{\mathcal{A}}, w)$ that could be obtained by adapting $\hat{\sigma}_{\mathcal{A}}$ using $w$. Consequently, it is ensured that $\mathsf{Extract}(\hat{\sigma}_{\mathcal{A}}, \sigma_{\mathcal{A}}, x)$ will yield the secret $w$. This impossibility suggests that AS-based applications that build upon the mechanism illustrated in the preimage sale example, cannot be realized in cryptocurrencies whose authorization mechanism relies on unique signatures.

This impossibility result teaches us that AS allows for computing the secret $w$ solely from its statement $x$ if one could, at the same time: (i) generate presignatures for arbitrary statements (without knowing the corresponding witness) and messages; (ii) generate signatures on arbitrary messages; and (iii) extract witnesses given any (presignature, signature) pair on the same message. Note that the existing AS definitions restrict the extraction in (iii) to signatures that were created *through adaptation* of the corresponding presignature. Thus, for randomized signature schemes, the impossibility result does not apply as signatures created using step (ii) use fresh randomness and do not coincide (except for negligible probability) with adapted signatures. So, signatures created using step (ii) cannot be used to extract the secret $w$ in step (iii).

Given the impossibility of using fresh randomness in unique signatures, our approach to bypass the impossibility result is to restrict the presignature generation in step (i) to scenarios where the party choosing the statement also knows the corresponding witness. In this way, the extraction in step (iii) (which requires both a signature from step (ii) and a presignature from step (i)) is only possible if the witness was already known, and hence, an unknown witness cannot be computed from the statement alone.

**Preimage sale from unique AS.** The impossibility result crucially leverages a party's ability to create a presignature $\hat{\sigma}$ from a statement $x$ without knowing the witness $w$. While the protocol from Figure 2a heavily relies on $B$ having this ability, it is, interestingly, possible to realize the preimage sale assuming that the party generating $\hat{\sigma}$ already has access to $w$. We illustrate such a protocol in Figure 2b. Here, exploiting their knowledge of $w$, $S$ prepares a presignature $\hat{\sigma}_S \leftarrow \mathsf{PreSign}_{\mathsf{sk}_S}(tx, w)$ that they send to $B$. After receiving $\hat{\sigma}_S$, $B$ can provide their signature $\sigma_B$ to $S$ that will enable $S$ to submit $tx$ to the blockchain. This protocol is secure since (i) the publication of $tx$ will reveal $w$ to $B$ as *the unique* valid signature $\sigma_S$ of $S$ will allow them

[4] When used as a building block within blockchain protocols, the presented preimage sale will be combined with a timelock mechanism that lets $B$ reclaim their funds after some time in case $S$ did not submit $tx$, avoiding that $S$ can indefinitely lock $B$'s funds.

to extract $w$ with their knowledge of $\hat{\sigma}_S$, and (ii) when revealing $\sigma_S$, $tx$ is guaranteed to be published on the blockchain (following the same argument as for the protocol in Figure 2a). Note that point (i) only holds true if the underlying signature scheme has unique signatures. Otherwise, $S$ could publish $tx$ with a fresh signature $\sigma'_S$ different from the adapted signature $\sigma_S \leftarrow \mathsf{Adapt}(\hat{\sigma}_S, w)$. From such $\sigma'_S$, $B$ could not extract the witness $w$. Also, the security of this protocol crucially relies on the order of interactions between $S$ and $B$: $B$ should only send $\sigma_B$ after having obtained $\hat{\sigma}_S$ because otherwise $S$ could publish $tx$ without providing $B$ with $\hat{\sigma}_S$, depriving them of the ability to extract $w$.

**AS as two-party protocols.** The previous example suggests the possibility of an AS notion that can be instantiated for unique signature schemes. Also, while at the first sight, the mechanics of the protocols shown in Figure 2, seem to differ fundamentally, it turns out that at a higher level of abstraction, both protocols share the same structure. More precisely (as highlighted in yellow), the initial presignature exchange in both protocols can be interpreted as a *two-party presigning protocol* run by the parties to agree on a presignature $\hat{\sigma}$ for the shared verification key $(\mathsf{vk}_S, \mathsf{vk}_B)$. At the end of this protocol, both parties hold a presignature $\hat{\sigma}$ w.r.t. to the verification key $(\mathsf{vk}_S, \mathsf{vk}_B)$ and statement $x$ where the respective presignatures take a different form depending on the protocol ($(\sigma_S, \hat{\sigma}_B)$ in Figure 2a and $(\hat{\sigma}_S, \sigma_B)$ in Figure 2b). To emphasize this correspondence, we added an initial protocol interaction in Figure 2a where $S$ sends their signature $\sigma_S$ to $B$. This addition does not impact the protocol security but ensures that $B$ at the end of the presigning protocol holds a proper presignature $\hat{\sigma}$.

Following the idea of treating presignature generation as an (interactive) protocol for computing a presignature on a joint key, in this work, we define AS with PreSign as a two-party protocol, and we call the primitive Two-Party Asymmetric-Input Solitary-Output Adaptor Signature (2P-AISO-AS), offering a flexible, unified interface for AS, which can capture a wide range of AS schemes, including known AS constructions for non-unique schemes, as well as new constructions for unique schemes.

**Asymmetric security notions.** One key observation from the protocols in Figure 2 is that their security crucially relies on the order of protocol interactions. In particular, in both protocols, $B$ needs to learn the presignature $\hat{\sigma}$ before $S$. This observation is in line with the well-known result from Cleve [18], which shows that no two-party protocol could guarantee to deliver its output to both protocol parties if one of the parties is malicious.[5] In such a scenario, only one party can be guaranteed to learn the output (we say such a party has *solitary output*). This limitation also applies to the two-party presigning protocol of 2P-AISO-AS where at most one of the parties (so $S$ or $B$) can be guaranteed to learn $\hat{\sigma}$.

To reflect the necessity of determining the protocol party who receives $\hat{\sigma}$ as solitary output, the security notions for 2P-AISO-AS differentiate between two different protocol roles: (1) the *Adaptor* who adapts the presignature $\hat{\sigma}$ (with the knowledge of $w$) (2) the *Extractor* who extracts $w$ using $\hat{\sigma}$. This distinction allows us to require in our security notions that the *Extractor* ($B$ in Figure 2a) should obtain $\hat{\sigma}$ first to ensure that if the *Adaptor* ($S$) should learn $\hat{\sigma}$ (and hence can adapt) then also the *Extractor* can extract, since they learned $\hat{\sigma}$ before.

The distinction between the *Adaptor* and the *Extractor* role lets us account for the possible asymmetries in the abilities of the different protocol roles, especially when it comes to choosing the statements $x$ for presignature creation. Our security definition assumes that the *Adaptor* can choose the statements on which the protocol parties run the presigning protocol. In particular, this entails that the *Adaptor* knows the witness $w$ and must use it as private input to the presigning protocol, hence resulting in *asymmetric input* of the protocol parties. Conversely, we reflect in our security definition that a malicious *Extractor* cannot choose the statement $x$ but only learns presignatures on *Adaptor*-chosen statements (through the presigning protocol). This distinction enables us to circumvent the impossibility result from [16]: By preventing the *Extractor* from generating presignatures on self-chosen statements, and ensuring that the *Adaptor* can only generate presignatures on statements $x$ for which they already know the witness $w$, we can provide a meaningful

[5] Choudhuri et al. [19] show how to guarantee output delivery in MPC against dishonest majority by using a blockchain. However, besides the additional overhead (i.e., each execution of the presignature protocol requires a blockchain transaction, an overhead that our approach fully removes), their construction requires a general-purpose witness encryption scheme, which is highly inefficient [20].

security notion that does not contradict the hardness of the relation $\mathsf{R}$ when instantiated with unique signatures. Intuitively, this holds since any party that can generate a presignature $\hat{\sigma}$ on a self-chosen statement $x$ is guaranteed to already know the witness $w$ and hence cannot gain additional knowledge through the attack described in Figure 3. To sanity check our 2P-AISO-AS notion, we can show that the impossibility result from [16] still applies to 2P-AISO-AS if we would remove the requirement that the *Adaptor* must have $w$ as input for the presign protocol.

**2P-AISO-AS constructions.** For showing that the definition of 2P-AISO-AS effectively overcomes the impossibility result, we first provide a general construction, which shows that it is theoretical feasible to build 2P-AISO-AS from any signature scheme. Since instances of this theoretical construction do not provide any performance guarantees in practice, we additionally present two concrete and efficient BLS-based 2P-AISO-AS constructions (for different relevant relations $\mathsf{R}$). These constructions leverage the aggregation properties of BLS to combine the different signature shares and achieve favorable performance on commodity hardware.

**Building applications from 2P-AISO-AS.** We demonstrate that 2P-AISO-AS constitutes a versatile building block for the construction of blockchain protocols. To this end, we provide a general template for 2P-AISO-AS-based protocols that generalizes the preimage sale protocol to exchange the application-dependent information. We instantiate this template for three different applications, namely zero-knowledge contingent payments, atomic swaps, and coin-mixing services, hence showing the versatility of 2P-AISO-AS.

## 3 Preliminaries

**Notation.** We denote the security parameter by $n \in \mathbb{N}$, by which each cryptographic scheme and adversary is parameterized. We denote by $\mathsf{negl}(n)$ a *negligible* function. We denote by $x \leftarrow_{\$} \mathcal{X}$ the uniform sampling of the variable $x$ from the set $\mathcal{X}$. We write $x \leftarrow \mathcal{A}(y)$ to denote that a probabilistic polynomial time ($\mathsf{PPT}$) algorithm $\mathcal{A}$ on input $y$ outputs $x$. When we want to make the randomness $r$ used by $\mathcal{A}$ explicit, we write as $x \leftarrow \mathcal{A}(y; r)$. If $\mathcal{A}$ is a deterministic polynomial time ($\mathsf{DPT}$) algorithm, we use the notation $x := \mathcal{A}(y)$. If $\mathcal{A}$ has oracle access to a procedure $\mathcal{O}$, we represent this by superscript $\mathcal{A}^{\mathcal{O}}$. We use the notation $\sigma := (\sigma_1, \sigma_2)$ for parsing a tuple $\sigma$ composed of two elements $\sigma_1$ and $\sigma_2$. We use the dot notation to access the elements of a tuple (e.g., we denote by $\sigma.\sigma_1$ the element $\sigma_1$ of $\sigma$). We denote by $[0, \ldots, n]$ the range of numbers from 0 to $n$. We denote by $|\sigma|$ the number of bits of an element $\sigma$.

**Two-party digital signatures.** A two-party signature scheme with aggregatable public keys $\mathsf{DS}_2$ is a tuple of algorithms $\mathsf{DS}_2 := (\mathsf{Setup}, \mathsf{KeyGen}, \Pi_{\mathsf{Sign}}, \mathsf{KAg}, \mathsf{Vrfy})$, where $\mathsf{pp} \leftarrow \mathsf{Setup}(n)$ is the setup algorithm; $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ is the key generation algorithm; $\sigma \leftarrow \Pi_{\mathsf{Sign}\langle \mathsf{sk}_i, \mathsf{sk}_{1-i} \rangle}(\mathsf{vk}_0, \mathsf{vk}_1, m)$ is the signature protocol; $\mathsf{avk} \leftarrow \mathsf{KAg}(\mathsf{vk}_0, \mathsf{vk}_1)$ is the key aggregation algorithm; and $\{0, 1\} \leftarrow \mathsf{Vrfy}(\mathsf{avk}, m, \sigma)$ is the verification algorithm. We require that a two-party signature scheme with aggregatable public keys satisfies *two-party existential unforgeability*.

**Hard relations.** Let $\mathsf{R}$ be a relation with the associated language $L_{\mathsf{R}} := \{x \mid \exists w \text{ s.t. } (x, w) \in \mathsf{R}\}$. We say $\mathsf{R}$ is *hard* if it is efficiently decidable, there exists an efficient instance sampling function $\mathsf{GenR}$, and it is one-way (i.e., it is computationally infeasible for an efficient adversary to output a witness $w$ for a given statement $x$ such that $(x, w) \in \mathsf{R}$). We say that a relation is $q$-hard if there is an adversary that, given $q$ statements, can find a witness for one of them.

**Public-key encryption scheme.** A public key encryption scheme $\mathsf{PKE}$ is a tuple of algorithms $\mathsf{PKE} := (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, where $(\mathsf{dk}, \mathsf{ek}) \leftarrow \mathsf{KeyGen}(1^n)$ is the key generation algorithm; $ct \leftarrow \mathsf{Enc}(\mathsf{ek}, m)$ is the encryption algorithm on input a public key $\mathsf{ek}$ and a message $m$; and $\{m, \perp\} \leftarrow \mathsf{Dec}(\mathsf{dk}, ct)$ is the decryption algorithm. A public key encryption scheme should satisfy IND-CPA.

**NIZK.** A non-interactive zero-knowledge argument system $\mathsf{NIZK}$ consists of three algorithms $\mathsf{NIZK} := (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$, where $\mathsf{Setup}$ is the public parameter (i.e., common reference string) generation algorithm; $\pi \leftarrow \mathsf{Prove}(crs, x, w)$ is the prover algorithm for a statement $x$ and a witness $w$; and $\{0, 1\} \leftarrow \mathsf{Verify}(crs, x, \pi)$ is the verification algorithm. A NIZK argument system allows a prover to convince a verifier,

using a proof $\pi$, about the existence of a witness $w$ for a statement $x$ without revealing any information apart from the fact that it knows the witness $w$. We require the NIZK argument system to satisfy the usual properties of *completeness*, *computational soundness* and *computational zero-knowledge*.

# 4 Our Definition For Adaptor Signatures

To put our definitions into context, we first provide an overview of existing definitions for adaptor signatures and motivate the need for new definitions. Then, we present the details for *Two-Party Asymmetric-Input Solitary-Output Adaptor Signature* (2P-AISO-AS).

## 4.1 Landscape of Existing Definitions

We next provide an overview of existing definitions for AS and the ones proposed in this work, and defer further related work to Section B. Figure 4 overviews the different settings in which AS are studied (single-party vs. two-party) and how the different security notions of AS evolved and are connected to the results in this work.

**Previous works: 1P-AS setting.** The foundational definitions for adaptor signatures were introduced in [1]. They considered the one-party (1P-AS) setting, where *presign is an algorithm that takes as secret input a single signing key*. In this setting, [1] defines two security properties for the holder of the signing key: existential *unforgeability* (aEUF-CMA) and *witness extractability* (aWitExt). The former models the unforgeability of signatures by an adversary. The latter establishes that if an adversary comes up with a forgery on a message $m$ and statement $x$, then the honest signing key holder must be able to extract the corresponding witness $w$.

The security notions of aEUF-CMA and aWitExt in [1] are limited to the case where the adversary can only observe a single presignature on the message of the forgery. Such a limitation has been first identified in [21] and further studied in [14], where it is shown that this limited security notion does not suffice to model the security of AS-based blockchain applications in practice. To address this limitation, the work in [21] defines *full extractability* (FExt), a definition ensuring security for the signing key holder.[6] Intuitively, this notion — where the adversary has access to two interfaces: (i) a signature oracle for adversarially chosen messages; and (ii) a presignature oracle for adversarially chosen pairs of message and statement — models the property that if the adversary gives a forgery, then the signing key holder can extract the corresponding witness from a presignature query previously asked by the adversary. In fact, [21] shows that FExt strictly implies both aEUF-CMA and aWitExt.

Note that the security notions in the 1P-AS setting are defined from the perspective of a single (honest) signing key holder, which intuitively plays the role of the *Extractor*, being confronted with a malicious *Adaptor*. For building a secure protocol (e.g., a preimage sale), 1P-AS needs to be combined with a (standard) signature from the *Adaptor* in a 2-of-2 multisignature to ensure that the protocol funds are jointly controlled and cannot be stolen by a malicious *Extractor*. This extractor-centric view fixes how AS can be used and excludes protocol flows as the one illustrated in Figure 2b where the *Adaptor* (instead of the *Extractor*) creates the presignature. The impossibility result for unique adaptor signatures was also established in the 1P-AS setting, explaining why the assumption that presignature generation should not require knowledge of the witness $w$ had not been questioned: If the *Extractor* is the party assumed to both create the presignature and to later learn $w$, then it does not make sense to assume the *Extractor* to know $w$ already at presignature generation time.

**Previous works: 2P-AS setting.** In the two-party setting introduced in [16] (2P-AS), the presignature generation is modeled as a two-party protocol, where both parties input their own (share of the) signing key and *they both get the output of the presignature generation*. This notion generalizes the deployment setting of

[6] The work in [21] also considers the notion of strong full extractability, a notion that resembles strong unforgeability in plain signatures. However, we restrict our discussion here to regular full extractability for brevity.

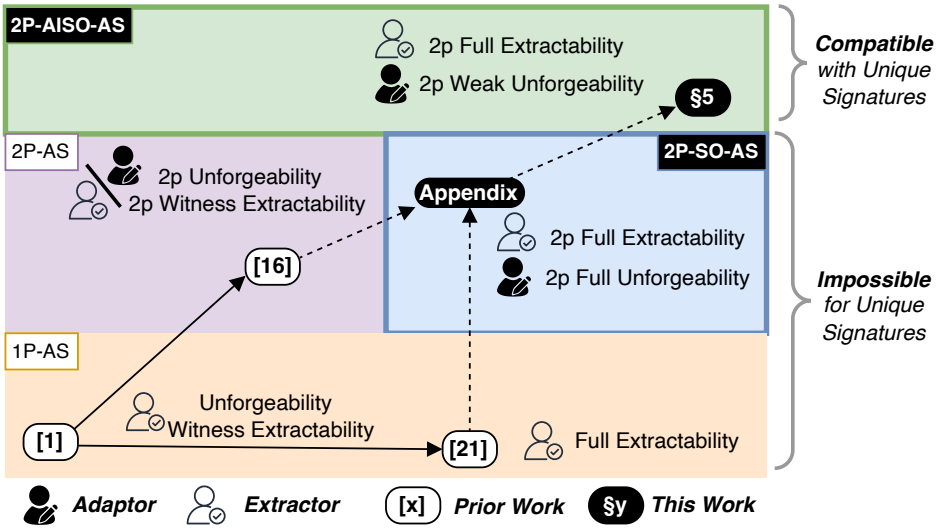



Fig. 4: Landscape of security definitions for adaptor signatures.

1P-AS in that it can capture scenarios where a joint account is controlled through a 2-of-2 multisignature, but also those where it is controlled through a shared key. The security definitions for this setting, also introduced in [16], are fully symmetric in that they do not distinguish between the *Adaptor* and *Extractor* roles. They directly mimic the definitions from the 1P-AS setting in a scenario where presignature generation is a two-party protocol that provides the presignature as output to both parties. More concretely, they define two security notions: two-party existential unforgeability (2P-aEUF-CMA) and two-party witness extractability (2P-aWitExt). The 2P-AS setting has three fundamental limitations. (1) the notions of 2P-aEUF-CMA and 2P-aWitExt share the same limitations as the corresponding ones in the 1P-AS setting: they only allow a single presignature on the message of the forgery, undermining their applicability in practice, as described above; (2) the 2P-AS setting models the two-party presignature as a protocol with guaranteed output delivery for both parties, a guarantee that is impossible to be achieved unless both parties are honest [18]; (3) as we show in Section D, it remains impossible to build 2P-AS from unique signatures (e.g., BLS).

**Our Work: 2P-SO-AS setting.** To overcome limitations (1) and (2) of the 2P-AS setting, we propose new security definitions, where each party has a clear role in the presignature generation: one party is the *Extractor*, and the other is the *Adaptor*. To enable the realizability of the presigning protocol in the dishonest majority setting (addressing limitation (2)), we model presignature generation as a protocol where only the *Extractor* gets the output of the presignature generation, as required in practice (c.f. Section 2), so the presigning protocol has *solitary output (SO)*. Finally, to overcome limitation (1), we strengthen the security properties in line with findings from the 1P-AS setting [21]: Here, we also reflect the protocol roles by providing role-specific security notions that take the asymmetric security needs of the protocol parties into account. For the *Extractor*, we adapt full extractability from the 1P-AS setting to the two-party setting (2P-FExt). For the *Adaptor*, who does not need guarantees in terms of extractability, we instead require a strong unforgeability notion (called full unforgeability, or 2P-FUnf), which ensures that the *Extractor* alone cannot forge signatures for the joint key. We defer the details for such *Two-Party Solitary-Output Adaptor Signature* (2P-SO-AS) to Section J.

Note that signatures in the 2P-SO-AS setting are not realizable for unique signature schemes (we show a corresponding impossibility result in Section D), but can be considered a relevant intermediate result that transfers recent advancements in defining meaningful security notions for 1P-AS to the 2P-AS setting in a role-aware manner.

**Our Work: 2P-AISO-AS setting.** Finally, to overcome limitation (3), we refine the 2P-SO-AS setting along two axes: First, we slightly adapt the interface to the presigning algorithm, requiring the *Adaptor* to provide the witness as input. Second, we weaken the full unforgeability guarantee of the *Adaptor* reflecting that unforgeability only needs to hold against a malicious *Extractor* $E$ who is restricted to learn presignatures for *Adaptor*-chosen statements (a requirement needed to overcome the impossibility result, as motivated in Section 2). Providing a concise characterization of this restriction is challenging because we need to make sure that while a malicious *Extractor* should not be able to choose the statements on which to create presignatures, *Extractor* should still have access to adapted signatures to ensure that such signatures do not leak any information that may help *Extractor* in creating forgeries. Since in this new setting the witness is an input to the protocol $\Pi\mathsf{PreSign}$, we require a new notion named *one-way presignatures*. This new notion ensures that no information about the witness is leaked during the presignature generation. We will provide the details for the security notions of the 2P-AISO-AS setting in the following.

### 4.2 2P-AISO-AS Definition

The 2P-AISO-AS scheme is in the following defined to contain a simple one-round presigning protocol consisting of one message from the *Adaptor* to the *Extractor*. For the constructions shown in this paper, such a single-round notion is sufficient. For completeness, we discuss a more generic definition of a multi-round presign protocol in Section I.

**Definition 1 (2P-AISO-AS Scheme).** *A Two-Party Asymmetric-Input Solitary-Output Adaptor Signature (2P-AISO-AS) scheme is defined w.r.t. a relation* $\mathsf{R} \subseteq (\mathcal{X}, \mathcal{W})$*, and a two-party signature scheme* $\mathsf{DS}_2 = (\mathsf{Setup}, \mathsf{KeyGen}, \Pi\mathsf{Sign}, \mathsf{Vrfy}, \mathsf{KAgg})$[7] *with message space* $\mathcal{M}$ *and signature space* $\mathcal{S}$*. It is run between parties Adaptor (A) and Extractor (E), and consists of a tuple* $\mathsf{AS}_2^{\mathsf{R},\mathsf{DS}_2} = (\mathsf{PreSign}_A, \mathsf{PreSign}_E, \mathsf{Adapt}, \mathsf{PreVrfy}, \mathsf{Extract})$ *of efficient algorithms defined as:*

– $(pm, st) \leftarrow \mathsf{PreSign}_A(\mathsf{sk}_A, m, w, x)$*: is a* PPT *algorithm that on input the signing key of the Adaptor* $\mathsf{sk}_A$*, a message* $m \in \mathcal{M}$*, a witness* $w$*, and the corresponding statement* $x$ *such that* $(x, w) \in \mathsf{R}$*, it outputs a protocol message* $pm$ *and a protocol state* $st$*.*

– $\hat{\sigma} \leftarrow \mathsf{PreSign}_E(\mathsf{sk}_E, m, x, pm)$*: is a* PPT *algorithm that on input the signing key of the Extractor* $\mathsf{sk}_E$*, a message* $m \in \mathcal{M}$*, a statement* $x \in \mathcal{X}$ *and a protocol message* $pm$*, it outputs a presignature* $\hat{\sigma}$*.*

– $b := \mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma})$*: is a* DPT *algorithm that on input a verification key* $\mathsf{vk}$*, a message* $m \in \mathcal{M}$*, a statement* $x \in \mathcal{X}$ *and a presignature* $\hat{\sigma}$*, outputs a bit* $b \in \{0, 1\}$*.*

– $\sigma := \mathsf{Adapt}(\mathsf{vk}, \hat{\sigma}, w, st)$*: is a* DPT *algorithm that on input a verification key* $\mathsf{vk}$*, a presignature* $\hat{\sigma}$*, a witness* $w \in \mathcal{W}$ *and a protocol state* $st$*, outputs a signature* $\sigma \in \mathcal{S}$*.*

– $w := \mathsf{Extract}(\mathsf{vk}, \sigma, \hat{\sigma}, x)$*: is a* DPT *algorithm that on input a verification key* $\mathsf{vk}$*, a signature* $\sigma \in \mathcal{S}$*, a presignature* $\hat{\sigma}$ *and a statement* $x \in \mathcal{X}$*, outputs a witness* $w$ *s.t.* $(x, w) \in \mathsf{R}$*, or an error symbol* $\perp$*.*

For ease of presentation, we define the presignature generation as a two-message protocol, where first the *Adaptor* sends a message to the *Extractor*, and then the *Extractor* outputs the presignature. Nevertheless, this can be easily extended to a multi-round presignature protocol (c.f. Section I).

Next, we present the correctness of 2P-AISO-AS, which captures the fact that if both parties are honest, then the presignature generated through the presignature generation protocol can be correctly preverified and adapted to a valid signature, and the witness can be extracted from the adapted signature and the presignature.

Intuitively, a 2P-AISO-AS scheme is correct if honestly created presignatures and adapted signatures (pre-) verify and enable the extraction of a valid witness:

**Definition 2 (Two-Party Presignature Correctness).** *A 2P-AISO-AS scheme* $\mathsf{AS}_2^{\mathsf{R},\mathsf{DS}_2}$ *satisfies* two-party presignature correctness *if for every* $n \in \mathbb{N}$*, and every message* $m \in \mathcal{M}$*, the following holds:*

[7] For simplicity, our syntax and security notions consider a centralized $\mathsf{KeyGen}$ that does not fail (following similar modeling from threshold signatures [22]). This is just a syntactic simplification and the key generation could be a distributed key generation (DKG).

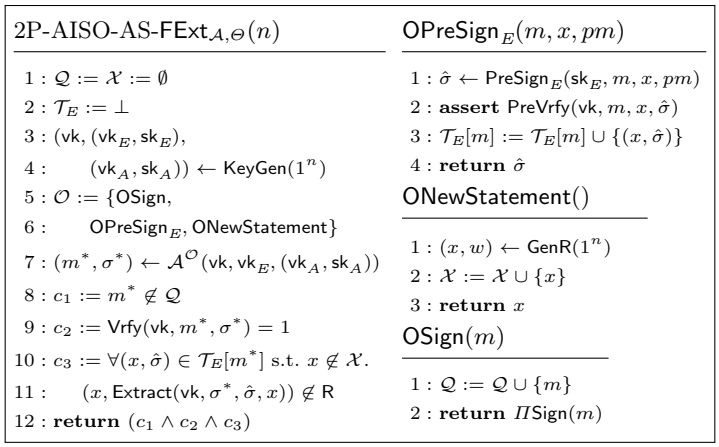



Fig. 5: 2P-AISO-AS full extractability game where the adversary corrupts the adaptor party $A$. We denote by $\Pi\mathsf{Sign}$ the two-party signature protocol.

$$\Pr\left[\begin{array}{r|r} \mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma}) \wedge & (\mathsf{vk}, (\mathsf{vk}_A, \mathsf{sk}_A), \\ \mathsf{Vrfy}(\mathsf{vk}, m, \sigma) \wedge & (\mathsf{vk}_E, \mathsf{sk}_E)) \leftarrow \mathsf{KeyGen}(1^n) \\ (x, w') \in \mathsf{R} & (x, w) \leftarrow \mathsf{GenR}(1^n) \\ & (pm, st) \leftarrow \mathsf{PreSign}_A(\mathsf{sk}_A, m, w, x) \\ & \hat{\sigma} \leftarrow \mathsf{PreSign}_E(\mathsf{sk}_E, m, x, pm) \\ & \sigma := \mathsf{Adapt}(\mathsf{vk}, \hat{\sigma}, w, st) \\ & w' := \mathsf{Extract}(\mathsf{vk}, \sigma, \hat{\sigma}, x) \end{array}\right] = 1.$$

Next, we define the security notions of interest for 2P-AISO-AS, motivated earlier in this section. First, we define full extractability that ensures to the honest *Extractor* that if an adversarial *Adaptor* produces a forgery after observing presignatures, then the *Extractor* can extract the corresponding witness from the forgery and one of the presignatures.

**Definition 3 (Full Extractability).** *A 2P-AISO-AS scheme* $\Theta := \mathsf{AS}_2^{\mathsf{R},\mathsf{DS}_2}$ *is* fully extractable *if for every* PPT *adversary* $\mathcal{A}$ *there exists a negligible function* negl *such that* $\Pr[\textit{2P-AISO-AS-}\mathsf{FExt}_{\mathcal{A},\Theta}(n) = 1] \leq \mathsf{negl}(n)$*, where the experiment 2P-AISO-AS-*$\mathsf{FExt}_{\mathcal{A},\Theta}$ *is defined in Figure 5.*

To fit the 2P-AISO-AS setting, in the *2P-AISO-AS-*FExt game a malicious $A$ (when interacting with an honest $E$) can learn presignatures on possibly adversarially chosen messages and statements. The *2P-AISO-AS-*FExt game allows this by not restricting the message $m$ and statement $x$ used as input to the PreSign oracle. Additionally, the adversary can also learn a presignature on an honestly generated statement (as encoded in the ONewStatement oracle).

The conditions of the game model that to win the adversary should produce a forgery $(m^*, \sigma^*)$, that is valid (condition $c_2$) and not be directly obtained from the signing oracle (condition $c_1$). Moreover, since an adversarial $A$ can get presignatures (through the OPreSign oracle) on a statement $x$ of their own choice (i.e., $x \notin \mathcal{X}$), the adversary can locally adapt such a presignature into a valid signature and claim it as a forgery. In such a case, the adversary wins if the honest $E$ cannot extract a witness from the forgery and the presignature (condition $c_3$).

Next, we define weak unforgeability that ensures to the honest *Adaptor* that no adversarial *Extractor* can forge signatures on their behalf, even after observing signatures and presignatures on honestly generated statements.

**Definition 4 (Weak Unforgeability).** *A 2P-AISO-AS scheme* $\Theta := \mathsf{AS}_2^{\mathsf{R},\mathsf{DS}_2}$ *is* weakly unforgeable *if for every* PPT *adversary* $\mathcal{A}$ *there exists a negligible function* negl *such that* $\Pr[\textit{2P-AISO-AS-}\mathsf{WUnf}_{\mathcal{A},\Theta}(n) = 1] \leq \mathsf{negl}(n)$*, where the experiment 2P-AISO-AS-*$\mathsf{WUnf}_{\mathcal{A},\Theta}$ *is defined in Figure 6.*

To build intuition for the *2P-AISO-AS*-WUnf game, consider it first without the lines highlighted in gray. To fit the 2P-AISO-AS setting, a malicious *Extractor* only learns presignatures on input statements chosen by the *Adaptor*. This is enforced by asserting that any query to $\mathsf{OPreSign}_A$ must have as input a statement whose corresponding witness has been previously generated through the oracle ONewStatement, and thus, is known to the challenger (who plays the role of the *Adaptor* in this game).

Through the OAdapt oracle, the adversarial *Extractor* learns signatures resulting from adapting honestly created presignatures (assert in line 2) on honestly generated statements (assert in line 1). Having the oracle OAdapt is crucial as otherwise, the adversary would never get access to adapted signatures — an assumption that is unrealistic in practice where it is part of the intended protocol flow that a (potentially malicious) *Extractor* eventually learns the adapted signature from the blockchain to extract $w$ from it. To showcase the necessity of the OAdapt oracle, in Section E, we provide a concrete example of an (intuitively) insecure construction, which leaks the honest *Adaptor*'s signing key to the *Extractor* upon signature adaptation, and that we could prove secure for a security definition that omits the OAdapt oracle.

Finally, similar to unforgeability notions for standard digital signatures, the adversary should not be able to claim a pair $(m^*, \sigma^*)$ as a valid forgery when the adversary obtains $\sigma^*$ through the signing oracle (i.e., $m^* \notin \mathcal{Q}$) or through the adapt oracle (i.e., $m^* \notin \mathcal{M}$).

Now we shift our attention to the parts highlighted in gray. The adversary has the ability to learn signatures leaked through the faithful execution of the adaptor signature scheme. This is intended behaviour and the adversary should not be able to claim these signatures as forgeries. This is what the highlighted lines in gray model. To give a concrete example, a query to OAdapt on input a statement $x$ and a message $m$ allows the adversary to faithfully extract the corresponding witness $w$ through Extract and thus convert any presignature on any message w.r.t. $x$ into a valid signature on their own. Therefore, $x$ is added to the set $\mathcal{X}$ of leaked statements and not permitted to be used in any of the queries to $\mathsf{OPreSign}_A$ for the forgery message $m^*$ (i.e., $\mathcal{X} \cap \mathcal{T}_A[m^*] = \emptyset$ in $c_1$). Moreover, any message $m$ for which the adversary has previously obtained a presignature on $x$ through the oracle $\mathsf{OPreSign}_A$ can be now faithfully adapted into a signature on their own using $w$, and thus, $m$ is added to the set $\mathcal{Q}$, removing the possibility from the adversary to claim it as a forgery. Finally, when the underlying signature scheme is unique, an adversary can extract witnesses from presignatures and signatures obtained through OSign.

The rest of highlighted lines handle similar bookkeeping of the information leaked to the adversary through the intended functionality of the adaptor signature scheme, as this information should not be used by the adversary to claim forgeries.

Next, we cast the rest of properties previously defined in the 1P-AS setting to the 2P-AISO-AS setting.

**Definition 5 (Adaptability and designated adaptability).** *For any message* $m \in \mathcal{M}$*, any verification key* vk*, any statement/witness pair* $(x, w) \in \mathsf{R}$*, and any presignature* $\hat{\sigma}$ *with* $\mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma}) = 1$*, we say that a 2P-AISO-AS scheme satisfies* designated adaptability *if there exists a protocol state st for which it holds that* $\mathsf{Vrfy}(\mathsf{vk}, m, \mathsf{Adapt}(\mathsf{vk}, \hat{\sigma}, w, st)) = 1$*. Alternatively, we say that a 2P-AISO-AS scheme satisfies* adaptability *if for every protocol state st it holds that* $\mathsf{Vrfy}(\mathsf{vk}, m, \mathsf{Adapt}(\mathsf{vk}, \hat{\sigma}, w, st)) = 1$*.*

The *designated adaptability* and *adaptability* notions ensure that valid presignatures $\hat{\sigma}$ can be adapted, that is, adapting a presignature (that preverifies) with a correct witness $w$ yields a signature that verifies. In particular, *designated adaptability* ensures that the *Adaptor* that participated in the generation of $\hat{\sigma}$ can adapt with a correct witness, while *adaptability* ensures that anyone can adapt with a correct witness. The reason behind having two adaptability notions is that the 2P-AISO-AS setting allows for a more flexible design of the presigning protocol: A 2P-AISO-AS construction may or may not allow anyone to adapt a presignature. In Section 5 we present two 2P-AISO-AS constructions, one for each case.

**Definition 6 (Unlinkability).** *A 2P-AISO-AS scheme* $\Theta := \mathsf{AS}_2^{\mathsf{R},\mathsf{DS}_2}$ *is* unlinkable *if for every* PPT *adversary* $\mathcal{A}$ *there exists a negligible function* negl *such that:*

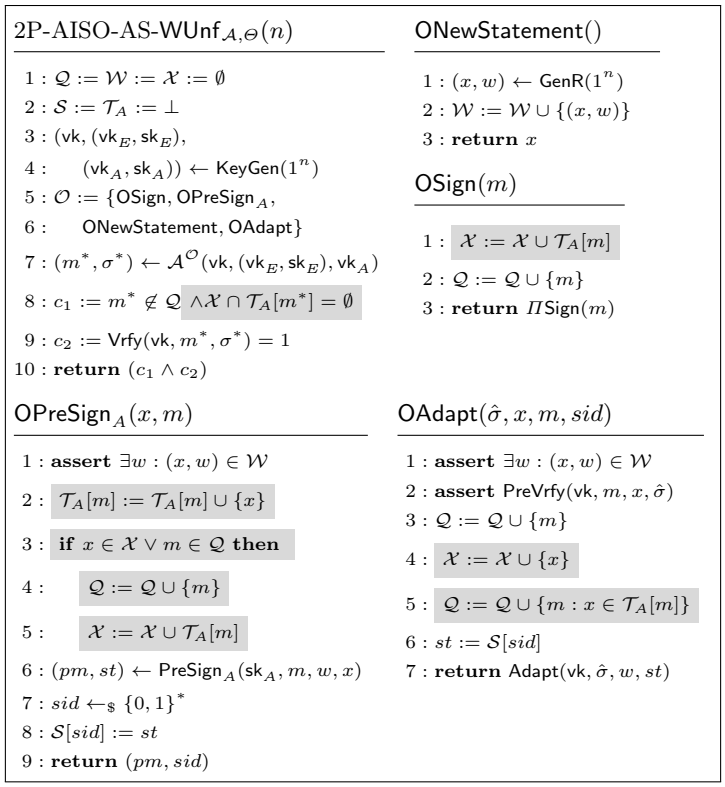



Fig. 6: 2P-AISO-AS weak unforgeability games where the adversary corrupts the party *Extractor E*. We denote by $\Pi\mathsf{Sign}$ the two-party signature protocol.

$$|\Pr[\mathsf{Unlinkability}_{\mathcal{A},\Theta}(n,0)] - \Pr[\mathsf{Unlinkability}_{\mathcal{A},\Theta}(n,1)]| \leq \mathsf{negl}(n)$$

*where* $\mathsf{Unlinkability}_{\mathcal{A},\Theta}$ *is defined in Figure 7.*

The *unlinkability* notion intuitively captures that a signature adapted from a presignature cannot be distinguished from a fresh signature computed in a signing protocol. This notion is defined w.r.t. an adversarial eavesdropper that is neither the *Adaptor* nor the *Extractor*. This adversary is the one trying to distinguish between signatures that were computed in a signing protocol, and signatures that were adapted. If the eavesdropper was either the *Adaptor* or the *Extractor*, they would trivially distinguish whether a signature was adapted since they participated in the two-party presigning protocol.

**Definition 7 (Computational preverify soundness).** *A 2P-AISO-AS scheme satisfies* computational preverify soundness *if for every* PPT *adversary* $\mathcal{A}$ *and for every* $n \in \mathbb{N}$ *and polynomially bounded* $x \notin L_{\mathsf{R}}$*, it holds that:*

$$\Pr\left[b = 1 \;\middle|\; \begin{array}{c}(\mathsf{vk}, (\mathsf{vk}_E, \mathsf{sk}_E), (\mathsf{vk}_A, \mathsf{sk}_A)) \leftarrow \mathsf{KeyGen}(1^n) \\ (m, \hat{\sigma}) \leftarrow \mathcal{A}(\mathsf{vk}, (\mathsf{vk}_E, \mathsf{sk}_E), (\mathsf{vk}_A, \mathsf{sk}_A)) \\ b := \mathsf{PreVrfy}(\mathsf{vk}, m, \hat{\sigma}, x)\end{array}\right] \leq \mathsf{negl}(n),$$

The *computational preverify soundness* notion models the impossibility for an adversary to use a statement $x$ that is not in the language of the relation $L_{\mathsf{R}}$ in order to compute a verifying presignature. This definition prevents the usage of ill-formed statements to compute presignatures that, although verifying, would not extract a valid witness.

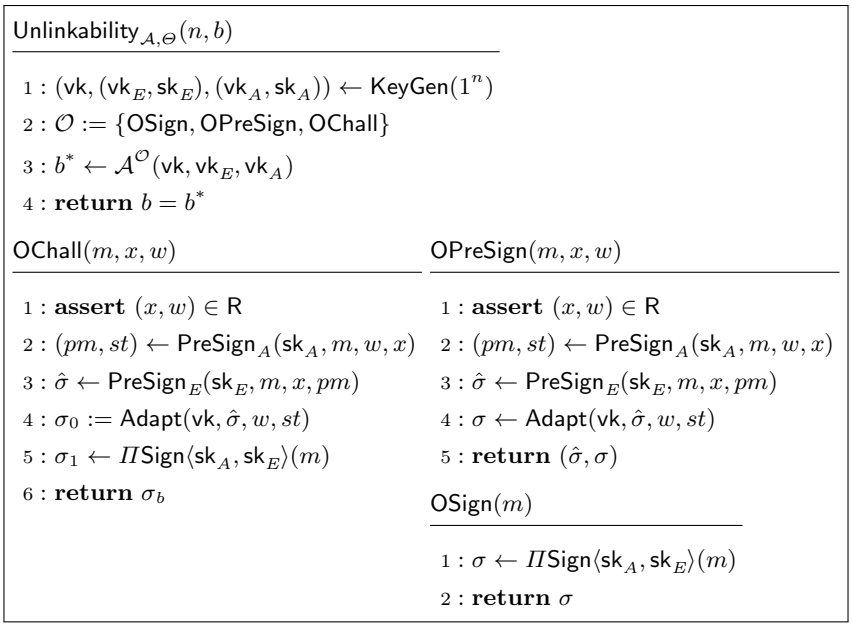



Fig. 7: 2P-AISO-AS unlinkability game. We denote by $\Pi\mathsf{Sign}$ the execution of the two-party signature protocol.

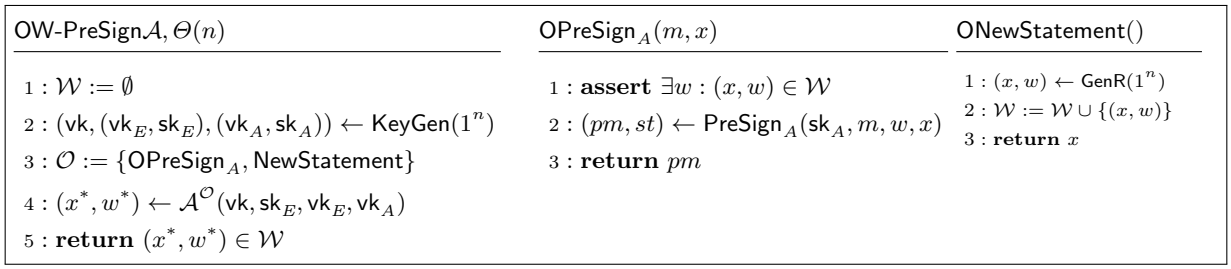



Fig. 8: 2P-AISO-AS one-way presignature game.

**Definition 8 (One-way presignature).** *A 2P-AISO-AS scheme* $\Theta := \mathsf{AS}_2^{\mathsf{R},\mathsf{DS}_2}$ *has* one-way presignatures *if for every* PPT *adversary* $\mathcal{A}$ *there exists a negligible function* negl *such that:*

$$|\Pr[\mathsf{OW\text{-}PreSign}\mathcal{A},\Theta(n)=1]| \leq \mathsf{negl}(n)$$

*where* $\mathsf{OW\text{-}PreSign}\mathcal{A},\Theta$ *is defined in Figure 8.*

The *one-way presignatures* notion ensures that the *Adaptor* does not leak the witness in the presignature generation protocol. Notice that this notion only makes sense against a malicious *Extractor* that tries to extract information about the witness from presignatures alone. This notion is implied by *adaptability* and *weak unforgeability* (Section L), but becomes relevant when a construction does not achieve *adaptability* but *designated adaptability* instead.

**Discussion.** In line with prior work, we define the security of 2P-AISO-AS using cryptographic games. In principle, it could be possible to characterize the security of 2P-AISO-AS also using simulation-based definitions in the Universal Composability (UC) framework. We opted against this approach for multiple reasons. First, defining security in the UC framework even for simple cryptographic schemes such as digital

signatures has been shown challenging and error-prone [23]. Second, security notions in the UC framework often result to be less comprehensive. Security is characterized through a single ideal functionality that encapsulates all security properties at once. In contrast, our game-based approach decouples these notions, preserving properties such as unlinkability and pre-verify soundness as in prior work, while refining weak unforgeability for the needs of 2P-AISO-AS. This, in particular, allows for establishing a clear connection with well-studied notions in prior works (such as [1, 16, 21, 14]), which also adopt game-based definitions. Further, the impossibility result from [16] does not seamlessly extend to the UC framework. Our approach, instead, enables a direct comparison between this result and our notion of 2P-AISO-AS, as detailed in Section D.

## 5 2P-AISO-AS Constructions

In this section, we showcase the versatility of the 2P-AISO-AS scheme by giving different constructions. They differ in the supported underlying digital signature scheme and hard relation. Later, Section 6 describes how these 2P-AISO-AS constructions enable AS-based applications for BLS-based blockchains. Additionally, in Section K we recast the 2P-AS construction of [16] to an 2P-AISO-AS scheme.

For readability, we review now BLS before delving into our constructions. BLS implements the three standard algorithms: $\mathsf{KeyGen}$, $\mathsf{Sign}$ and $\mathsf{Vrfy}$. Interestingly in practice, BLS additionally supports an aggregation algorithm $\mathsf{Agg}$ to combine multiple signatures into a single one. As an illustrative example, on input two signatures $\sigma_1 \leftarrow \mathsf{Sign}(\mathsf{sk}_1, m_1)$ and $\sigma_2 \leftarrow \mathsf{Sign}(\mathsf{sk}_2, m_2)$, the algorithm $\mathsf{Agg}(\sigma_1, \sigma_2)$ produces a single signature $\sigma^*$ that is simultaneously a valid signature on the message $m_1$ under the verification key $\mathsf{vk}_1$ and on the message $m_2$ under the verification key $\mathsf{vk}_2$. To verify that, BLS supports an aggregated verification algorithm $\mathsf{AggVerify}$ that takes as input multiple verification keys, multiple messages and a single aggregated signature, and checks that the aggregated signature is valid for each message under the corresponding verification key. In our running example, $\mathsf{AggVerify}(\{\mathsf{vk}_1, \mathsf{vk}_2\}, \{m_1, m_2\}, \sigma^*)$ would output 1.

### 5.1 BLS-BLS Construction

**Setting.** The first construction enables efficient BLS-based AS for witnesses that themselves constitute valid BLS signatures w.r.t. some (public) message $m_\mathsf{R}$ and verification key $\mathsf{vk}_\mathsf{R}$. More formally, witness-statement pairs come from the relation

$$\mathsf{R}_{BLS} := \left\{ \begin{array}{l} x := (\mathsf{vk}_\mathsf{R}, m_\mathsf{R}), w := (\sigma_\mathsf{R}) \ \text{ s.t.} \\ \mathsf{BLS.Vrfy}(\mathsf{vk}_\mathsf{R}, m_\mathsf{R}, \sigma_\mathsf{R}) = 1 \end{array} \right\}$$

**Construction details.** The construction is presented in Figure 9. In the presignature generation, the *Adaptor* masks their signature with the witness. For that, they use the aggregation function $\mathsf{Agg}$ of BLS to compute the output $pm$ that combines *Adaptor*'s BLS signature $\sigma_A$ on the input message $m$, and the witness $w$, which is also a BLS signature $\sigma_\mathsf{R}$ on a different (application-dependent) message $m_\mathsf{R}$. Conversely, the *Extractor* uses $\mathsf{Agg}$ to combine the *Adaptor*'s output $pm$ with its own BLS signature $\sigma_E$ on the input message $m$ to produce the presignature $\hat{\sigma}$. At this point, the presignature $\hat{\sigma}$ is an aggregation of three BLS signatures: the *Adaptor*'s signature $\sigma_A$, the *Extractor*'s signature $\sigma_E$ and the witness $w$. The rest of the algorithms of this construction are straightforward. $\mathsf{PreVrfy}$ checks that the presignature $\hat{\sigma}$ is a valid BLS aggregated signature on the messages $(m, m_\mathsf{R})$ under the verification keys $(\mathsf{vk}, \mathsf{vk}_\mathsf{R})$, where $\mathsf{vk} := \mathsf{Agg}(\mathsf{vk}_A, \mathsf{vk}_E)$ as expected from the key generation. $\mathsf{Adapt}$ removes the witness $w$ from the presignature $\hat{\sigma}$ to produce a valid BLS signature on the message $m$ under the verification key $\mathsf{vk}$. Finally, $\mathsf{Extract}$ removes the aggregated signature $\mathsf{Agg}(\sigma_A, \sigma_E)$ from the presignature $\hat{\sigma}$ to obtain back the witness $w$.

**Security analysis.** The BLS-BLS construction satisfies all security properties of 2P-AISO-AS, as stated in Theorem 1. The proof of this theorem is deferred to Section F.

**Theorem 1.** *If BLS signatures are unforgeable and unique, then the construction in Figure 9 satisfies full extractability, weak unforgeability, adaptability, unlinkability, computational preverify soundness and one-way signatures in the Random Oracle Model.*

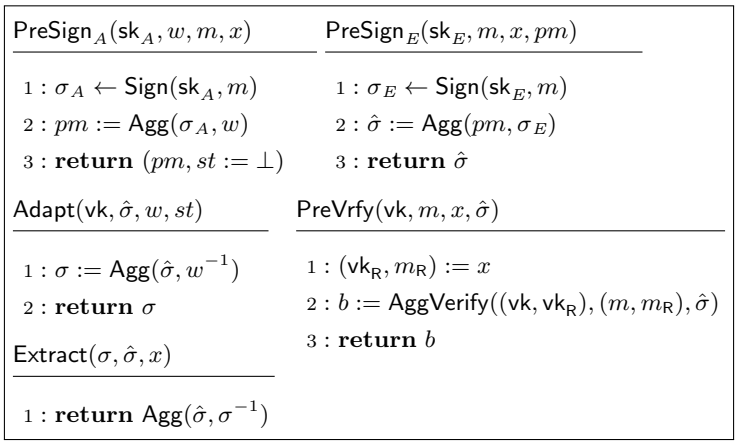



Fig. 9: 2P-AISO-AS construction for BLS signatures and relation $\mathsf{R}_{BLS}$.

### 5.2 BLS-DLog Construction

**Setting.** This second construction also uses BLS signatures as the underlying signature scheme, but it supports the discrete logarithm relation $\mathsf{R}_{DLog}$ defined as follows, where $g$ is a generator of a prime order group:

$$\mathsf{R}_{DLog} := \{\, x := (z, g), w := s \;\text{ s.t. }\; z = g^s \,\}$$

This setting is more general than that of BLS-BLS, which makes it specially interesting in applications: Any application-dependent secret $\gamma$ can be encrypted using the statement $z$ as encryption key, and then this 2P-AISO-AS construction can be used to exchange a signature for the corresponding decryption key $s$ (i.e., the witness).

**Building block: verifiable witness encryption for a relation (VWER) [12].** This construction relies on verifiable witness encryption based on signatures, a cryptographic scheme recently introduced in [12] where intuitively, elements from a signature based relation (e.g., $\mathsf{R}_{BLS}$ mentioned earlier) serve as the public/private key pair for encryption/decryption. Although general-purpose witness encryption is known to be highly inefficient [20], verifiable witness encryption for the concrete case of $\mathsf{R}_{DLog}$ can be implemented efficiently as shown in [12, 13].

A VWER can be seen as a standard encryption scheme augmented as follows. First, instead of using standard encryption/decryption keys, the encryption and decryption algorithms take as input a statement-witness pair for a relation: The encryption takes as input a statement $x$ and decryption is possible with the corresponding witness $w$. For instance, taking the relation $\mathsf{R}_{BLS}$ defined above, encryption is done w.r.t. a pair $(\mathsf{vk}_\mathsf{R}, m_\mathsf{R})$ and decryption would require a witness $\sigma_\mathsf{R}$ such that $\mathsf{BLS.Vrfy}(\mathsf{vk}_\mathsf{R}, m_\mathsf{R}, \sigma_\mathsf{R}) = 1$. Second, instead of encrypting/decrypting arbitrary messages, VWER is restricted to encrypt/decrypt witnesses of a relation, possibly different to the one used as encryption/decryption keys. For instance, taking the relation $\mathsf{R}_{DLog}$ defined above, VWER can be used to encrypt/decrypt a witness $w_{DLog}$. Finally, the restricted message space facilitates the support of an additional verify functionality: On input two statements $x_{BLS}$ and $x_{DLog}$, the encryption algorithm produces a ciphertext $c$ along with a proof $\pi$ of two facts about $c$: (i) $c$ encrypts a witness $w_{DLog}$ such that $(x_{DLog}, w_{DLog}) \in \mathsf{R}_{DLog}$; and (ii) $c$ can be decrypted with a witness $w_{BLS}$ such that $(x_{BLS}, w_{BLS}) \in \mathsf{R}_{BLS}$.

Next, we introduce the notation for the algorithms of VWER as used in our second construction of 2P-AISO-AS. Here, the relation used for encryption/decryption keys is $\mathsf{R}_{BLS}$ and the relation for the message space is $\mathsf{R}_{DLog}$. A detailed description of VWER is deferred to Section A.

A verifiable witness encryption for a relation scheme VWER is a tuple of algorithms $\mathsf{VWER} := (\mathsf{EncR}, \mathsf{VfEncR}, \mathsf{DecR})$, where $(c, \pi) \leftarrow \mathsf{EncR}((\mathsf{vk}, m), w)$ is the encryption algorithm; $\{0, 1\} \leftarrow \mathsf{VfEncR}((c, \pi), (\mathsf{vk},$

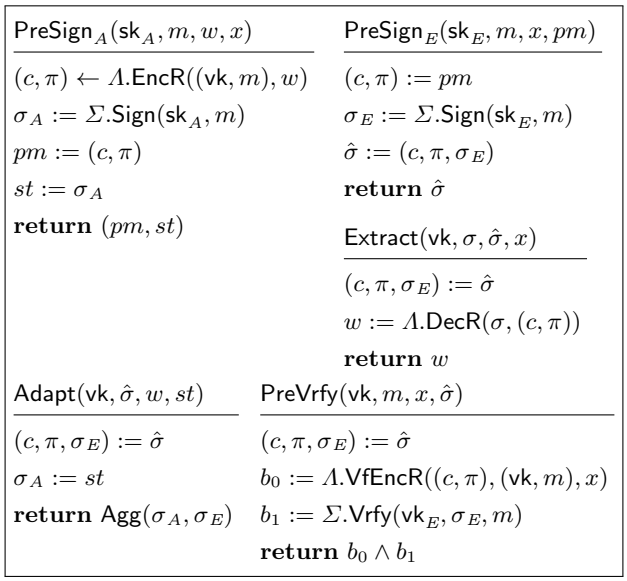



Fig. 10: Our construction of 2P-AISO-AS for BLS signatures. We denote by $\Lambda$ a VWER and by $\Sigma$ a DS. Agg denotes the BLS aggregation function of two signatures. To ease notation, we assume that after executing $(\mathsf{vk}, (\mathsf{sk}_A, \mathsf{vk}_A), (\mathsf{sk}_E, \mathsf{vk}_E)) \leftarrow \mathsf{KeyGen}(1^n)$, both $\mathsf{vk}_A$ and $\mathsf{vk}_E$ are implicit parameters.

$m), x)$ is the proof verification algorithm; and $w \leftarrow \mathsf{DecR}(\sigma, (c, \pi))$ is the decryption algorithm. A verifiable witness encryption for a relation scheme should satisfy *one wayness*, *verifiability* and *zero-knowledge*.

**Construction details.** Our construction, presented in Figure 10, follows the high-level approach outlined in Section 2. For PreSign, the *Adaptor* creates the witness encryption $(c, \pi)$ of $w$ under the tuple $(\mathsf{vk}, m)$ of the aggregated verification key and the message using VWER. This implies that a valid signature for $(\mathsf{vk}, m)$ will allow for decrypting the witness. The presignature now consists of $(c, \pi)$, as well as the signature share $\sigma_E$ of $E$. This intuitively allows (1) the *Adaptor* to adapt by creating their own signature share $\sigma_A$ and to aggregate it with $\sigma_E$ to a full signature; (2) the *Extractor* to extract decrypting $c$ using the full signature $\sigma$.

**Security analysis.** Next, we state the security of our construction and defer the proofs to Section G, where we give its own theorem statement and proof for each property separately.

**Theorem 2.** *If BLS signatures are unforgeable and unique and if* VWER *is zero-knowledge and verifiable and if* R *is q-hard, then the construction in Figure 10 satisfies full extractability, weak unforgeability, designated adaptability, unlinkability, computational preverify soundness and one-way signatures in the Random Oracle Model.*

In the appendix (Section H), we also present a generic construction of our 2P-AISO-AS for any underlying signature scheme and any hard relation using general multi-party-computation. Since this construction does not immediately result in practical instantiations, this construction serves as a feasibility result, which shows that 2P-AISO-AS can be realized in principle, leaving the development of further practical schemes (e.g., for other unique signature schemes or relations) as an open research venue.

## 6 2P-AISO-AS Evaluation

In this section, we evaluate the practicality of using 2P-AISO-AS to deploy blockchain applications in existing BLS-based cryptocurrencies. For that, we focus on answering three questions:

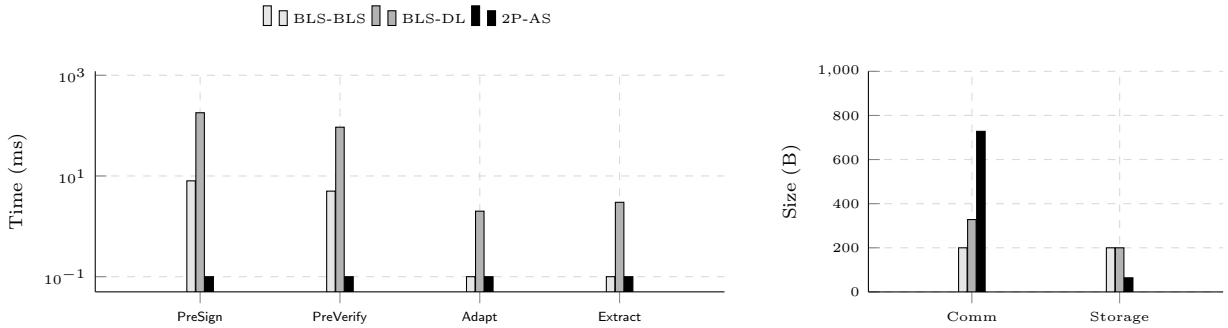



Fig. 11: Performance comparison: (Left) Computation times, with y-axis in logarithmic scale. (Right) Communication and on-chain storage overhead.

- *Q1: How does the performance of 2P-AISO-AS constructions compare to existing AS constructions?*
- *Q2: Is it possible to use 2P-AISO-AS to deploy existing AS-based blockchain protocols in BLS-based cryptocurrencies?*
- *Q3: How does the performance of 2P-AISO-AS-based applications compare to existing custom-designed protocols for BLS-based cryptocurrencies?*

**Source code and testbed.** The complete source code for our evaluation is available in [24]. We use a commodity laptop with a 10-core 4.6GHz processor and 16 GB RAM memory, running Ubuntu 22.04. We run 100 times each test and report the average computation times. For the communication overhead, we report the size of messages that need to be communicated between *Adaptor* and *Extractor*. Finally, we evaluate the on-chain storage overhead as the size of the signatures that are stored on-chain. For an in-depth discussion of the testbed and implementation, see Section C.

### 6.1 2P-AISO-AS Constructions Performance

The performance evaluation is summarized in Figure 11. As expected, the presignature creation and verification in the BLS-DLog construction is the most computationally expensive of the three evaluated constructions as it relies on the computationally taxing VWER. Yet, the computation time is below 200 ms, which is practical for blockchain applications. The rest of the operations are significantly more efficient. In terms of communication overhead, both BLS-BLS and BLS-DLog constructions impose a communication overhead smaller than in the 2P-AS construction, intuitively as they do not need to exchanges to create a shared randomness for the presignature. Finally, the slightly larger size of a single BLS signature compared to a Schnorr signature (as required in 2P-AS) is amortized in the 2P-AISO-AS setting, as many BLS signatures can be aggregated into a single signature.

In summary, from this evaluation we can answer Q1 observing that our BLS-based 2P-AISO-AS constructions still provide practical performance in terms of computation and storage by outperforming existing AS constructions in terms of communication.

### 6.2 Building Blockchain Applications with 2P-AISO-AS

To assess the utility of 2P-AISO-AS, we use this building block to realize zero-knowledge contingent payments (ZKCP), universal atomic swaps (UAS) and coin mixing (CM), three prominent blockchain applications that are known to be realizable from AS [11, 2, 5]. These AS-based realizations were believed useless for blockchains whose transaction authorization is based on unique signature schemes, such as BLS, due to the impossibility result in [16]. Contrary to this belief, our 2P-AISO-AS-based constructions enable these applications to be instantiated for BLS-based blockchains, as well as in any other blockchain that supports unique signatures.

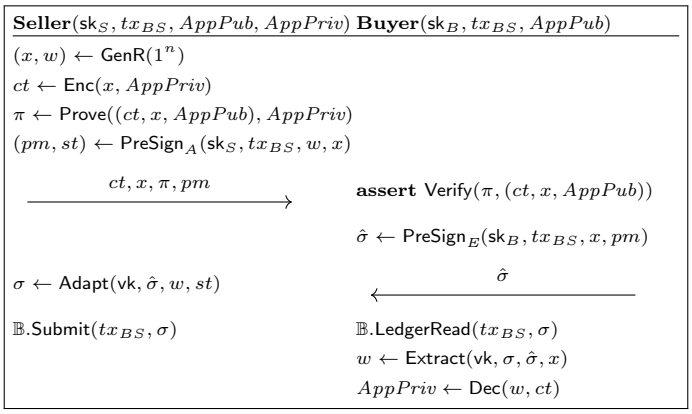



Fig. 12: Template for 2P-AISO-AS-based blockchain applications. Here, *AppPub* and *AppPriv* denote some public and private application-specific information, respectively.

**Generic design strategy.** Interestingly, many AS-based blockchain protocols follow the same template, which can be considered a generalization of the preimage sale presented in Section 2. This template, which has been made explicit in [25], is also compatible with 2P-AISO-AS, allowing us to easily cast protocols that follow the template to the BLS setting. We overview the template (obtained by replacing AS with 2P-AISO-AS in the original template from [25]) in Figure 12. At its core, the template describes a preimage sale between a seller and a buyer, with the only difference that the secret $w$ purchased by the buyer corresponds to a decryption key with some application-specific secret information *AppPriv*. In the course of the (preimage sale) protocol, the sender provides the ciphertext $ct$ containing the encryption of *AppPriv* as well as a zero-knowledge proof $\pi$ that certifies that $ct$ indeed contains the correct private information *AppPriv* (w.r.t. to some public application-specific information *AppPub*), encrypted with $w$. After that, the protocol executes the preimage sale protocol using 2P-AISO-AS for exchanging $w$ with a signature $\sigma$ on the payment transaction $tx_{BS}$. From observing $tx_{BS}$ signed with $\sigma$ on the blockchain, the buyer then learns $w$, which they can use to decrypt $ct$ and obtain *AppPriv*.

In the following, to answer Q2 and Q3, we describe how to obtain ZKCP, UAS, and CM as from the template through the right instantiation of (*AppPub*, *AppPriv*) with an application-specific relation, and evaluate the performance of instantiating the resulting protocols with the constructions BLS-BLS and BLS-DLog. We overview the evaluation results in Table 1, where we also report the performance of existing custom BLS-based protocols from the literature for comparison for those applications where such protocols exist.

**ZKCP.** In zero-knowledge contingent payments (ZKCP), a buyer pays a seller in exchange for a digital good $s$ (e.g., a signature on a digital document), such that $f(s) = 1$, where $f$ is a predicate that describes valid digital goods. As such ZKCP are realized through a straightforward instantiation of the template in Figure 12 (setting $AppPriv = s$ and $AppPub = f$), resulting in known standard AS-based ZKCP protocols (e.g., [11]), with the only difference that AS is now replaced by 2P-AISO-AS. By implementing 2P-AISO-AS with BLS-BLS and BLS-DLog, we obtain practical protocols (with sub-second runtime) for contingent payments where $f$ can be expressed in terms of $\mathsf{R}_{BLS}$ or $\mathsf{R}_{DLog}$, respectively. To our knowledge, these protocols constitute the first ever reported ZKCP protocols compatible with BLS-based blockchains.

**UAS.** In an atomic swap protocol (which is known to be realizable from standard AS [2]), the buyer transfers coins to the seller in $\mathbb{B}_1$ in exchange for coins that the seller owns in a different blockchain $\mathbb{B}_2$. Hence, setting $AppPub = (tx_{SB}, \mathsf{vk}_{SB})$ and $AppPriv = \sigma_S$, where $tx_{SB}$ is the transaction that transfers coins from seller to buyer in $\mathbb{B}_2$ and $\mathsf{vk}_{SB}$ is the aggregated verification key of seller and buyer owning the coins to be transferred in $tx_{SB}$, the corresponding instantiation of template in Figure 12 resembles the standard AS-based atomic swap protocols (e.g., [2]), (where AS is replaced by 2P-AISO-AS).

Implementing 2P-AISO-AS with BLS-BLS yields an atomic swap protocol for swaps between two BLS-based blockchains. The work in [17] proposes a custom-designed atomic swap protocol, which also targets exactly this scenario. Indeed, it turns out that the protocol from [17] constitutes a special case of our approach. Consequently, the performance of both approaches is identical (cf. Table 1).

When implementing 2P-AISO-AS in the UAS-instantiation of the template with BLS-DLog, we obtain a practical atomic swap protocol (with sub-second execution time) for swaps between a BLS-based blockchain (e.g., the Chia Network) and a non-BLS blockchain (e.g., Bitcoin). To our knowledge, there exist no other protocols so far, which enable such functionality between a BLS and a non-BLS chain.

**CM.** Coin mixing services enable users to make transfers via an intermediary (the *mixer*), which obfuscates the connection between the in-going and out-going payments (e.g., to hide the relationship between two existing accounts of two users). One can think of such protocols as realizing an atomic execution of two transactions $tx_{AM}$ (between a user $A$ and the mixer $M$) and $tx_{MB}$ (between the mixer $M$ and another user $B$), similar to an atomic swap, but ensuring that the link between these two transactions is hidden from the mixer. Correspondingly, we can find a fitting instantiation for the template in Figure 12. We can then obtain a CM protocol by implementing 2P-AISO-AS in the instantiated template with our BLS-DLog. We need to opt for the more general option here because the CM-specific instantiation mandates an exchange of a BLS signature against a *blind BLS signature* (and not another plain BLS signature). We report the execution time of the resulting protocol in Table 1.

The problem of realizing coin mixing protocols from AS has been extensively studied [3, 5]. However, due to the impossibility result from [16], existing protocols were thought not to be realizable for BLS signatures. As a response, recent work [4] proposes Sweep-UC, a novel BLS-based coin mixing protocol that deviates from the structure of previously proposed AS-based protocols and instead proposes a complex, custom protocol that incurs high communication and computation costs (as reported in Table 1). More concretely, Sweep-UC proposes two subprotocols: *exchange* and *redeem*. The redeem protocol simply encrypts the signature on $tx_{MB}$ from the mixer requiring a blind signature on message *bsm* as decryption key. The exchange protocol, the core component of Sweep-UC, is used to exchange the blind signature on *bsm* from the mixer in return for the payment $tx_{AM}$ from user $A$ to the mixer. Interestingly, the authors in [4] use the template of our protocol in Figure 12 as an illustrative example of an efficient, desirable AS-based implementation of the *exchange protocol*. However, the authors argue that a new protocol with a different structure is necessary due to the impossibility of building AS for BLS. Contrary to that belief, using 2P-AISO-AS, not only it is indeed possible to realize BLS-based coin mixing protocols following the desired blueprint in Figure 12, but also in a more efficient way than the ad-hoc protocol proposed in [4]. As shown in Table 1, our protocol incurs a computation improvement by a factor of 22 as opposed to the best case performance of the protocol from [4] (266 milliseconds vs. 6 seconds), and an improvement of factor 24 in terms of communication overhead (1832 bytes vs. 43 KiB).

The main reason for this efficiency improvement is that the ad-hoc protocol in [4] for the case of BLS encrypts the blind signature $\overline{\sigma}_S$ using a one-time pad with the BLS signature $\sigma_S$ from the mixer on $tx_{AM}$ as decryption key, and then requires a complex cut-and-choose-based zero-knowledge proof to prove that the ciphertext is well-formed. Instead, our approach in Figure 12 directly encrypts $\overline{\sigma}_S$ under the statement $x$ using a simpler verifiable encrypted signature (realizable with ElGamal encryption and a Sigma protocol as zero-knowledge proof), which is significantly more efficient than the combination of one-time pad and cut-and-choose proof in [4]. The concrete implementation of the 2P-AISO-AS for BLS also requires a cut-and-choose proof as part of the $\mathsf{VWER}$, but this is done over a ciphertext that encrypts a freshly chosen random value (i.e., the witness $w$), which permits to precompute most of the required computation and hence, it is significantly more efficient than the one in [4].

**Limitations.** The 2P-AISO-AS notion can be used to realize from BLS any application where the adaptor knows the witness $w$ at the time of executing the presigning protocol. As shown in this section, this is the case for ZKCP, UAS and CM. A representative application where this is not the case is multi-hop payments [26], where a payment is forwarded across (potentially many) intermediaries. In this case, the sender of the multi-hop payment prepares the statements that the intermediary users need to create presignatures for both incoming and outgoing transactions. At this stage, the intermediary does not know the witness that will

Table 1: Summary of blockchains applications realizable for unique signatures from 2P-AISO-AS (our work) and in the literature (custom-designs). Time is shown in seconds and communication in bytes.

| | Setting | | Our Work | | | Literature | | |
|---|---|---|---|---|---|---|---|---|
| | $\mathbb{B}$ sig. | Product $s$ | $\exists$ | Time | Comm. | $\exists$ | Time | Comm. |
| **ZKCP** | BLS | BLS sig. | ✓ | 0.015 | 400 | ✗ | – | – |
| | BLS | Any | ✓ | 0.436 | 84074 | ✗ | – | – |
| | Any Unique | Any | ✓ | Feasible | Feasible | ✗ | – | – |
| | $\mathbb{B}_1$ sig. | $\mathbb{B}_2$ sig. | $\exists$ | Time | Comm. | $\exists$ | Time | Comm. |
| **UAS** | BLS | BLS | ✓ | 0.015 | 400 | ✓(1) | 0.015 | 400 |
| | BLS | Any | ✓ | 0.268 | 1232 | ✗ | – | – |
| | Any Unique | Any | ✓ | Feasible | Feasible | ✗ | – | – |
| | $\mathbb{B}$ sig. | | $\exists$ | Time | Comm. | $\exists$ | Time | Comm. |
| **CM** | BLS | | ✓ | 0.266 | 1832 | ✓(2) | Best: 6.4<br>Worst: 19 | 43 kB |
| | Any Unique | | ✓ | Feasible | Feasible | ✗ | – | – |

(1): Prior ad-hoc construction in [17]. It is a 2P-AISO-AS-based instance (Section F).
(2): Prior ad-hoc construction in [4]. Our approach is more efficient.
Cases marked as *Feasible* are theoretically realizable from our Any-Any construction.

be used to adapt the presignature into a valid signature, since it is enough for an intermediary to know that extracting the witness from an outgoing transaction will be sufficient to adapt and claim the incoming transaction. Therefore, our methodology cannot be directly applied to realize multi-hop payments from 2P-AISO-AS over blockchains that use BLS for transaction authorization. Designing multi-hop payment protocols in this setting remains an interesting open research question.

## 7 Conclusions

We propose 2P-AISO-AS, a two-party adaptor signature scheme that can be realized from unique signatures such as BLS. To this end, we treat presignature generation as a two-party protocol between an *Extractor* and an *Adaptor* and devise fine-grained security definitions for 2P-AISO-AS that meet the requirements of many blockchain protocols while at the same time evading the existing impossibility result from [16], which, so far, excluded the existence of a unique adaptor signatures. To showcase the utility of 2P-AISO-AS, we provide efficient BLS-based constructions of 2P-AISO-AS and show that they can be used to obtain efficient BLS-based blockchain protocols by replacing AS with 2P-AISO-AS in existing AS-based protocols, including zero-knowledge contingent payments, atomic swaps and coin mixing.

## Acknowledgements

The work of Erkan Tairi is supported by the Austrian Science Fund (FWF) Project J4879-N, and research grants from Bakar Funds, Peder Sather Funds and Input Output Global. This work is also partially supported by projects PRODIGY (TED2021-132464B-I00) and ESPADA (PID2022-142290OB-I00), which are funded by MCIN/AEI/10.13039/501100011033/ and the European Union NextGenerationEU / PRTR, and by the Excellence Seal Mar'ia de Maeztu with grant number CEX2024-001471-M, which is funded by MICIU/AEI/10.13039/501100011033. This work has been supported by the Heinz Nixdorf Foundation through a Heinz Nixdorf Research Group (HN-RG) and funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy-EXC 2092 CASA-390781972.

## References


1. Lukas Aumayr, Oguzhan Ersoy, Andreas Erwig, Sebastian Faust, Kristina Hostáková, Matteo Maffei, Pedro Moreno-Sanchez, and Siavash Riahi. Generalized channels from limited blockchain scripts and adaptor signatures. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part II*, volume 13091 of *LNCS*, pages 635–664. Springer, Cham, December 2021.
2. Sri Aravinda Krishnan Thyagarajan, Giulio Malavolta, and Pedro Moreno-Sanchez. Universal atomic swaps: Secure exchange of coins across all blockchains. In *2022 IEEE Symposium on Security and Privacy*, pages 1299–1316. IEEE Computer Society Press, May 2022.
3. Erkan Tairi, Pedro Moreno-Sanchez, and Matteo Maffei. $A^2L$: Anonymous atomic locks for scalability in payment channel hubs. In *2021 IEEE Symposium on Security and Privacy*, pages 1834–1851. IEEE Computer Society Press, May 2021.
4. Lucjan Hanzlik, Julian Loss, Sri Aravinda Krishnan Thyagarajan, and Benedikt Wagner. Sweep-UC: Swapping coins privately. In *2024 IEEE Symposium on Security and Privacy*, pages 3822–3839. IEEE Computer Society Press, May 2024.
5. Noemi Glaeser, Matteo Maffei, Giulio Malavolta, Pedro Moreno-Sanchez, Erkan Tairi, and Sri Aravinda Krishnan Thyagarajan. Foundations of coin mixing services. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 1259–1273. ACM Press, November 2022.
6. Xianrui Qin, Shimin Pan, Arash Mirzaei, Zhimei Sui, Oguzhan Ersoy, Amin Sakzad, Muhammed F. Esgin, Joseph K. Liu, Jiangshan Yu, and Tsz Hon Yuen. BlindHub: Bitcoin-compatible privacy-preserving payment channel hubs supporting variable amounts. In *2023 IEEE Symposium on Security and Privacy*, pages 2462–2480. IEEE Computer Society Press, May 2023.
7. Zhonghui Ge, Jiayuan Gu, Chenke Wang, Yu Long, Xian Xu, and Dawu Gu. Accio: Variable-amount, optimized-unlinkable and NIZK-free off-chain payments via hubs. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 1541–1555. ACM Press, November 2023.
8. Lukas Aumayr, Sri Aravinda Krishnan Thyagarajan, Giulio Malavolta, Pedro Moreno-Sanchez, and Matteo Maffei. Sleepy channels: Bi-directional payment channels without watchtowers. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 179–192. ACM Press, November 2022.
9. Lukas Aumayr, Kasra Abbaszadeh, and Matteo Maffei. Thora: Atomic and privacy-preserving multi-channel updates. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 165–178. ACM Press, November 2022.
10. Ertem Nusret Tas, István András Seres, Yinuo Zhang, Márk Melczer, Mahimna Kelkar, Joseph Bonneau, and Valeria Nikolaenko. Atomic and fair data exchange via blockchain. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024*, pages 3227–3241. ACM Press, October 2024.
11. Javier Gomez-Martinez, Dimitrios Vasilopoulos, Pedro Moreno-Sanchez, and Dario Fiore. Algebraic zero knowledge contingent payment. In Marc Fischlin and Veelasha Moonsamy, editors, *ACNS 2025, Part II*, volume 15826 of *LNCS*, pages 224–254. Springer, Cham, June 2025.
12. Diego Castejon-Molina, Dimitrios Vasilopoulos, and Pedro Moreno-Sanchez. MixBuy: Contingent payment in the presence of coin mixers. *PoPETs*, 2025(1):671–706, January 2025.
13. Varun Madathil, Sri Aravinda Krishnan Thyagarajan, Dimitrios Vasilopoulos, Lloyd Fournier, Giulio Malavolta, and Pedro Moreno-Sanchez. Cryptographic oracle-based conditional payments. In *NDSS 2023*. The Internet Society, February 2023.
14. Paul Gerhart, Dominique Schröder, Pratik Soni, and Sri Aravinda Krishnan Thyagarajan. Foundations of adaptor signatures. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part II*, volume 14652 of *LNCS*, pages 161–189. Springer, Cham, May 2024.
15. Chia Network. Chia consensus. https://www.chia.net/wp-content/uploads/2022/09/Chia-New-Consensus-0.9.pdf.
16. Andreas Erwig, Sebastian Faust, Kristina Hostáková, Monosij Maitra, and Siavash Riahi. Two-party adaptor signatures from identification schemes. In Juan Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 451–480. Springer, Cham, May 2021.
17. Huseyin Gokay, Foteini Baldimtsi, and Giuseppe Ateniese. Atomic swaps for boneh-lynn-shacham (BLS) based blockchains. In Joaquin Garcia-Alfaro, Rafał Kozik, Michał Choraś, and Sokratis Katsikas, editors, *ESORICS 2024, Part III*, volume 14984 of *LNCS*, pages 341–361. Springer, Cham, September 2024.
18. Richard Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In *18th ACM STOC*, pages 364–369. ACM Press, May 1986.

19. Arka Rai Choudhuri, Matthew Green, Abhishek Jain, Gabriel Kaptchuk, and Ian Miers. Fairness in an unfair world: Fair multiparty computation from public bulletin boards. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 719–728. ACM Press, October / November 2017.
20. Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Witness encryption and null-IO from evasive LWE. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part I*, volume 13791 of *LNCS*, pages 195–221. Springer, Cham, December 2022.
21. Wei Dai, Tatsuaki Okamoto, and Go Yamamoto. Stronger security and generic constructions for adaptor signatures. In Takanori Isobe and Santanu Sarkar, editors, *INDOCRYPT 2022*, volume 13774 of *LNCS*, pages 52–77. Springer, Cham, December 2022.
22. Elizabeth C. Crites, Chelsea Komlo, and Mary Maller. Fully adaptive Schnorr threshold signatures. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part I*, volume 14081 of *LNCS*, pages 678–709. Springer, Cham, August 2023.
23. Ran Cohen, Jack Doerner, Eysa Lee, Anna Lysyanskaya, and Lawrence Roy. An unstoppable ideal functionality for signatures and a modular analysis of the dolev-strong broadcast. Cryptology ePrint Archive, Report 2024/1807, 2024.
24. Adaptor signatures meet bls: Enabling efficient blockchain applications with unique adaptor signatures: Source code. https://anonymous.4open.science/r/Adaptor_Signatures_Meet_BLS-ECDO/README.md.
25. Diego Castejon-Molina, Erkan Tairi, Dimitrios Vasilopoulos, and Pedro Moreno-Sanchez. Foundations of verifiably encrypted (blind) signatures. Cryptology ePrint Archive, Paper 2026/789, 2026.
26. Giulio Malavolta, Pedro Moreno-Sanchez, Clara Schneidewind, Aniket Kate, and Matteo Maffei. Anonymous multi-hop locks for blockchain scalability and interoperability. In *NDSS 2019*. The Internet Society, February 2019.
27. Diego Castejon-Molina, Dimitrios Vasilopoulos, and Pedro Moreno-Sanchez. MixBuy: Contingent payment in the presence of coin mixers. Cryptology ePrint Archive, Report 2024/953, 2024.
28. Andrew Poelstra. Scriptless scripts. *Presentation Slides*, 2017.
29. Muhammed F. Esgin, Oguzhan Ersoy, and Zekeriya Erkin. Post-quantum adaptor signatures and payment channel networks. In Liqun Chen, Ninghui Li, Kaitai Liang, and Steve A. Schneider, editors, *ESORICS 2020, Part II*, volume 12309 of *LNCS*, pages 378–397. Springer, Cham, September 2020.
30. Erkan Tairi, Pedro Moreno-Sanchez, and Matteo Maffei. Post-quantum adaptor signature for privacy-preserving off-chain payments. In Nikita Borisov and Claudia Díaz, editors, *FC 2021, Part II*, volume 12675 of *LNCS*, pages 131–150. Springer, Berlin, Heidelberg, March 2021.
31. Xiangyu Liu, Ioannis Tzannetos, and Vassilis Zikas. Adaptor signatures: New security definition and a generic construction for NP relations. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part II*, volume 15485 of *LNCS*, pages 168–193. Springer, Singapore, December 2024.
32. Michele Ciampi, Xiangyu Liu, Ioannis Tzannetos, and Vassilis Zikas. Universal adaptor signatures from blackbox multi-party computation. Cryptology ePrint Archive, Paper 2024/1773, 2024.
33. Sergiu Bursuc and Sjouke Mauw. Contingent payments from two-party signing and verification for abelian groups. In *CSF 2022 Computer Security Foundations Symposium*, pages 195–210. IEEE Computer Society Press, August 2022.
34. Ruben Baecker, Paul Gerhart, Jonathan Katz, and Dominique Schröder. Fair exchange for decentralized autonomous organizations via threshold adaptor signatures. Cryptology ePrint Archive, Paper 2025/388, 2025.
35. Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 416–432. Springer, Berlin, Heidelberg, May 2003.
36. Sri Aravinda Krishnan Thyagarajan and Giulio Malavolta. Lockable signatures for blockchains: Scriptless scripts for all signatures. In *2021 IEEE Symposium on Security and Privacy*, pages 937–954. IEEE Computer Society Press, May 2021.
37. Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 468–499. Springer, Cham, April / May 2018.
38. Eike Kiltz, Daniel Masny, and Jiaxin Pan. Optimal security proofs for signatures from identification schemes. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 33–61. Springer, Berlin, Heidelberg, August 2016.

# A Extended Preliminaries

## A.1 Public-Key Encryption Scheme

**Definition 9 (Public key encryption scheme).** *A* public key encryption scheme $\mathsf{PKE} := (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is defined as follows:*

- $(\mathsf{dk}, \mathsf{ek}) \leftarrow \mathsf{KeyGen}(1^n)$. *A* PPT *algorithm that takes as input the security parameter* $n$*. It outputs a pair of encryption and decryption keys* $(\mathsf{dk}, \mathsf{ek})$.
- $ct \leftarrow \mathsf{Enc}(\mathsf{ek}, m)$. *A* PPT *algorithm that takes as input the encryption key* $\mathsf{ek}$ *and a message* $m$*. It outputs a ciphertext* $ct$.
- $\{m, \perp\} \leftarrow \mathsf{Dec}(\mathsf{dk}, ct)$. *A* DPT *algorithm that takes as input the decryption key* $\mathsf{dk}$ *and a ciphertext* $ct$*. It outputs a message* $m$ *or a special symbol* $\perp$ *denoting failure.*

*A public key encryption scheme is* correct *if for every message* $m \in \mathcal{M}$ *it holds that:*

$$\Pr\left[m = m' \middle| \begin{matrix}(\mathsf{dk}, \mathsf{ek}) \leftarrow \mathsf{KeyGen}(1^n) \\ m' := \mathsf{Dec}(\mathsf{dk}, \mathsf{Enc}(\mathsf{ek}, m))\end{matrix}\right] = 1$$

We now review the notion of security for a public key encryption scheme.

**Definition 10 (Indistinguishable encryptions under chosen-plaintext attacks).** *A public key encryption scheme* $\mathsf{PKE} := (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *has* indistinguishable encryptions under chosen-plaintext attacks *(or is* CPA-secure*) if for all* PPT *adversaries* $\mathcal{A}$*, there is a negligible function* $\mathsf{negl}(n)$ *such that:*

$$\Pr\left[\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{PKE}}(n) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(n)$$

*where the experiment* $\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{PKE}}(n)$ *is defined as follows:*

$\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{PKE}}(n)$

1: $(\mathsf{dk}, \mathsf{ek}) \leftarrow \mathsf{KeyGen}(1^n)$
2: $(m_0, m_1) \leftarrow \mathcal{A}(\mathsf{ek})$
3: $b \leftarrow_{\$} \{0, 1\}$
4: $ct_b \leftarrow \mathsf{Enc}(\mathsf{ek}, m_b)$
5: $b^* \leftarrow \mathcal{A}(ct_b)$
6: $b_0 := (|m_0| = |m_1|)$
7: $b_1 := (b = b^*)$
8: **return** $b_0 \wedge b_1$

## A.2 Symmetric-Key Encryption Scheme

**Definition 11 (Symmetric key encryption scheme).** *A* symmetric key encryption scheme $\mathsf{SKE} := (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is defined as follows:*

- $\mathsf{k} \leftarrow \mathsf{KeyGen}(1^n)$. *A* PPT *algorithm that takes as input the security parameter* $n$*. It outputs a encryption and decryption key* $\mathsf{k}$.
- $ct \leftarrow \mathsf{Enc}(\mathsf{k}, m)$. *A* PPT *algorithm that takes as input the key* $\mathsf{k}$ *and a message* $m$*. It outputs a ciphertext* $ct$.
- $\{m, \perp\} \leftarrow \mathsf{Dec}(\mathsf{k}, ct)$. *A* DPT *algorithm that takes as input the key* $\mathsf{k}$ *and a ciphertext* $ct$*. It outputs a message* $m$ *or a special symbol* $\perp$ *denoting failure.*

*A symmetric key encryption scheme is* correct *if for every message $m \in \mathcal{M}$ it holds that:*

$$\Pr\left[m = m' \middle| \begin{array}{c} \mathsf{k} \leftarrow \mathsf{KeyGen}(1^n) \\ m' := \mathsf{Dec}(\mathsf{k}, \mathsf{Enc}(\mathsf{k}, m)) \end{array}\right] = 1$$

We now review the notion of security for a symmetric key encryption scheme.

**Definition 12 (Indistinguishable encryptions under chosen-plaintext attacks).** *A symmetric key encryption scheme* $\mathsf{SKE} := (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *has* indistinguishable encryptions under chosen-plaintext attacks *(or is* CPA-secure*) if for all* PPT *adversaries $\mathcal{A}$, there is a negligible function* $\mathsf{negl}(n)$ *such that:*

$$\Pr\left[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{SKE}}(n) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(n)$$

*where the experiment* $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{SKE}}(n)$ *is defined as follows:*

$\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{SKE}}(n)$

1: $\mathsf{k} \leftarrow \mathsf{KeyGen}(1^n)$
2: $(m_0, m_1) \leftarrow \mathcal{A}(\cdot)$
3: $b \leftarrow_{\$} \{0,1\}$
4: $ct_b \leftarrow \mathsf{Enc}(\mathsf{k}, m_b)$
5: $b^* \leftarrow \mathcal{A}(ct_b)$
6: $b_0 := (|m_0| = |m_1|)$
7: $b_1 := (b = b^*)$
8: **return** $b_0 \wedge b_1$

### A.3 Hash Functions

**Definition 13 (Hash function).** *A* hash *function* $\mathsf{H}$ *is defined with respect an input space $\mathcal{M}$ and an output space $\mathcal{H}$, as follows:*

- $h \leftarrow \mathsf{H}(m)$*. A* DPT *algorithm that takes as input a message $m \in \mathcal{M}$. It outputs a hash value $h$.*

We now review the notions of security for a hash function.

**Definition 14 (Colllision resistance).** *A hash function* $\mathsf{H}$ *is* collision resistant *if for all* PPT *adversaries $\mathcal{A}$, there is a negligible function* $\mathsf{negl}(n)$ *such that:*

$$\Pr[\mathsf{CollRes}_{\mathcal{A},\mathsf{H}}(n) = 1] \leq \mathsf{negl}(n)$$

*where* $\mathsf{CollRes}_{\mathcal{A},\mathsf{H}}(n)$ *is defined as follows:*

$\mathsf{CollRes}_{\mathcal{A},\mathsf{H}}(n)$

1: $(m, m') \leftarrow \mathcal{A}^{\mathsf{H}}(n)$
2: $b_0 := (m \neq m')$
3: $b_1 := (\mathsf{H}(m) = \mathsf{H}(m'))$
4: **return** $b_0 \wedge b_1$

**Definition 15 (Pre-image resistance).** *A hash function* $\mathsf{H}$ *is* preimage resistant *if for all* PPT *adversaries* $\mathcal{A}$*, there is a negligible function* $\mathsf{negl}(n)$ *such that:*

$$\Pr[\mathsf{PreImageRes}_{\mathcal{A},\mathsf{H}}(n) = 1] \leq \mathsf{negl}(n)$$

*where* $\mathsf{PreImageRes}_{\mathcal{A},\mathsf{H}}(n)$ *is defined as follows:*

$\mathsf{PreImageRes}_{\mathcal{A},\mathsf{H}}(n)$

1: $h \leftarrow_{\$} \mathcal{H}$
2: $m \leftarrow \mathcal{A}(h)$
3: **return** $(h = \mathsf{H}(m))$

**Definition 16 (Second Pre-image resistance).** *A hash function* $\mathsf{H}$ *is* second preimage resistant *if for all* PPT *adversaries* $\mathcal{A}$*, there is a negligible function* $\mathsf{negl}(n)$ *such that:*

$$\Pr[\mathsf{SecondPreImageRes}_{\mathcal{A},\mathsf{H}}(n) = 1] \leq \mathsf{negl}(n)$$

*where* $\mathsf{SecondPreImageRes}_{\mathcal{A},\mathsf{H}}(n)$ *is defined as follows:*

$\mathsf{SecondPreImageRes}_{\mathcal{A},\mathsf{H}}(n)$

1: $m \leftarrow_{\$} \mathcal{M}$
2: $m' \leftarrow \mathcal{A}(m)$
3: $b_0 := m \neq m'$
4: $b_1 := \mathsf{H}(m) = \mathsf{H}(m')$
5: **return** $b_0 \wedge b_1$

### A.4 Extractable Commitments

**Definition 17 (Commitment Scheme).** *A* commitment scheme $\mathsf{COM} := (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ *is defined as follows:*

- $p \leftarrow \mathsf{Setup}(1^n)$*. A* PPT *algorithm that takes as input the security parameter* $n$ *and outputs the commitment parameters* $p$*.*
- $(com, open) \leftarrow \mathsf{Commit}(p, x)$*. A* PPT *algorithm that takes as input the commitment parameters* $p$ *and a message* $x$*. It outputs the commitment* $com$ *and the opening information* $open$*.*
- $\{0,1\} \leftarrow \mathsf{Open}(p, x, com, open)$*. A* PPT *algorithm tha takes as input the commitment parameters* $p$*, a commited value* $x$*, a commitment* $com$ *and the opening information* $open$*. It outputs a bit* $b \in \{0,1\}$*.*

*A commitment scheme is* correct *if for every message* $x \in \mathcal{X}$ *it holds that:*

$$\Pr\left[\mathsf{Open}(p, x, com, open) \,\middle|\, \begin{array}{c} p \leftarrow \mathsf{Setup}(1^n) \\ (com, open) \leftarrow \mathsf{Commit}(p, x) \end{array}\right] \geq 1 - \mathsf{negl}(n)$$

We now review the notions of security for a commitment scheme.

**Definition 18 (Binding).** *A commitment scheme* $\mathsf{COM} := (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ *is* binding *if for all* PPT *adversaries* $\mathcal{A}$*, there is a negligible function* $\mathsf{negl}(n)$ *such that:*

$$\Pr[\mathsf{Binding}_{\mathcal{A},\mathsf{COM}}(n) = 1] \leq \mathsf{negl}(n)$$

*where the experiment* $\mathsf{Binding}_{\mathcal{A},\mathsf{COM}}(n)$ *is defined as follows:*

$\mathsf{Binding}_{\mathcal{A},\mathsf{COM}}(n)$

1: $p \leftarrow \mathsf{Setup}(1^n)$
2: $(com, open, x, open', x') \leftarrow \mathcal{A}(p)$
3: $b_0 := \mathsf{Open}(p, x, com, open)$
4: $b_1 := \mathsf{Open}(p, x', com, open')$
5: $b_2 := x \neq x'$
6: **return** $b_0 \wedge b_1 \wedge b_2$

**Definition 19 (Hiding).** *A commitment scheme* $\mathsf{COM} := (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ *is* hiding *if for all* PPT *adversaries* $\mathcal{A}$*, there is a negligible function* $\mathsf{negl}(n)$ *such that*

$$\Pr[\mathsf{Hiding}_{\mathcal{A},\mathsf{COM}}(n) = 1] \leq \frac{1}{2} + \mathsf{negl}(n)$$

*where the experiment* $\mathsf{Hiding}_{\mathcal{A},\mathsf{COM}}(n)$ *is defined as follows:*

$\mathsf{Hiding}_{\mathcal{A},\mathsf{COM}}(n)$

1: $p \leftarrow \mathsf{Setup}(1^n)$
2: $(x_0, x_1) \leftarrow \mathcal{A}(p)$
3: $b \leftarrow_{\$} \{0,1\}$
4: $(com_b, open_b) \leftarrow \mathsf{Commit}(p, x_b)$
5: $b^* \leftarrow \mathcal{A}(com_b)$
6: **return** $b = b^*$

**Definition 20 (Extractable commitments).** *A commitment scheme is* extractable *if* Setup *outputs an additional secret trapdoor* $td$ *and there exists an efficient* PPT *algorithm* Extract *such that for any value* $x$ *and any tuple* $(com, open) \leftarrow \mathsf{Commit}(p, x)$ *it holds that* $\mathsf{Extract}(p, td, com) = x$ *except with negligible probability.*

### A.5 Hard Relations

**Definition 21 (Binary relation).** *A binary relation* R *is a function* $\mathsf{R} : \mathcal{X} \times \mathcal{W} \to \{0,1\}$*. The language associated to* R *is*

$$L_{\mathsf{R}} := \{x \in \mathcal{X} \mid \exists w \in \mathcal{W} : \mathsf{R}(x, w) = 1\}$$

**Definition 22 (Hard relation).** *A relation* R *is hard if (i) there exists a* PPT *sampling algorithm* $\mathsf{GenR}(1^n)$ *that on input the security parameter* $n$*, it outputs* $(x, w) \in \mathsf{R}$*; (ii) there exists a* PPT *algorithm* $\mathcal{D}$ *such that* $\mathcal{D}(x, w) = 1 \Leftrightarrow (x, w) \in \mathsf{R}$*; (iii) for all* PPT *algorithms* $\mathcal{A}$ *it holds that:*

$$\Pr\left[(x, w^*) \in \mathsf{R} \middle| \begin{matrix}(x, w) \leftarrow \mathsf{GenR}(1^n) \\ w^* \leftarrow \mathcal{A}(x)\end{matrix}\right] \leq \mathsf{negl}(n)$$

**Definition 23 ($q$-hardness).** *A relation* R *is* $q$*-hard if for all* PPT *algorithms* $\mathcal{A}$ *it holds that:*

$$\Pr\left[\exists i \in [q] : (x_i, w^*) \in \mathsf{R} \middle| \begin{matrix}\textbf{for } i \in [q] : (x, w)_i \leftarrow \mathsf{GenR}(1^n) \\ w^* \leftarrow \mathcal{A}(\{x_i\}_{i \in [q]})\end{matrix}\right] \leq \mathsf{negl}(n)$$

**Definition 24 (One-More-Hardness).** *A relation* R *is one-more-hard if for all* $q$ *in* $\mathbb{N}$ *and for any* PPT *algorithms* $\mathcal{A}$ *there exists a negligible function* negl *such that* $\Pr[\textit{One-More}_{\mathcal{A},\mathsf{R}}(n, q) = 1] \leq \mathsf{negl}(n)$*, where the experiment* $\textit{One-More}_{\mathcal{A},\mathsf{R}}$ *is defined in Figure 13.*

| $\mathit{One\text{-}More}_{\mathcal{A},\mathsf{R}}(n,q)$ | $\mathsf{OWit}(x)$ |
|---|---|
| $k := 0$ | **assert** $\mathcal{T}[x] \neq \bot$ |
| **for** $i \in [q]$ : | **assert** $k < q$ |
| $\quad (x_i, w_i) \leftarrow \mathsf{GenR}(n)$ | $k := k + 1$ |
| $\quad \mathcal{T}[x_i] := w_i$ | **return** $w := \mathcal{T}[x]$ |
| $(w_1^*, \ldots, w_q^*) \leftarrow \mathcal{A}^{\mathsf{OWit}}(x_1, \ldots, x_q)$ | |
| $c_1 := \forall i \in [q] : (x_i, w_i^*) \in \mathsf{R}$ | |
| $c_2 := k < q$ | |
| **return** $c_1 \wedge c_2$ | |

Fig. 13: One-More-Hardness of a relation $\mathsf{R}$ where $q$ is the maximum number of queries to the oracle $\mathsf{OWit}$.

### A.6 Non-Interactive Zero-Knowledge Argument system

**Definition 25 (Non-Interactive Zero Knowledge Proof).** *A* non-interactive zero knowledge proof *for a relation* $\mathsf{R}$ *is a tuple of algorithms* $\mathsf{NIZK} := (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *defined as follows:*

- $crs \leftarrow \mathsf{Setup}(1^n)$. *A* PPT *algorithm that takes as input the security parameter* $n$*. It outputs a common reference string* $crs$*.*
- $\pi \leftarrow \mathsf{Prove}(crs, x, w)$. *A* PPT *algorithm that takes as input the common reference string* $crs$*, a statement* $x$ *and a witness* $w$*. It outputs a proof* $\pi$
- $\{0,1\} \leftarrow \mathsf{Verify}(crs, x, \pi)$. *A* DPT *algorithm that takes as input the common reference string, a statement* $x$ *and a proof* $\pi$*. It outputs a bit* $b \in \{0,1\}$*.*

*A* NIZK *is* complete *if for all* $(x,w) \in \mathsf{R}$ *and all* $crs \leftarrow \mathsf{Setup}(1^n)$ *it holds that:*

$$\Pr\left[\mathsf{Verify}(crs, x, \pi) = 1 \middle| \begin{matrix} crs \leftarrow \mathsf{Setup}(1^n) \\ \pi \leftarrow \mathsf{Prove}(crs, x, w) \end{matrix}\right] \geq 1 - \mathsf{negl}(n)$$

We now review the notion of security for a non-interactive zero-knowledge proof.

**Definition 26 (Zero-knowledge).** *A non-interactive zero knowledge proof* $\mathsf{NIZK} := (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *is* zero-knowledge *if there exist two* PPT *simulators* $(\mathsf{SetupSim}, \mathsf{ProveSim})$ *such that for every* PPT *adversary* $\mathcal{A}$*, then the following probability is negligible in* $n$*:*

$$\left| \Pr\left[(x,w) \in \mathsf{R} \wedge \middle| \begin{matrix} (crs) \leftarrow \mathsf{Setup}(1^n) \\ (x,w) \leftarrow \mathcal{A}(crs) \\ \pi \leftarrow \mathsf{Prove}(crs, w, x) \end{matrix}\right] - \Pr\left[(x,w) \in \mathsf{R} \wedge \middle| \begin{matrix} (crs, \tau) \leftarrow \mathsf{SetupSim}(1^n) \\ (x,w) \leftarrow \mathcal{A}(crs) \\ \pi \leftarrow \mathsf{ProveSim}(crs, \tau, x) \end{matrix}\right] \right|$$

**Definition 27 (Knowledge-soundness).** *A non-interactive zero knowledge proof* $\mathsf{NIZK} := (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *is* knowledge-sound *if there exists a* PPT *algorithm* $\mathsf{Ext}$ *such that for all adversaries* $\mathcal{A}$*, it holds that:*

$$\Pr\left[1 \leftarrow \mathsf{Verify}(crs, x, \pi) \middle| \begin{matrix} crs \leftarrow \mathsf{Setup}(1^n) \\ (x,\pi) \leftarrow \mathcal{A}(crs) \\ (x,w) \notin \mathsf{R} \end{matrix} \middle| w \leftarrow \mathsf{Ext}^{\mathcal{A}}(crs, x, \pi)\right] \leq \mathsf{negl}(n)$$

**Definition 28 (Simulation Extractability).** *A* $\mathsf{NIZK} := (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ $\mathsf{NIZK}$ *for a relation* $\mathsf{R}$ *is* simulation-extractable *if for any* PPT *adversary* $\mathcal{A}$*, there exists a* PPT *extractor* $\mathsf{Ext}^{\mathcal{A}}$ *such that* $\Pr[\mathsf{SE}_{\mathcal{A},\mathsf{NIZK}}(n) = 1] \leq \mathsf{negl}(n)$*, where the experiment* $\mathsf{SE}_{\mathcal{A},\mathsf{NIZK}}$ *is defined in Figure 14*

| $\mathsf{SE}_{\mathcal{A},\mathsf{NIZK}}(n)$ | $\mathsf{OProveSim}(x)$ |
|---|---|
| $1: \mathcal{Q} := \emptyset$ | $1: \pi \leftarrow \mathsf{ProveSim}(crs, \tau, x)$ |
| $2: (crs, \tau) \leftarrow \mathsf{SetupSim}(n)$ | $2: \mathcal{Q} := \mathcal{Q} \cup \{(x, \pi)\}$ |
| $3: (x, \pi) \leftarrow \mathcal{A}^{\mathsf{OProveSim}}(crs)$ | $3:$ **return** $\pi$ |
| $4: w \leftarrow \mathsf{Ext}^{\mathcal{A}}(x, \pi)$ | |
| $5: c_1 := \mathsf{Verify}(crs, x, \pi) = 1$ | |
| $6: c_2 := (x, \pi) \notin \mathcal{Q}$ | |
| $7: c_3 := (x, w) \notin \mathsf{R}$ | |
| $8:$ **return** $c_1 \wedge c_2 \wedge c_3$ | |

Fig. 14: Simulation extractability experiment of a $\mathsf{NIZK} := (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ NIZK.

### A.7 Identification Schemes

**Definition 29 (Canonical Identification Scheme).** *A canonical identification scheme* $\mathsf{ID}$ *is defined as a tuple of four algorithms* $\mathsf{ID} := (\mathsf{IGen}, \mathsf{P}, \mathsf{ChSet}, \mathsf{V})$.

- *The key generation algorithm* $\mathsf{IGen}$ *takes the system parameters pp as input and returns public and secret key* $(\mathsf{pk}, \mathsf{sk})$. *We assume that* $\mathsf{pk}$*defines a set of challenges, namely* $\mathsf{ChSet}$.
- *The prover algorithm* $\mathsf{P}$ *consists of two algorithms namely* $\mathsf{P}_1$ *and* $\mathsf{P}_2$*:*
  - $\mathsf{P}_1$ *takes as input the secret key* $\mathsf{sk}$*and returns a commitment* $R \in \mathcal{D}_r$*and and state* $St$.
  - $\mathsf{P}_2$ *takes as input the secret key* $\mathsf{sk}$*, a commitment* $R \in \mathcal{D}_r$*and, a challenge* $h \in \mathsf{ChSet}$ *and a state* $St$ *and returns a response* $r \in \mathcal{D}_r$*esp.*
- *The verifier algorithm* $\mathsf{V}$ *is a deterministic algorithm that takes the public key* $\mathsf{pk}$*and the conversation transcript as input and outputs* $1$ *(acceptance) or* $0$ *(rejection).*

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

**Definition 30 (Commitment-recoverable).** *An identification scheme* $\mathsf{ID}$ *is* Commitment-recoverable *if* $\mathsf{V}$ *first internally calls a function* $\mathsf{V}_0$ *which recomputes* $R_0 = \mathsf{V}_0(\mathsf{pk}, h, r)$ *and then outputs* $1$ *iff* $R = R_0$.

### A.8 Two-Party Signatures with Aggregatable Public Keys

**Definition 31 (Two-party signature scheme with aggregatable public keys ).** *A* two-party signature scheme with aggregatable public keys *is a tuple of* PPT *protocols and algorithms* $\mathsf{DS}_2 = (\mathsf{Setup}, \mathsf{KeyGen}, \Pi_{\mathsf{Sign}}, \mathsf{KAg}, \mathsf{Vrfy})$*, formally defined as:*

- $\mathsf{Setup}(n)$ : *is a* PPT *algorithm that on input a security parameter* $n$*, outputs public parameters* $pp$.
- $\mathsf{KeyGen}(pp)$ : *is a* PPT *algorithm that on input public parameters* $pp$*, outputs a key pair* $(\mathsf{sk}, \mathsf{vk})$.
- $\Pi_{\mathsf{Sign}\langle \mathsf{sk}_i, \mathsf{sk}_{1-i} \rangle}(\mathsf{vk}_0, \mathsf{vk}_1, m)$ : *is an interactive,* PPT *protocol that on input secret keys* $\mathsf{sk}_i$ *from party* $P_i$ *with* $i \in \{0, 1\}$ *and common values* $m \in \{0, 1\}^*$ *and* $\mathsf{vk}_0$*,* $\mathsf{vk}_1$ *outputs a signature* $\sigma$.
- $\mathsf{KAg}(\mathsf{vk}_0, \mathsf{vk}_1)$ : *is a* DPT *algorithm that on input two public keys* $\mathsf{vk}_0$*,* $\mathsf{vk}_1$*, outputs an aggregated public key* $\mathsf{avk}$.
- $\mathsf{Vrfy}(\mathsf{avk}, m, \sigma)$ : *is a* DPT *algorithm that on input public parameters* $pp$*, a public key* $\mathsf{avk}$*, a message* $m \in \{0, 1\}^*$ *and a signature* $\sigma$*, outputs a bit* $b$.

**Definition 32 (Completeness).** *A two-party signature with aggregatable public keys* $\mathsf{DS}_2$ *satisfies completeness, if for all key pairs* $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^n)$ *and messages* $m \in \{0, 1\}^*$*, the protocol* $\Pi_{\mathsf{Sign}\langle \mathsf{sk}_i, \mathsf{sk}_{1-i} \rangle}(\mathsf{vk}_0, \mathsf{vk}_1, m)$ *outputs a signature* $\sigma$ *to both parties* $P_0$*,* $P_1$ *such that* $\mathsf{Vrfy}(\mathsf{avk}, m, \sigma) = 1$ *where* $\mathsf{avk} := \mathsf{KAg}(\mathsf{vk}_0, \mathsf{vk}_1)$.

**Definition 33 (Two-party existential unforgeability security).** *A two-party, public key aggregatable signature scheme* $\mathsf{DS}_2$ *is unforgeable if for every* PPT *adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(n)$ *such that: for* $b \in \{0,1\}$*,* $\Pr[\mathsf{Sig\text{-}Forge}^{b}_{\mathcal{A},\mathsf{DS}_2}(n) = 1] \leq \mathsf{negl}(n)$*, where the experiment* $\mathsf{Sig\text{-}Forge}^{b}_{\mathcal{A},\mathsf{DS}_2}(n)$ *is defined as follows:*

| $\mathsf{Sig\text{-}Forge}^{b}_{\mathcal{A},\mathsf{DS}_2}(n)$ | $\mathsf{OSign}(m)$ |
|---|---|
| $\mathcal{Q} := \emptyset, pp \leftarrow \mathsf{Setup}(1^n)$ | $\sigma \leftarrow \Pi_{\mathsf{Sign}\langle\cdot,\mathsf{sk}_{1-b}\rangle}(m)$ |
| $(\mathsf{vk}_{1-b}, \mathsf{sk}_{1-b}) \leftarrow \mathsf{KeyGen}(pp)$ | $\mathcal{Q} := \mathcal{Q} \cup m$ |
| $\mathsf{vk}_b \leftarrow \mathcal{A}(pp, \mathsf{vk}_{1-b})$ | **return** $\sigma$ |
| $m, \sigma \leftarrow \mathcal{A}^{\mathsf{OSign}}(\mathsf{vk}_b, \mathsf{vk}_{1-b})$ | |
| $\mathsf{avk} := \mathsf{KAg}(\mathsf{vk}_b, \mathsf{vk}_{1-b})$ | |
| $b_0 := m \notin \mathcal{Q}$ | |
| $b_1 := \mathsf{Vrfy}(\mathsf{avk}, m, \sigma)$ | |
| **return** $b_0 \wedge b_1$ | |

### A.9 Verifiable Witness Encryption for a Relation

**Definition 34 (Verifiable witness encryption for a relation scheme).** *A* verifiable witness encryption for a relation scheme *is a tuple of algorithms* $\mathsf{VES} := (\mathsf{AKeyGen}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{VESign}, \mathsf{VEVerify}, \mathsf{Resolve})$ *defined w.r.t. a relation* R *and a signature scheme* DS *as follows:*

- $(c, \pi) \leftarrow \mathsf{EncR}((\mathsf{vk}, m), w)$*. A* PPT *algorithm that takes as input a tuple, comprising a verification key* vk *and a message* $m$*, and a witness* $w$*. It outputs the ciphertext tuple, containing a ciphertext* $c$ *and a proof* $\pi$*.*
- $\{0,1\} \leftarrow \mathsf{VfEncR}((c, \pi), (\mathsf{vk}, m), x)$*. A* DPT *algorithm that takes as input a tuple, comprising a ciphertext* $c$ *and a proof* $\pi$*, another tuple, comprising a verification key* vk *and a message* $m$*, and a public statement* $x$*. It outputs a bit* $b \in \{0,1\}$*.*
- $w \leftarrow \mathsf{DecR}(\sigma, (c, \pi))$*. A* DPT *algorithm that takes as input a signature* $\sigma$ *and a tuple comprimising a ciphertext* $c$ *and a proof* $\pi$*. It outputs a witness* $w$*.*

*A verifiable witness encryption for a relation scheme is* correct *if for all* $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^n)$*, all messages* $m \in \mathcal{M}$*, all statements and witness* $(x, w) \in \mathsf{R}$*, it holds that:*

1. $\Pr[\mathsf{VfEncR}(\mathsf{EncR}((\mathsf{vk}, m), w), x) = 1] = 1$
2. *If* $\mathsf{Vrfy}(\mathsf{vk}, m, \sigma) = 1$*, then:* $\Pr[(x, \mathsf{DecR}(\sigma, \mathsf{EncR}((\mathsf{vk}, m), w))) \in \mathsf{R}] = 1$

We now review the notions of one wayness and verifiability for a verifiable witness encryption for a relation scheme.

**Definition 35 (One Wayness).**
*A verifiable witness encryption for a relation scheme* $\mathsf{VWER} := (\mathsf{EncR}, \mathsf{VfEncR}, \mathsf{DecR})$ *achieves* one wayness *if for all* PPT *adversaries* $\mathcal{A}$ *there is a negligible function* $\mathsf{negl}(n)$ *such that:*

$$\Pr[\mathsf{ExpOW}_{\mathcal{A},\mathsf{VWER}}(n) = 1] \leq \mathsf{negl}(n)$$

*where the experiment* $\mathsf{ExpOW}_{\mathcal{A},\mathsf{VWER}}(n)$ *is defined as follows:*

$\mathsf{ExpOW}_{\mathcal{A},\mathsf{VWER}}(n)$

1 : $\mathcal{Q} := \emptyset$
2 : $(\mathsf{vk}, (\mathsf{vk}_0, \mathsf{sk}_0), (\mathsf{vk}_1, \mathsf{sk}_1)) \leftarrow \mathsf{KeyGen}(1^n)$
3 : $(x, w) \leftarrow \mathsf{GenR}(1^n)$
4 : $w^* \leftarrow \mathcal{A}^{\mathsf{OSign},\mathsf{OEncR}}(\mathsf{vk}, (\mathsf{vk}_0, \mathsf{sk}_0), x)$
5 : $b := ((x, w^*) \in \mathsf{R})$
6 : **return** $b$

$\mathsf{OSign}(m)$

1 : **if** $m \in \mathcal{Q}$ **abort**
2 : $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_1, m)$
3 : $\mathcal{Q} := \mathcal{Q} \cup m$
4 : **return** $\sigma$

$\mathsf{OEncR}(m)$

1 : **if** $m \in \mathcal{Q}$ **abort**
2 : $\mathcal{Q} := \mathcal{Q} \cup m$
3 : $(c, \pi) \leftarrow \mathsf{EncR}((\mathsf{vk}m), w)$
4 : **return** $(c, \pi)$

**Definition 36 (Verifiability).** *A verifiable witness encryption for a relation scheme* $\mathsf{VWER} := (\mathsf{EncR}, \mathsf{VfEncR}, \mathsf{DecR})$ *achieves* verifiability *if for all* PPT *adversaries* $\mathcal{A}$ *there is a negligible function* $\mathsf{negl}(n)$ *such that:*

$$\Pr[\mathsf{ExpVer}_{\mathcal{A},\mathsf{VWER}}(n) = 1] \leq \mathsf{negl}(n)$$

*where the experiment* $\mathsf{ExpVer}_{\mathcal{A},\mathsf{VWER}}(n)$ *is defined as follows:*

$\mathsf{ExpVer}_{\mathcal{A},\mathsf{VWER}}(n)$

1 : $(m, \mathsf{vk}, \sigma, c, \pi, x) \leftarrow \mathcal{A}(n)$
2 : $w^* \leftarrow \mathsf{DecR}(\sigma, (c, \pi))$
3 : $b_0 := (\mathsf{EncR}(((c, \pi)), (\mathsf{vk}, m), x) = 1)$
4 : $b_1 := (\mathsf{Vrfy}(\mathsf{vk}, m, \sigma) = 1)$
5 : $b_2 := ((x, w^*) \notin \mathsf{R})$
6 : **return** $b_0 \wedge b_1 \wedge b_2$

**Definition 37 (Zero-knowledge).** *A verifiable witness encryption for a relation scheme* $\mathsf{VWER} := (\mathsf{EncR}, \mathsf{VfEncR}, \mathsf{DecR})$ *achieves* zero-knowledge *if for all* PPT *adversaries* $\mathcal{A}$ *there exists a simulator algorithm* $\mathsf{EncRSim}$ *such that:*

$$\Pr[\mathsf{ZK}_{\mathcal{A},\mathsf{VWER}}(n) = 1] \leq \frac{1}{2} + \mathsf{negl}(n)$$

*where the experiment* ZK *is defined as follows:*

$\mathsf{ZK}_{\mathcal{A},\mathsf{VWER}}$

1 : $b \leftarrow_{\$} \{0, 1\}$
2 : $(\mathsf{vk}, (\mathsf{vk}_0, \mathsf{sk}_0), (\mathsf{vk}_1, \mathsf{sk}_1)) \leftarrow \mathsf{KeyGen}(1^n)$
3 : $(x, w) \leftarrow \mathsf{GenR}(1^n)$
4 : $b' \leftarrow \mathcal{A}^{\mathsf{OEncR}}(\mathsf{vk}, (\mathsf{vk}_0, \mathsf{sk}_0))$
5 : **return** $b = b'$

$\mathsf{OEncR}(m)$

1 : **if** $b = 0$ **then**
2 : $(c, \pi) \leftarrow \mathsf{EncR}((\mathsf{vk}, m), w)$
3 : **else**
4 : $(c, \pi) \leftarrow \mathsf{EncRSim}((\mathsf{vk}, m))$
5 : **return** $(c, \pi)$

### A.10 Instantiation of Verifiable Witness Encryption for a Relation

$\mathsf{EncR}((\mathsf{vk}, m), w)$

$\mathcal{S}_{\text{op}} := \mathcal{S}_{\text{unop}} := \emptyset$
**for** $i \in [1, \gamma]$ :
  $r_i \leftarrow_{\$} \mathbb{Z}_q; R_i := g^{r_i}$
  $c_i := \mathsf{WES.Enc}((\mathsf{vk}, m), r_i, r'_i)$
    // where $r'_i$ are the random coins used in $\mathsf{WES.Enc}$
$(b_1, \ldots, b_\gamma) := \mathsf{H}((c_i, R_i)_{i \in [1,\gamma]})$
**for** $i \in [1, \gamma]$ :
  **if** $b_i = 1$ **then**
    $\mathcal{S}_{\text{op}} := \mathcal{S}_{\text{op}} \cup \{(i, r_i, r'_i)\}$
  **if** $b_i = 0$ **then**
    $s_i := r_i + w$
    $\mathcal{S}_{\text{unop}} := \mathcal{S}_{\text{unop}} \cup \{(i, s_i, c_i)\}$
**return** $c := \{c_i\}_{i \in [1,\gamma]}, \pi := \left\{\mathcal{S}_{\text{op}}, \mathcal{S}_{\text{unop}}, \{R_i\}_{i \in [1,\gamma]}\right\}$

$\mathsf{DecR}(\sigma, c, \pi)$

$\{c_i\}_{i \in [1,\gamma]} := c; \left\{\mathcal{S}_{\text{op}}, \mathcal{S}_{\text{unop}}, \{R_i\}_{i \in [1,\gamma]}\right\} := \pi$
**for** $(i, s_i, c_i) \in \mathcal{S}_{\text{unop}}$ :
  $r_i := \mathsf{WES.Dec}(\sigma, c_i)$
  // There exists at least one $r_a$ s.t. $R_a = g^{r_a}$
$w := s_a - r_a$
**return** $w$

$\mathsf{VfEncR}(c, \pi, (\mathsf{vk}, m), x)$

$\{c_i\}_{i \in [1,\gamma]} := c; \left\{\mathcal{S}_{\text{op}}, \mathcal{S}_{\text{unop}}, \{R_i\}_{i \in [1,\gamma]}\right\} := \pi$
$(b_1, \ldots, b_\gamma) := \mathsf{H}((c_i, R_i)_{i \in [1,\gamma]})$
**for** $i \in [1, \gamma]$ :
  **if** $b_i = 1$ **then**
    **assert** $(i, r_i, r'_i) \in \mathcal{S}_{\text{op}}$
    **assert** $c_i = \mathsf{WES.Enc}((\mathsf{vk}, m), r_i; r'_i)$
  **if** $b_i = 0$ **then**
    **assert** $(i, s_i, c'_i) \in \mathcal{S}_{\text{unop}}$
    **assert** $g^{s_i} = R_i * x$
**return** 1

**Theorem 3.** *If* WES *is IND-CPA, then the construction of [27] satisfies* zero-knowledge

*Proof.* We build a simulator that would satisfy the zero-knowledge property step by step making indistinguishable changes between simulators.

**Simulator** $\mathsf{Sim}_0$: This simulator runs the exact same code of $\mathsf{EncR}$ but programming the random oracle.

*Claim.* For all PPT distinguisher $\mathcal{A}$, it holds that $|\Pr[\mathcal{A}^{\mathsf{EncR}}(\mathsf{vk})] - \Pr[\mathcal{A}^{\mathsf{Sim}_0}(\mathsf{vk})]| \leq \mathsf{negl}(n)$

**Simulator** $\mathsf{Sim}_1$: This simulator runs the following code:

$\mathsf{Sim}_1((\mathsf{vk}, m))$

$\mathcal{S}_{\text{op}} := \mathcal{S}_{\text{unop}} := \emptyset$
$(b_1, \ldots, b_\gamma) \leftarrow_\$ \{0,1\}^\gamma$
**for** $i \in [1, \gamma]$ :
    $r_i \leftarrow_\$ \mathbb{Z}_q$
    **if** $b_i = 1$ :
        $c_i := \mathsf{WES.Enc}((\mathsf{vk}, m), r_i, r_i')$
        $R_i := g^{m_0}$
        $\mathcal{S}_{\text{op}} := \mathcal{S}_{\text{op}} \cup \{(i, r_i, r_i')\}$
    **if** $b_i = 0$ :
        $c_i := \mathsf{Enc}(m, r_i, r_i')$
        $s_i \leftarrow_\$ \mathbb{Z}_q$
        $R_i := g^{s_i} * x^{-1}$
        $\mathcal{S}_{\text{unop}} := \mathcal{S}_{\text{unop}} \cup \{(i, s_i, c_i)\}$
$\mathsf{H}((c_i, R_i)_{i \in [1,\gamma]}) := (b_1, \ldots, b_\gamma)$
**return** $c := \{c_i\}_{i \in [1,\gamma]}, \pi := \left\{\mathcal{S}_{\text{op}}, \mathcal{S}_{\text{unop}}, \{R_i\}_{i \in [1,\gamma]}\right\}$

*Claim.* If the non-interactive schnorr-protocol is zero-knowledge, then for all PPT distinguisher $\mathcal{A}$, it holds that $|\Pr[\mathcal{A}^{\mathsf{Sim}_0}(\mathsf{vk})] - \Pr[\mathcal{A}^{\mathsf{Sim}_1}(\mathsf{vk})]| \leq \mathsf{negl}(n)$

*Proof.* If there exists a distinguisher $\mathcal{A}$ that is able to distinguish between $\mathsf{Sim}_0$ and $\mathsf{Sim}_1$, then we can create a PPT adversary $\mathcal{B}$ that breaks the zero-knowledge of the non-interactive schnorr-protocol.

Whenever $\mathcal{B}$ receives queries for the simulator from $\mathcal{A}$, the adversary $\mathcal{B}$ runs $\mathsf{Sim}_1$ until right before it has to compute $s_i$ and $R_i$. Then, $\mathcal{B}$ queries the oracle from the challenger of the zero-knowledge of the non-interactive schnorr-protocol. This oracle replies with a proof $\pi := (s_i, R_i)$ such that $g^{s_i} = R_i * x$. The adversary uses these $s_i$ and $R_i$ and completes the run of the simulator. Finally, $\mathcal{A}$ sends their guess $b'$ and $\mathcal{B}$ forwards this guess to the challenger.

If the challenger of the zero-knowledge of the non-interactive schnorr-protocol chose $b = 0$, then the execution of $\mathcal{B}$ is the same as $\mathsf{Sim}_0$. Otherwise, it is the same as $\mathsf{Sim}_1$. If $\mathcal{A}$ distinguishes correctly $\mathsf{Sim}_0$ and $\mathsf{Sim}_1$ with non-negligible probability, then $\mathcal{B}$ breaks the zero-knowledge of the non-interactive schnorr-protocol with non-negligible probability.

**Simulator** $\mathsf{Sim}_2$: This simulator runs the following code:

$\mathsf{Sim}_2((\mathsf{vk}, m))$

$\mathcal{S}_{\text{op}} := \mathcal{S}_{\text{unop}} := \emptyset$
$(b_1, \ldots, b_\gamma) \leftarrow_\$ \{0,1\}^\gamma$
$n := 0$
$\mathit{threshold} := 1$
**for** $i \in [1, \gamma]$ :
    $r_i \leftarrow_\$ \mathbb{Z}_q$
    **if** $b_i = 1$ :
        $c_i := \mathsf{WES.Enc}((\mathsf{vk}, m), r_i, r_i')$
        $R_i := g^{r_i}$
        $\mathcal{S}_{\text{op}} := \mathcal{S}_{\text{op}} \cup \{(i, r_i, r_i')\}$
    **if** $b_i = 0$ :
        **if** $n < \mathit{threshold}$ :
            $c_i := \mathsf{WES.Enc}((\mathsf{vk}, m), 0, r_i')$
            $n := n + 1$
        **else** :
            $c_i := \mathsf{WES.Enc}((\mathsf{vk}, m), r_i, r_i')$
        $s_i \leftarrow_\$ \mathbb{Z}_q$
        $R_i := g^{s_i} * x^{-1}$
        $\mathcal{S}_{\text{unop}} := \mathcal{S}_{\text{unop}} \cup \{(i, s_i, c_i)\}$
$\mathsf{H}((c_i, R_i)_{i \in [1,\gamma]}) := (b_1, \ldots, b_\gamma)$
**return** $c := \{c_i\}_{i \in [1,\gamma]}, \pi := \left\{\mathcal{S}_{\text{op}}, \mathcal{S}_{\text{unop}}, \{R_i\}_{i \in [1,\gamma]}\right\}$

*Claim.* If $\mathsf{WES}$ is IND-CPA, then for all $\mathsf{PPT}$ distinguisher $\mathcal{A}$, it holds that $|\Pr[\mathcal{A}^{\mathsf{Sim}_1}(\mathsf{vk})] - \Pr[\mathcal{A}^{\mathsf{Sim}_2}(\mathsf{vk})]| \leq \mathsf{negl}(n)$

*Proof.* If there exists a distinguisher $\mathcal{A}$ that is able to distinguish between $\mathsf{Sim}_1$ and $\mathsf{Sim}_2$, then we can create a $\mathsf{PPT}$ adversary $\mathcal{B}$ that breaks the IND-CPA of the $\mathsf{WES}$.

Whenever $\mathcal{B}$ receives queries for the simulator from $\mathcal{A}$, the adversary $\mathcal{B}$ runs $\mathsf{Sim}_2$ until right before it has to compute $c_i$ in the conditional branch of $n < 1$. Then, $\mathcal{B}$ queries the encrypt oracle from the challenger of the IND-CPA on inputs $m_0 := r_i$ and $m_1 := 0$. This oracle replies with a ciphertext $c_i$. The adversary $\mathcal{B}$ uses this $c_i$ and completes the run of the simulator. Finally, $\mathcal{A}$ sends their guess $b'$ and $\mathcal{B}$ forwards this guess to the challenger.

If the challenger of the IND-CPA of the $\mathsf{WES}$ chose $b = 0$, then the execution of $\mathcal{B}$ is the same as $\mathsf{Sim}_1$. Otherwise, it is the same as $\mathsf{Sim}_2$. If $\mathcal{A}$ distinguishes correctly $\mathsf{Sim}_1$ and $\mathsf{Sim}_2$ with non-negligible probability, then $\mathcal{B}$ breaks the IND-CPA of $\mathsf{WES}$ with non-negligible probability.

**Simulators** $\mathsf{Sim}_{3,\ldots,\theta}$ where $\theta$ is the number of 0 that appear in the programmed output of the random oracle $(b_1, \ldots, b_\gamma)$. These simulators are identical to $\mathsf{Sim}_2$ except that in each consecutive simulator, the *threshold* increases by one. The proofs are also equivalent to the proof for $\mathsf{Sim}_2$. In these proofs, we prove that the simulator is indistinguishable from its predecesor.

Once we arrive at the final simulator, all the ciphertexts $c_i$ for the cases where $b_i = 0$ are simulated and there is no more information that helps the adversary to distinguish between the simulator and the real execution of $\mathsf{EncR}$. Therefore, the construction satisfies *zero-knowledge*.

## B Extended Related Work

**Security definitions and constructions for 1P-AS.** Single-party adaptor signatures (1P-AS) were first informally mentioned in [28] and later formalized by Aumayr et al. [1]. 1P-AS have been used in many blockchain applications such as payment channel hubs [3, 5], contingent payments [12, 11] and fair data exchange [10] to name just a few. This has sparked the research interest on it and recent works have extended the security and privacy notions for 1P-AS [21, 14] as well as the relations among them. Similarly, several constructions of 1P-AS have been proposed for different signatures schemes including identification schemes [16], dichotomic signatures [14], lattice-based signature [29], isogeny-based signature [30]. We also have constructions for random-self-reducible relations [21], and from any NP relation [31, 32].

**Security definitions and constructions for 2P-AS.** The only existing definitions that consider two-party AS are given in [16] where the presigning protocol is defined in a fully *symmetric* manner. This symmetry reflects in (i) that the presigning protocol expects the same input from both protocol users; and (ii) that the presigning protocol provides the same output (the presignature $\hat{\sigma}$ w.r.t. shared key) to both protocol users. Condition (ii) can (due to the impossibility result from [18]) only be met in honest-majority settings. For the two-party setting, this means that a presigning protocol that delivers the presignature to both parties can only be realized provided that both protocol users are (semi-)honest – an assumption that does not reflect the attacker models in most blockchain applications (such as atomic swaps or coin mixing). Otherwise, it needs to be assumed that only a single party may obtain $\hat{\sigma}$, an aspect that is not considered in the security definitions given in [16]. For this reason, the definitions provided in [16] do not serve for analyzing AS-based protocols, such as atomic swaps, whose security critically depends on the order of output delivery. Furthermore, because of condition (i), the impossibility result for 1P-AS regarding unique signatures immediately applies also to the two-party symmetric case (as we also show in Section D). Consequently, such signatures cannot be instantiated for unique signature schemes.

Bursuc and Mauw [33] introduced the notion of *weak adaptor signatures*, where the presigning protocol takes the cryptographic secret as input. For this setting, they provide an ECDSA-based construction that simplifies the construction of adaptor signatures in the standard setting. While this hints at the significance of input asymmetry for AS, the authors characterize security in a symbolic model of cryptography, which idealizes the protocol interactions of the presigning protocol. It is unclear whether this idealized security notion soundly abstracts the security of the ECDSA-based construction or is generally achievable.

**Beyond 2P-AS.** Recently, Baecker et al. [34] defined threshold AS and provided an instantiation based on the Schnorr signature scheme. Interestingly, they acknowledge the impossibility of a multiparty fair exchange protocol in a dishonest majority setting, and the security definition sticks to the honest majority setting. The proposed security definition is also symmetric and does not consider the security model with aborts. Hence, it would be interesting to study if the implications due to the asymmetries in 2P-AISO-AS also carry over to the threshold AS setting.

## C Extended Testbed Explanation

**Implementation Optimizations.** In Section 6, we measure the execution time of all the algorithms in the construction from Figure 10 and Figure 9, which constitute the core of the preimage sale protocol, as shown in Figure 12 without the application-specific steps. The performance evaluation is summarized in Figure 11. These measurements include a slight time optimization for the construction in Figure 10: we assume that the *Adaptor* is stateful. From this assumption, in $\mathsf{PreSign}_A$ we do not need to encrypt the signature $\sigma_A$ and prove the correctness of this encryption. Instead, the *Adaptor* can store the signature in its internal state to use it later when running $\mathsf{Adapt}$. This optimization corresponds to removing the parts from Figure 10 highlighted in gray. As a result, the implementation reduces the communication overhead and computation time slightly. We remark that the costly computation lies in using the $\mathsf{VWER}$, which is still present.

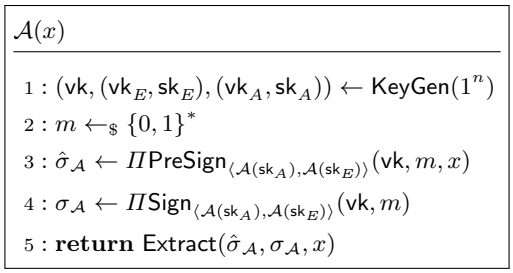



Fig. 15: **Impossibility of building 2P-AS and 2P-SO-AS from unique signatures.** Adversary breaking hardness of a relation if a two-party adaptor signature (or 2P-SO-AS) scheme with underlying unique signatures existed.

These results show that the BLS-based instantiation of 2P-AISO-AS is practical. Even though it relies on computationally taxing building blocks such as VWER, the most time-consuming algorithms take less than half a second on commodity hardware.

**Implementation Details.** In out implementation, we instantiated DS using BLS signature, VWER using the construction given in [12]. For the unoptimized construction, we implemented PKE and NIZK jointly as a verifiably encrypted signature (VES) scheme based on BLS given by Boneh et al. [35]. Our implementation is written in Rust using crates *bls12_381*, *ff* and *group*. The implementation of the ZKCP protocol is based on the implementation given in [11], which implements primitives on its own using rust crates *elliptic-curve* and *k256* for implementing arithmetic over the curve Secp256k1 of bitcoin. The source code is available at [24]. In our tests, we have not measured time spent waiting for the network to deliver the messages nor time spent waiting for the signatures to be published on the ledger after being submitted.

## D Impossibility Of Building 2P-AS And 2P-SO-AS From Unique Signatures

In this section we present the impossibility of building 2P-AS and 2P-SO-AS from unique signatures. The formal proof follows the one of [16] although extended to 2P-AS and 2P-SO-AS. In Figure 15 we present the reduction that proves the impossibility result for both 2P-AS and 2P-SO-AS. The reduction follows the same intuition described in Section 2. The only difference is that, in this case, the adversary must generate keys for the two-party setting and, instead of running PreSign and Sign algorithms, the adversary runs the protocols for PreSign and Sign interactively with themself. In this case, since the adversary is running these protocols interactively with themself, the solitary output of 2P-SO-AS does not make a difference in the reduction: the output delivery is guaranteed to only one party, but in this reduction the adversary plays both parties. Thus, the same reduction works exactly the same for both 2P-AS and 2P-SO-AS.

## E Counterexample Motivating Necessity Of Oracle OAdapt In 2P-AISO-AS-WUnf

In Section 4 we defined *weak unforgeability* for 2P-AISO-AS in Figure 6. In this definition, we emphasized that, due to the API of 2P-AISO-AS, the security game only allows the adversary to query the presign oragle $\mathsf{OPreSign}_A$ on honestly generated statements. These statements are generated through the oracle ONewStatement for the challenger to know the corresponding witnesses. As a consequence, the adversary would not be able to query $\mathsf{OPreSign}_A$ on statements whose witnesses are known to the adversary. Therefore, the adversary would not be able to adapt presignatures on their own. To amend this, we included the oracle OAdapt. This oracle allows the adversary to obtain adapted signatures as intended.

$\mathcal{A}^{\mathcal{O}}(\mathsf{vk}, \mathsf{vk}_E, \mathsf{sk}_E, \mathsf{vk}_A)$

$m \leftarrow_{\$} \{0,1\}^*$
$x \leftarrow \mathsf{ONewStatement}()$
$pm \leftarrow \mathsf{OPreSign}_A(m, x)$
$\hat{\sigma} := \mathsf{PreSign}_E(\mathsf{sk}_E, m, x, pm)$
$\sigma \leftarrow \mathsf{OAdapt}(\hat{\sigma}, x, m)$
$w := \mathsf{Extract}(\mathsf{vk}, \sigma, \hat{\sigma}, x)$
$(pm', ct) := pm$
$\mathsf{sk}_A := \mathsf{Dec}(w, ct)$
$m^* \leftarrow_{\$} \{0,1\}^*$
$\sigma^* \leftarrow \Pi\mathsf{Sign}_{\langle \mathcal{A}(\mathsf{sk}_A), \mathcal{A}(\mathsf{sk}_E) \rangle}(\mathsf{vk}, m^*)$
**return** $(m^*, \sigma^*)$

Fig. 16: **Attack for 2P-AISO-AS-WUnf of the construction $\Theta'$:** The adversary $\mathcal{A}$ plays against the 2P-AISO-AS-WUnf game. The available oracles include OAdapt and the attack hinges on this fact.

To further motivate the inclusion of the OAdapt oracle we present an insecure 2P-AISO-AS construction $\Theta'$. The construction assumes and extends a secure 2P-AISO-AS construction $\Theta$ where the underlying signature scheme is randomized and the hard relation R is the discrete logarithm. All the algorithms of $\Theta'$ are the same as those of $\Theta$ with the exception of $\mathsf{PreSign}_A$. The algorithm $\Theta'.\mathsf{PreSign}_A$ first calls the corresponding $\Theta.\mathsf{PreSign}_A$ to obtain $pm$, and then uses ElGamal to encrypt the signing key $\mathsf{sk}_A$ into the ciphertext $ct$ using $x$ as the encryption key. The output of $\Theta'.\mathsf{PreSign}_A$ is the tuple $(pm, ct)$. The construction $\Theta'$ is clearly insecure as being able to observe an adapted signature, would leak the decryption key $w$ that allows to obtain the signing key $\mathsf{sk}_A$ of the *Adaptor*. Indeed, in Figure 16 we show an attack on 2P-AISO-AS-WUnf proving this construction insecure. The attack crucially relying on the oracle OAdapt. If the oracle OAdapt was not available, then the attack of Figure 16 would not apply anymore and $\Theta'$ would be considered secure even though it is not suitable for applications.

## F Security Analysis Of BLS-BLS

**Lemma 1 (Figure 9 satisfies correctness).**

*Proof.* First, notice that $(x, w) = ((\mathsf{vk}_\mathsf{R}, m_\mathsf{R}), \sigma_\mathsf{R})$ such that $\mathsf{Vrfy}(\mathsf{vk}_\mathsf{R}, m_\mathsf{R}, \sigma_\mathsf{R}) = 1$ where $(x, w) \leftarrow \mathsf{GenR}(n)$. Notice that $\mathsf{vk}_\mathsf{R}$ has an associated signing key denoted $\mathsf{sk}_\mathsf{R}$ and that, by the definition of BLS signatures, $\sigma_\mathsf{R} = \mathsf{H}(m_\mathsf{R})^{\mathsf{sk}_\mathsf{R}}$. Then, by running $pm \leftarrow \mathsf{PreSign}_A(\mathsf{sk}_A, w, m, x)$ we obtain:

$$pm = \mathrm{Agg}\left(\sigma_A, w\right) = \mathrm{Agg}\left(\sigma_A, \sigma_\mathsf{R}\right), \quad \text{where } \sigma_A \leftarrow \mathsf{Sign}(\mathsf{sk}_A, m).$$

Next, by running $\hat{\sigma} \leftarrow \mathsf{PreSign}_E(\mathsf{sk}_E, m, x, pm)$ we obtain

$$\hat{\sigma} = \mathrm{Agg}\left(\sigma_E, pm\right) = \mathrm{Agg}\left(\sigma_E, \sigma_A, \sigma_\mathsf{R}\right), \quad \text{where } \sigma_E \leftarrow \mathsf{Sign}(\mathsf{sk}_E, m).$$

By running $\sigma := \mathsf{Adapt}(\mathsf{sk}_A, \mathsf{vk}, \hat{\sigma}, w, m)$ we obtain:

$$\sigma = \mathrm{Agg}\left(\hat{\sigma}, w^{-1}\right) = \mathrm{Agg}\left(\sigma_E, \sigma_A, \sigma_\mathsf{R}, \sigma_\mathsf{R}^{-1}\right) = \mathrm{Agg}\left(\sigma_E, \sigma_A\right).$$

Finally, running $w := \mathsf{Extract}(\mathsf{vk}, \sigma, \hat{\sigma}, x)$ we obtain:

$$w = \mathrm{Agg}\left(\hat{\sigma}, \sigma^{-1}\right) = \mathrm{Agg}\left(\sigma_E, \sigma_A, \sigma_{\mathsf{R}}, (\mathrm{Agg}\left(\sigma_E, \sigma_A\right))^{-1}\right) = \sigma_{\mathsf{R}}.$$

First, we observe that trivially $(x, w) = ((\mathsf{vk}_{\mathsf{R}}, m_{\mathsf{R}}), \sigma_{\mathsf{R}}) \in \mathsf{R}$ due to the correctness of BLS signatures and the fact that $\sigma_{\mathsf{R}} = \mathsf{Sign}(\mathsf{sk}_{\mathsf{R}}, m_{\mathsf{R}})$.

Next, we observe that $\mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma}) = \mathsf{AggVerify}((\mathsf{vk}, \mathsf{vk}_{\mathsf{R}}), (m, m_{\mathsf{R}}), \hat{\sigma})$ given that $x = (\mathsf{vk}_{\mathsf{R}}, m_{\mathsf{R}})$ and that $\mathsf{vk} = \mathrm{Agg}\left(\mathsf{vk}_A, \mathsf{vk}_E\right)$. By the definition of $\mathsf{AggVerify}$ given in [36] (and included in Figure 9), this algorithm checks whether the equality $\mathsf{e}(\hat{\sigma}, g_2) = \mathsf{e}(\mathsf{H}(m), \mathsf{vk}) \cdot \mathsf{e}(\mathsf{H}(m_{\mathsf{R}}), \mathsf{vk}_{\mathsf{R}})$ holds. Indeed:

$$\begin{gathered}
\mathsf{e}(\hat{\sigma}, g_2) = \mathsf{e}(\mathrm{Agg}\left(\sigma_E, \sigma_A, \sigma_{\mathsf{R}}\right), g_2). \\
\text{By the definition of the BLS } \mathsf{Sign} \text{ algorithm: } \mathsf{e}(\hat{\sigma}, g_2) = \\
\mathsf{e}(\mathrm{Agg}\left(\mathsf{H}(m)^{\mathsf{sk}_E}, \mathsf{H}(m)^{\mathsf{sk}_A}, \mathsf{H}(m_{\mathsf{R}})^{\mathsf{sk}_{\mathsf{R}}}\right), g_2) = \\
\mathsf{e}(\mathrm{Agg}\left(\mathsf{H}(m)^{\mathsf{sk}_E + \mathsf{sk}_A}, \mathsf{H}(m_{\mathsf{R}})^{\mathsf{sk}_{\mathsf{R}}}\right), g_2) = \\
\mathsf{e}(\mathsf{H}(m)^{\mathsf{sk}_E + \mathsf{sk}_A}, g_2) \cdot \mathsf{e}(\mathsf{H}(m_{\mathsf{R}})^{\mathsf{sk}_{\mathsf{R}}}, g_2) = \\
\mathsf{e}(\mathsf{H}(m), g_2^{\mathsf{sk}_E + \mathsf{sk}_A}) \cdot \mathsf{e}(\mathsf{H}(m_{\mathsf{R}}), g_2^{\mathsf{sk}_{\mathsf{R}}}) = \\
\mathsf{e}(\mathsf{H}(m), \mathsf{vk}) \cdot \mathsf{e}(\mathsf{H}(m_{\mathsf{R}}), \mathsf{vk}_{\mathsf{R}}).
\end{gathered}$$

Thus, $\mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma}) = 1$.

Finally, we observe that $\mathsf{Vrfy}(\mathsf{vk}, m, \sigma)$ checks whether the equality $\mathsf{e}(\sigma, g_2) = \mathsf{e}(\mathsf{H}(m), \mathsf{vk})$ Holds. Indeed:

$$\begin{gathered}
\mathsf{e}(\sigma, g_2) = \\
\mathsf{e}(\mathrm{Agg}\left(\sigma_A, \sigma_E\right), g_2) = \\
\mathsf{e}(\mathrm{Agg}\left(\mathsf{H}(m)^{\mathsf{sk}_A}, \mathsf{H}(m)^{\mathsf{sk}_E}\right), g_2) = \\
\mathsf{e}(\mathsf{H}(m)^{\mathsf{sk}_A + \mathsf{sk}_E}, g_2) = \\
\mathsf{e}(\mathsf{H}(m), \mathsf{vk}).
\end{gathered}$$

Thus, $\mathsf{Vrfy}(\mathsf{vk}, m, \sigma) = 1$.

**Lemma 2 (Figure 9 satisfies computational preverify soundness).**

*Proof.* Assume that there exists a PPT adversary $\mathcal{A}$ such that for every $n \in \mathbb{N}$ and polynomially bounded $x \notin L_{\mathsf{R}}$, it holds that:

$$\Pr\left[\mathsf{PreVrfy}(\mathsf{vk}, m, \hat{\sigma}, x) = 1 \;\middle|\; \begin{array}{c} (\mathsf{vk}, (\mathsf{vk}_E, \mathsf{sk}_E), (\mathsf{vk}_A, \mathsf{sk}_A)) \leftarrow \mathsf{KeyGen}(1^n) \\ (m, \hat{\sigma}) \leftarrow \mathcal{A}(\mathsf{vk}, (\mathsf{vk}_E, \mathsf{sk}_E), (\mathsf{vk}_A, \mathsf{sk}_A)) \end{array}\right] > \mathsf{negl}(n),$$

Then, since $x \notin L_{\mathsf{R}}$ and $x = (\mathsf{vk}_{\mathsf{R}}, m_{\mathsf{R}})$, it must be the case that there does not exist a witness $w = \sigma_{\mathsf{R}}$ such that $((\mathsf{vk}_{\mathsf{R}}, m_{\mathsf{R}}), w) \in \mathsf{R}$. In other words, for $\mathsf{vk}_{\mathsf{R}}, m_{\mathsf{R}}$ there does not exist a signature $\sigma_{\mathsf{R}}$ such that $\mathsf{Vrfy}(\mathsf{vk}_{\mathsf{R}}, m_{\mathsf{R}}, \sigma_{\mathsf{R}}) = 1$. Therefore, for $(\mathsf{vk}_{\mathsf{R}}, m_{\mathsf{R}})$, no signature $\sigma_{\mathsf{R}}$ satisfies $\mathsf{e}(\sigma_{\mathsf{R}}, g_2) = \mathsf{e}(\mathsf{H}(m_{\mathsf{R}}), \mathsf{vk}_{\mathsf{R}})$. The adversary $\mathcal{A}$ on input a tuple of honestly generated keys $(\mathsf{vk}, (\mathsf{vk}_A, \mathsf{sk}_A), (\mathsf{vk}_E, \mathsf{sk}_E))$, returns $(m, \hat{\sigma})$ such that $\mathsf{PreVrfy}(\mathsf{vk}, m, \hat{\sigma}, x) = 1$. The preverify algorithm checks whether $\mathsf{AggVerify}((\mathsf{vk}, \mathsf{vk}_{\mathsf{R}}), (m, m_{\mathsf{R}}), \hat{\sigma}) = 1$, which internally checks whether the equality $\mathsf{e}(\hat{\sigma}, g_2) = \mathsf{e}(\mathsf{H}(m), \mathsf{vk}) \cdot \mathsf{e}(\mathsf{H}(m_{\mathsf{R}}), \mathsf{vk} \cdot \mathsf{vk}_{\mathsf{R}})$ holds. Without loss of generality, let $\hat{\sigma}$ be the product of two group elements $w_1 \cdot w_2$. By using the bilinear properties of the pairing, the $\mathsf{AggVerify}$ algorithm checks whether the equivalent equality $\mathsf{e}(w_1 \cdot w_2, g_2) = \mathsf{e}(w_1, g_2) \cdot \mathsf{e}(w_2, g_2) = \mathsf{e}(\mathsf{H}(m), \mathsf{vk}) \cdot \mathsf{e}(\mathsf{H}(m_{\mathsf{R}}), \mathsf{vk}_{\mathsf{R}})$ holds. However, this equality can only hold if there exists $w_2$ such that $\mathsf{e}(w_2, g_2) = \mathsf{e}(\mathsf{H}(m_{\mathsf{R}}), \mathsf{vk}_{\mathsf{R}})$. In that case $((\mathsf{vk}_{\mathsf{R}}, m_{\mathsf{R}}), w_2) \in \mathsf{R}$ which contradicts $x \notin \mathsf{R}$.

**Lemma 3 (Figure 9 satisfies adaptability).**

*Proof.* Given any message $m$, any tuple of honestly generated keys $(\mathsf{vk}, (\mathsf{sk}_A, \mathsf{vk}_A), (\mathsf{sk}_E, \mathsf{vk}_E))$, any pair $(x, w) \in \mathsf{R}$ and any presignature $\hat{\sigma}$ such that $\mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma}) = 1$, we want to prove that $\mathsf{Vrfy}(\mathsf{vk}, m, \mathsf{Adapt}(\mathsf{sk}_A, \mathsf{vk}, \hat{\sigma}, w, m)) = 1$. Let $\sigma$ be the output of $\mathsf{Adapt}(\mathsf{sk}_A, \mathsf{vk}, \hat{\sigma}, w, m)$. By the definition of $\mathsf{Vrfy}$ we need to prove that the equality $\mathsf{e}(\sigma, g_2) = \mathsf{e}(\mathsf{H}(m), \mathsf{vk})$ holds. Given that $x = (\mathsf{vk}_{\mathsf{R}}, m_{\mathsf{R}})$

and $w = \sigma_{\mathsf{R}}$, and given the definition of the $\mathsf{Adapt}$ algorithm, we get that $\sigma = \hat{\sigma} \cdot \sigma_{\mathsf{R}}^{-1}$. Therefore $\mathsf{e}(\sigma, g_2) = \mathsf{e}(\hat{\sigma} \cdot \sigma_{\mathsf{R}}^{-1}, g_2) = \mathsf{e}(\hat{\sigma}, g_2) \cdot \mathsf{e}(\sigma_{\mathsf{R}}^{-1}, g_2)$. By the fact that $\mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma}) = 1$, we know that $\mathsf{e}(\hat{\sigma}, g_2) = \mathsf{e}(\mathsf{H}(m), \mathsf{vk}) \cdot \mathsf{e}(\mathsf{H}(m_{\mathsf{R}}), \mathsf{vk}_{\mathsf{R}})$. Therefore $\mathsf{e}(\sigma, g_2) = \mathsf{e}(\mathsf{H}(m), \mathsf{vk}) \cdot \mathsf{e}(\mathsf{H}(m_{\mathsf{R}}), \mathsf{vk}_{\mathsf{R}}) \cdot \mathsf{e}(\sigma_{\mathsf{R}}^{-1}, g_2)$. By the fact that $((\mathsf{vk}_{\mathsf{R}}, m_{\mathsf{R}}), \sigma_{\mathsf{R}}) \in \mathsf{R}$, we know that $\mathsf{Vrfy}(\mathsf{vk}_{\mathsf{R}}, m_{\mathsf{R}}, \sigma_{\mathsf{R}}) = 1$ and thus $\mathsf{e}(\sigma_{\mathsf{R}}, g_2) = \mathsf{e}(\mathsf{H}(m_{\mathsf{R}}), \mathsf{vk}_{\mathsf{R}})$. Therefore $\mathsf{e}(\sigma, g_2) = \mathsf{e}(\mathsf{H}(m_{\mathsf{R}}), \mathsf{vk}_{\mathsf{R}}) \cdot \mathsf{e}(\mathsf{H}(m), \mathsf{vk}) \cdot \mathsf{e}(\mathsf{H}(m_{\mathsf{R}}), \mathsf{vk}_{\mathsf{R}})^{-1}$. By the commutativity of the group operation, the first and the last term cancel out and we arrive at $\mathsf{e}(\sigma, g_2) = \mathsf{e}(\mathsf{H}(m), \mathsf{vk})$, which implies that $\mathsf{Vrfy}(\mathsf{vk}, m, \sigma := \mathsf{Adapt}(\mathsf{sk}_A, \mathsf{vk}, \hat{\sigma}, w, m)) = 1$.

**Lemma 4 (Figure 9 satisfies unlinkability).**

*Proof.* Using the fact that BLS signatures are unique, we will prove that, given the tuple of keys

$$(\mathsf{vk}, (\mathsf{sk}_A, \mathsf{vk}_A), (\mathsf{sk}_E, \mathsf{vk}_E)) \leftarrow \mathsf{BLS.KeyGen}(n)$$

the output distribution of $D_1$ and $D_2$ are exactly the same for all $(x, w) \in \mathsf{R}$, where

$$D_1 := \left\{ \Pi\mathsf{Sign}_{\langle \mathsf{sk}_A, \mathsf{sk}_E \rangle}(\mathsf{vk}, m) \right\}_m$$

$$D_2 := \{\mathsf{Adapt}(\mathsf{sk}_A, \mathsf{vk}, \mathsf{PreSign}_E(\mathsf{sk}_E, m, x, \mathsf{PreSign}_A(\mathsf{sk}_A, m, w, x)), w, m)\}_m$$

For any message $m$, the output of $\Pi\mathsf{Sign}_{\langle \mathsf{sk}_A, \mathsf{sk}_E \rangle}(\mathsf{vk}, m)$ is a signature $\sigma = \mathsf{H}(m)^{\mathsf{sk}_A} \cdot \mathsf{H}(m)^{\mathsf{sk}_E}$. On the other hand, the output of $\mathsf{PreSign}_E(\mathsf{sk}_E, m, x, \mathsf{PreSign}_A(\mathsf{sk}_A, m, w, x))$ is a presignature $\hat{\sigma} = \mathsf{H}(m)^{\mathsf{sk}_A} \cdot \mathsf{H}(m_{\mathsf{R}})^{\mathsf{sk}_{\mathsf{R}}} \cdot \mathsf{H}(m)^{\mathsf{sk}_E}$. Then, the output of $\mathsf{Adapt}(\mathsf{sk}_A, \mathsf{vk}, \hat{\sigma}, w, m)$ is a signature $\sigma' = \mathsf{H}(m)^{\mathsf{sk}_A} \cdot \mathsf{H}(m)^{\mathsf{sk}_E}$.

It is easy to see that both $\sigma'$ and $\sigma$ are identically the same object. This holds for any message $m$ and any $(x, w) \in \mathsf{R}$. Therefore both distributions are identical and therefore no adversary $\mathcal{A}$ can distinguish between them. As a consequence, $\Pr[\mathsf{Unlinkability}_{\mathcal{A},\Theta}(n, 0)] = \Pr[\mathsf{Unlinkability}_{\mathcal{A},\Theta}(n, 1)]$. Therefore $|\Pr[\mathsf{Unlinkability}_{\mathcal{A},\Theta}(n, 0)] - \Pr[\mathsf{Unlinkability}_{\mathcal{A},\Theta}(n, 1)] | \leq \mathsf{negl}(n)$.

**Lemma 5 (Figure 9 satisfies full extractability).**

*Proof.* In order to prove full extractability, we will make a succession of hybrid games, proving that each hybrid is indistinguishable from the previous one. At the final hybrid, we will make a reduction to the unforgeability of BLS signatures. Let $\mathcal{H}_i \approx \mathcal{H}_{i+1}$ denote $|\Pr[\mathcal{H}_i = 1] - \Pr[\mathcal{H}_{i+1} = 1]| \leq \mathsf{negl}(n)$.

**Hybrid $\mathcal{H}_0$** This hybrid corresponds to the original experiment $\text{2P-AISO-AS-FExt}_{\mathcal{A},\Theta}(n)$ (Figure 5).

**Hybrid $\mathcal{H}_1$** This hybrid is identical to $\mathcal{H}_0$ with the exception that we abort at the end of the game if for any of the statements $x$ in the set $\{(x, \hat{\sigma}) \in \mathcal{T}_E[m] \mid x \notin \mathcal{X}\}$ it holds that $x \notin L_{\mathsf{R}}$. In Figure 17 we present the changes in the game.

*Claim.* If the 2P-AISO-AS scheme satisfies computational preverify soundness, then $\mathcal{H}_0 \approx \mathcal{H}_1$.

*Proof.* Let *Bad* be the event that, for the adversarial forgery $(m, \sigma)$ there exists a statement $(x, _) \in \{(x, \hat{\sigma}) \in \mathcal{T}_E[m] \mid x \notin \mathcal{X}\}$ such that $x \notin L_{\mathsf{R}}$. Assume that a $\mathsf{PPT}$ adversary $\mathcal{A}$ was able to trigger *Bad* with non-negligible probability. If $\mathcal{A}$ is able to trigger *Bad* with non-negligible probability, then we can create a $\mathsf{PPT}$ adversary $\mathcal{B}$ that breaks the computational preverify soundness of the 2P-AISO-AS scheme. The adversary $\mathcal{B}$ receives the (honestly generated) key tuple $(\mathsf{vk}, (\mathsf{vk}_A, \mathsf{sk}_A), (\mathsf{vk}_E, \mathsf{sk}_E))$ from the preverify soundness challenger. Next, $\mathcal{B}$ forwards $\mathsf{vk}, \mathsf{vk}_E, \mathsf{vk}_A, \mathsf{sk}_A$ to $\mathcal{A}$ and $\mathcal{B}$ acts as the challenger of $\mathcal{H}_1$. At the end of the game, $\mathcal{A}$ causes event *Bad* to trigger. This implies that, $\mathcal{A}$ returned a forgery $(m, \sigma)$ and in the table entry $\mathcal{T}[m]$ there is a tuple $(x, \hat{\sigma})$ where $x \notin L_{\mathsf{R}}$. Then $\mathcal{B}$ simply outputs $(\hat{\sigma}, m)$.

Notice that for the event to trigger, it is necessary that the set $\{(x, \hat{\sigma}) \in \mathcal{T}_E[m] \mid x \notin \mathcal{X}\}$ is non-empty. This implies that $\mathcal{A}$ had to query oracle $\mathsf{OPreSign}_E$ at least once. Let input $(x, m, pm)$ be the query input

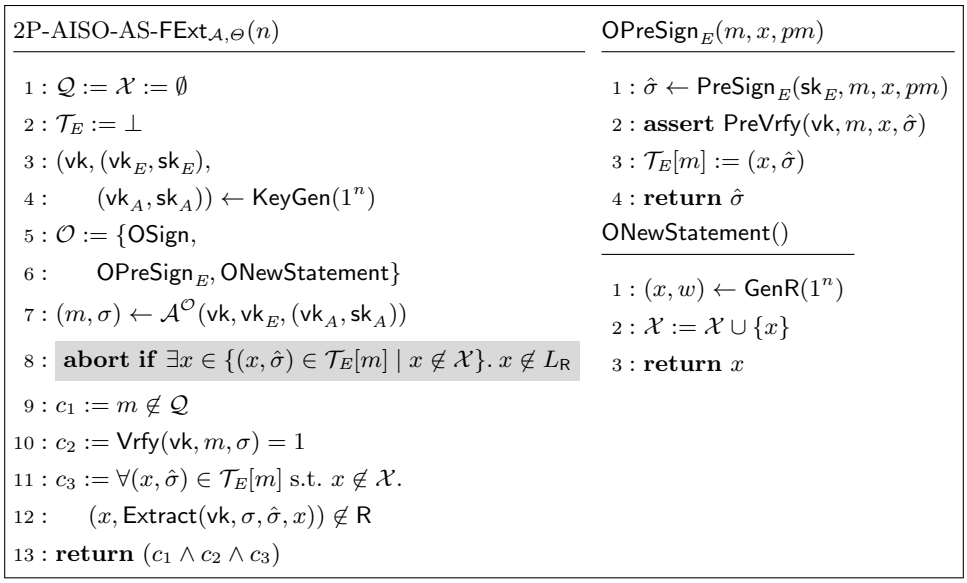



Fig. 17: Hybrid one of the proof for Figure 9 satisfying full extractability. Changes are highlighted in grey.

to $\mathsf{OPreSign}_E$ where $m$ is the message of the final forgery. Crucially, the assertion of line 2 of this oracle passed. This implies that the oracle ran $\hat{\sigma} \leftarrow \mathsf{PreSign}_E(\mathsf{sk}_E, x, m, pm)$ and this presignature satisfied $\mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma}) = 1$. Therefore the output $(\hat{\sigma}, m)$ of $\mathcal{B}$ satisfies $\mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma}) = 1$ for a statement $x \notin L_{\mathsf{R}}$. Thus, $\mathcal{B}$ breaks computational preverify soundness with the same probability that $\mathcal{A}$ causes *Bad* to trigger.

**Hybrid** $\mathcal{H}_2$ This hybrid is identical to $\mathcal{H}_1$ with the exception that we abort at the end of the game if the set $\{(x, \hat{\sigma}) \in \mathcal{T}_E[m] \mid x \notin \mathcal{X}\}$ is non-empty. In Figure 18 we present the changes in the game.

*Claim.* If the 2P-AISO-AS scheme satisfies adaptability, if the BLS signature is unique and if the hard relation $\mathsf{R}$ is injective, then $\mathcal{H}_1 \approx \mathcal{H}_2$.

*Proof.* Let $Bad_2$ be the event that, for the adversarial forgery $(m, \sigma)$ there exists a statement the set $\{(x, \hat{\sigma}) \in \mathcal{T}_E[m] \mid x \notin \mathcal{X}\}$ is non-empty. From $\mathcal{H}_1$ we know that for all pairs $(x, \hat{\sigma}) \in \{(x, \hat{\sigma}) \in \mathcal{T}_E[m] \mid x \notin \mathcal{X}\}$ it holds that $x \in L_{\mathsf{R}}$. Therefore, for each statement $x$ of these pairs, there must exist a corresponding witness $w$ such that $(x, w) \in \mathsf{R}$. By the assertion on line 2 of oracle $\mathsf{OPreSign}_E$ we also know that $\mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma}) = 1$. Let $\sigma' := \mathsf{Adapt}(\mathsf{sk}_A, \mathsf{vk}, \hat{\sigma}, w, m)$. By adaptability of the 2P-AISO-AS scheme, it must hold that $\mathsf{Vrfy}(\mathsf{vk}, m, \sigma') = 1$. Since BLS signatures are unique for a given message, it must be the case that $\sigma' = \sigma$. Let $w' := \mathsf{Extract}(\mathsf{Adapt}(\mathsf{sk}_A, \mathsf{vk}, \hat{\sigma}, w, m), \hat{\sigma}, x) = \mathsf{Extract}(\sigma, \hat{\sigma}, x)$. Since the hard relation is injective, $w' = w$. Therefore $(x, w) \in \mathsf{R}$ implies that $(x, w') \in \mathsf{R}$. Thus, if there exists an adversary that wins $\mathcal{H}_2$, then it does not trigger event $Bad_2$.

**Hybrid** $\mathcal{H}_3$ This hybrid is identical to $\mathcal{H}_2$ with the exception that we abort if $\mathcal{T}_E[m] \neq \emptyset$, where $m$ is the forgery message. In Figure 19 we include the changes in the game.

*Claim.* If $\mathsf{R}$ is $q$-hard and the number of queries to $\mathsf{ONewStatement}$ is bounded by $q$, then $\mathcal{H}_2 \approx \mathcal{H}_3$.

*Proof.* Let $Bad_3$ be the event that $\mathcal{T}_E[m] \neq \emptyset$, where $m$ is the forgery message. If there exists a PPT adversary $\mathcal{A}$ that triggers $Bad_3$, then we can create a PPT adversary $\mathcal{B}$ that breaks the $q$-hardness of $\mathsf{R}$, where $q$ denotes the number of $\mathsf{ONewStatement}$ queries that $\mathcal{A}$ makes.

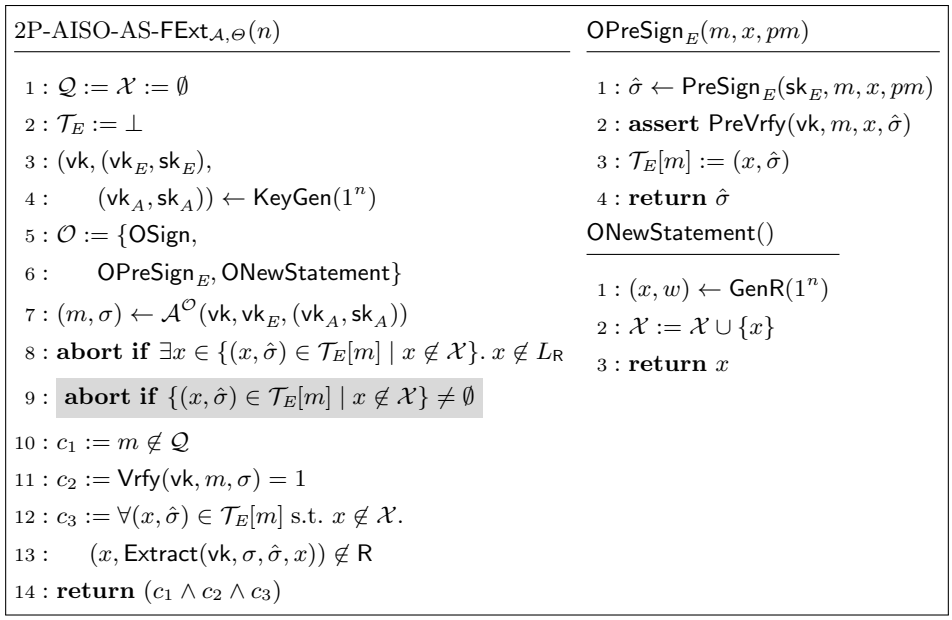



Fig. 18: Hybrid two of the proof for Figure 9 satisfying full extractability. Changes are highlighted in grey.

The challenger of $q$-hardness first samples $q$ instances $(x_i, w_i)$, for $i \in [q]$, of $\mathsf{R}$ and sends all the statements $\{x_i\}_{i\in[q]}$ to $\mathcal{B}$. The adversary $\mathcal{B}$ plays $\mathcal{H}_3$ with $\mathcal{A}$ honestly. Whenever $\mathcal{A}$ makes the $i$-th query to oracle $\mathsf{ONewStatement}$, $\mathcal{B}$ stores $x_i$ in $\mathcal{X}$ and returns $x_i$. Finally, $\mathcal{A}$ returns a forgery $(m, \sigma)$. If the event $Bad_3$ triggers, then $\mathcal{T}_E[m] \neq \emptyset$. However, by $\mathcal{H}_2$ we know that $\{(x, \hat{\sigma}) \in \mathcal{T}_E[m] \mid x \notin \mathcal{X}\} = \emptyset$. Therefore, it must be the case that for all $(x, \hat{\sigma}) \in \mathcal{T}_E[m]$ it holds that $x \in \mathcal{X}$. Thus, $\mathcal{B}$ can compute $w := \mathsf{Extract}(\mathsf{vk}, \sigma, \hat{\sigma}, x)$ and returns $w$ to the challenger of the $q$-hardness of $\mathsf{R}$.

We know that for each statement $x_i$ there exists a corresponding witness $w_i$ such that $(x_i, w_i) \in \mathsf{R}$. Thus, for $(x, \hat{\sigma}) \in \mathcal{T}_E[m]$ there exists a witness $w$ such that $(w, w) \in \mathsf{R}$. The assertion of line 2 of $\mathsf{OPreSign}_E$ ensures that $\mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma}) = 1$. Let $\sigma' := \mathsf{Adapt}(\mathsf{sk}_A, \mathsf{vk}, \hat{\sigma}, w, m)$. By adaptability of the 2P-AISO-AS scheme, it must hold that $\mathsf{Vrfy}(\mathsf{vk}, m, \sigma') = 1$. Since BLS signatures are unique for a given message, it must be the case that $\sigma' = \sigma$. Let $w' := \mathsf{Extract}(\mathsf{Adapt}(\mathsf{sk}_A, \mathsf{vk}, \hat{\sigma}, w, m), \hat{\sigma}, x) = \mathsf{Extract}(\sigma, \hat{\sigma}, x)$. Since the hard relation is injective, $w' = w$. Therefore $w'$ is the witness corresponding to one of the $x_i$ statements.

**Reduction to EUF-CMA of BLS signatures.** Finally, we reduce the security to the EUF-CMA security of the BLS signature scheme.

*Claim.* If BLS signature scheme satisfies EUF-CMA security, then for all $\mathsf{PPT}$ adversaries $\mathcal{A}$ it holds that $\Pr[\mathsf{FExt}\mathcal{A}, \Theta(n) = 1] \leq \mathsf{negl}(n)$.

*Proof.* Assume that there exists a $\mathsf{PPT}$ adversary $\mathcal{A}$ that wins $\mathcal{H}_3$ with non-negligible probability. Then we can create a $\mathsf{PPT}$ adversary that wins EUF-CMA against the BLS signature scheme.

First, the challenger of the EUF-CMA computes keys $(\mathsf{vk}, (\mathsf{vk}_A, \mathsf{sk}_A), (\mathsf{vk}_E, \mathsf{sk}_E)) \leftarrow \mathsf{KeyGen}(n)$ and sends $(\mathsf{vk}, \mathsf{vk}_E, \mathsf{vk}_A, \mathsf{sk}_A)$ to $\mathcal{B}$. The adversary $\mathcal{B}$ forwards $(\mathsf{vk}, \mathsf{vk}_E, \mathsf{vk}_A, \mathsf{sk}_A)$ to $\mathcal{A}$. Whenever $\mathcal{A}$ makes a query to the signing oracles, $\mathcal{B}$ simply forwards this query to the challenger of the EUF-CMA. Whenever $\mathcal{A}$ makes a query to the oracle $\mathsf{ONewStatement}$, $\mathcal{B}$ computes this oracle honestly. Whenever $\mathcal{A}$ makes a query to the oracle $\mathsf{OPreSign}_E$, $\mathcal{B}$ queries the signing oracles of the EUF-CMA challenger to obtain $\sigma$. Then, $\mathcal{B}$ computes $\sigma_A := \mathsf{Sign}(\mathsf{sk}_A, m)$ on their own and thus obtains $\sigma_E := \mathrm{Agg}(\sigma, \sigma_A)$. Finally, $\mathcal{B}$ outputs $\hat{\sigma} := \mathrm{Agg}(pm, \sigma_E)$. Finally, when $\mathcal{A}$ produces the forgery $(m, \sigma)$, $\mathcal{B}$ outputs this as their own forgery.

2P-AISO-AS-FExt$_{\mathcal{A},\Theta}(n)$

$\mathcal{Q} := \mathcal{X} := \emptyset$
$\mathcal{T}_E := \bot$
$(\mathsf{vk}, (\mathsf{vk}_E, \mathsf{sk}_E),$
$\quad (\mathsf{vk}_A, \mathsf{sk}_A)) \leftarrow \mathsf{KeyGen}(1^n)$
$\mathcal{O} := \{\mathsf{OSign},$
$\quad \mathsf{OPreSign}_E, \mathsf{ONewStatement}\}$
$(m, \sigma) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{vk}, \mathsf{vk}_E, (\mathsf{vk}_A, \mathsf{sk}_A))$
**abort if** $\exists x \in \{(x, \hat{\sigma}) \in \mathcal{T}_E[m] \mid x \notin \mathcal{X}\}.\ x \notin L_{\mathsf{R}}$
**abort if** $\{(x, \hat{\sigma}) \in \mathcal{T}_E[m] \mid x \notin \mathcal{X}\} \neq \emptyset$
**abort if** $\mathcal{T}_E[m] \neq \emptyset$
$c_1 := m \notin \mathcal{Q}$
$c_2 := \mathsf{Vrfy}(\mathsf{vk}, m, \sigma) = 1$
$c_3 := \forall (x, \hat{\sigma}) \in \mathcal{T}_E[m]$ s.t. $x \notin \mathcal{X}.$
$\quad (x, \mathsf{Extract}(\mathsf{vk}, \sigma, \hat{\sigma}, x)) \notin \mathsf{R}$
**return** $(c_1 \wedge c_2 \wedge c_3)$

$\mathsf{OPreSign}_E(m, x, pm)$

$\hat{\sigma} \leftarrow \mathsf{PreSign}_E(\mathsf{sk}_E, m, x, pm)$
**assert** $\mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma})$
$\mathcal{T}_E[m] := (x, \hat{\sigma})$
**return** $\hat{\sigma}$

ONewStatement()

$(x, w) \leftarrow \mathsf{GenR}(1^n)$
$\mathcal{X} := \mathcal{X} \cup \{x\}$
**return** $x$

Fig. 19: Hybrid three of the proof for Figure 9 satisfying full extractability. Changes are highlighted in grey.

By the assumption that $\mathcal{A}$ wins $\mathcal{H}_3$ with non-negligible probability, it holds that (i) $m \notin \mathcal{Q}$, (ii) $\mathsf{Vrfy}(\mathsf{vk}, m, \sigma) = 1$ and (iii) $\forall (x, \hat{\sigma}) \in \mathcal{T}_E[m]$ s.t. $x \notin \mathcal{X}.(x, \mathsf{Extract}(\mathsf{vk}, \sigma, \hat{\sigma}, x)) \notin \mathsf{R}$.

Observe that $\mathcal{H}_3$ aborts if the $\mathcal{T}_E[m]$ is non-empty. Therefore $\mathcal{T}_E[m]$ must be empty and thus (iii) is trivially satisfied. This set being empty in particular implies that $m$ was never queried to $\mathsf{OPreSign}_E$. This implies that $\mathsf{OPreSign}_E$ did not trigger a query on $m$ to the signing oracles of the EUF-CMA challenger. Moreover, (i) ensures that $m$ was not queried to the signing oracles of $\mathcal{B}$ and thus, these signing oracles did not trigger a query on $m$ to the signing oracles of the EUF-CMA either. Therefore it holds that $m \notin \mathcal{Q}'$ where $\mathcal{Q}'$ is the set of messages queried to the signing oracles of the EUF-CMA challenger. Finally, (ii) ensures that the signature verifies. Therefore, $\mathcal{B}$ would have returned a valid forgery that breaks the EUF-CMA of BLS signatures with the same probability that $\mathcal{A}$ would win $\mathcal{H}_3$.

**Lemma 6 (Figure 9 satisfies weak unforgeability).**

*Proof.* In order to prove weak unforgeability, we will make a succession of hybrid games, proving that each hybrid is indistinguishable from the previous one. At the final hybrid, we will make a reduction to the unforgeability of BLS signatures. Let $\mathcal{H}_i \approx \mathcal{H}_{i+1}$ denote $|\Pr[\mathcal{H}_i = 1] - \Pr[\mathcal{H}_{i+1} = 1]| \leq \mathsf{negl}(n)$.

**Hybrid $\mathcal{H}_0$** This hybrid corresponds to the original experiment 2P-AISO-AS-WUnf$_{\mathcal{A},\Theta}(n)$ (Figure 6).

**Hybrid $\mathcal{H}_1$** This hybrid is identical to $\mathcal{H}_0$ with the exception that we change oracle $\mathsf{OAdapt}$. In $\mathsf{OAdapt}$, we abort if $x \notin \mathcal{T}_A[m]$ where $x$ and $m$ are part of the query. Intuitively, this hybrid implies that the adversarial *Extractor* could not create a valid presignature on their own (i.e., without invoking $\mathsf{OPreSign}_A$). Therefore, all queries to $\mathsf{OAdapt}$ must have a corresponding and preceding query to $\mathsf{OPreSign}_A$.

*Claim.* If $\mathsf{R}$ is a hard relation, then $\mathcal{H}_0 \approx \mathcal{H}_1$.

*Proof.* Let *Bad* be the event that during a query to $\mathsf{OAdapt}$ (on input $x$, $m$, and any presignature $\hat{\sigma}$) it holds that $x \notin \mathcal{T}_A[m]$. Intuitively, if the event *Bad* triggers, then it means that the presignature $\hat{\sigma}$ queried

to $\mathsf{OAdapt}$ is such that (i) preverifies and (ii) was not computed from the output $pm$ of a query to oracle $\mathsf{OPreSign}_A$. Assume that there exists an adversary $\mathcal{A}$ that triggers this event with non-negligible probability. If this is the case, then we can create an adversary $\mathcal{B}$ that using $\mathcal{A}$ breaks the hardness of $\mathsf{R}$.

The challenger of the hard relation samples an instance $(x^*, w^*)$ and sends $x^*$ to $\mathcal{B}$. Then, $\mathcal{B}$ plays $\mathcal{H}_1$ with $\mathcal{A}$ normally, with the exception than $\mathcal{B}$ makes a guess: out of the $q$ queries to $\mathsf{ONewStatement}$, $\mathcal{B}$ picks one at random. At that query, $\mathcal{B}$ will not run the code for $\mathsf{ONewStatement}$ from $\mathcal{H}_0$ and instead will respond with $x^*$. If $\mathcal{A}$ triggers the event *Bad* with non-negligible probability, then there exists an honestly generated statement $x$ that is not used on queries to $\mathsf{OPreSign}_A$. The guessing aims to set that statement in particular to be $x^*$. On a query to $\mathsf{OPreSign}_A$ with input $x$, if $x = x^*$, then $\mathcal{B}$ aborts. On a query $(\hat{\sigma}, x, m)$ to $\mathsf{OAdapt}$, $\mathcal{B}$ checks whether the event *Bad* triggers. If it does, then $\mathcal{B}$ simply computes the signature $\sigma := \mathit{\Pi}\mathsf{Sign}_{\langle \mathsf{sk}_A, \mathsf{sk}_E \rangle}(\mathsf{vk}, m)$. Then, $\mathcal{B}$ extracts the witness $w := \mathsf{Extract}(\mathsf{vk}, \hat{\sigma}, \sigma)$ and returns $w$ to the adversary of the hard relation.

Since BLS signatures are unique, the signature $\sigma$ is the same signature as $\sigma' := \mathsf{Adapt}(\mathsf{sk}_A, \mathsf{vk}, \hat{\sigma}, w, m)$. We know that $\sigma := \mathsf{H}(m)^{\mathsf{sk}_A} \cdot \mathsf{H}(m)^{\mathsf{sk}_E}$. We know that $\mathsf{PreVrfy}(\mathsf{vk}, m, \hat{\sigma}, x) = 1$ and therefore $\mathsf{e}(\hat{\sigma}, g) = \mathsf{e}(\mathsf{H}(m), \mathsf{vk}) \cdot \mathsf{e}(\mathsf{H}(m_\mathsf{R}), \mathsf{vk}_\mathsf{R}) = \mathsf{e}(\mathsf{H}(m), \mathsf{vk}_A) \cdot \mathsf{e}(\mathsf{H}(m), \mathsf{vk}_E) \cdot \mathsf{e}(\mathsf{H}(m_\mathsf{R}), \mathsf{vk}_\mathsf{R})$. We also know that $\mathsf{Extract}(\hat{\sigma}, \sigma)$ performs the operation $w := \hat{\sigma} \cdot \sigma^{-1}$ and returns $w$ if $(x, w) \in \mathsf{R}$. The result $w$ of computing $\hat{\sigma} \cdot \sigma^{-1}$ thus satisfies

$$\begin{gathered}\mathsf{e}(\hat{\sigma} \cdot \sigma^{-1}, g) = \\ \mathsf{e}(\hat{\sigma}, g) \cdot \mathsf{e}(\sigma, g)^{-1} = \\ \mathsf{e}(\mathsf{H}(m), \mathsf{vk}_A) \cdot \mathsf{e}(\mathsf{H}(m), \mathsf{vk}_E) \cdot \mathsf{e}(\mathsf{H}(m_\mathsf{R}), \mathsf{vk}_\mathsf{R}) \cdot (\mathsf{e}(\mathsf{H}(m), \mathsf{vk}_A) \cdot \mathsf{e}(\mathsf{H}(m), \mathsf{vk}_E))^{-1} = \\ \mathsf{e}(\mathsf{H}(m_\mathsf{R}), \mathsf{vk}_\mathsf{R})\end{gathered}$$

Thus $\mathsf{Extract}(\hat{\sigma}, \sigma)$ yields a witness that is a valid signature for $(\mathsf{vk}_\mathsf{R}, m_\mathsf{R}) = x$. The guessing argument incurs only a polynomial security loss, since $\mathcal{A}$ is $\mathsf{PPT}$, and hence, the number of queries it makes to $\mathsf{ONewStatement}$ is $q = \mathsf{poly}(n)$. Thus, if $\mathcal{A}$ triggers event *Bad* with non-negligible probability, then $\mathcal{B}$ breaks the hardness of $\mathsf{R}$ with non-negligible probability.

**Hybrid $\mathcal{H}_2$** This hybrid is identical to $\mathcal{H}_1$ with the exception that we change oracle $\mathsf{ONewStatement}$. In Figure 20 we include the code of the new oracle. We modify the sampling of statements in $\mathsf{ONewStatement}$. We first sample $r \leftarrow_\$ \mathbb{Z}_q$, and then set $\mathsf{vk}_\mathsf{R} := \mathsf{vk}_A \cdot g^r$. We still sample $m_\mathsf{R}$ uniformly at random as before. Thus, the witness is computed as the signature on $m_\mathsf{R}$ under the key $\mathsf{sk}_\mathsf{R} := \mathsf{sk}_A + r$. We store $r$ inside of $\mathcal{R}[x]$ so that we can recover $r$ for a given statement $x$ later in other oracles.

*Claim.* $\mathcal{H}_1 \approx \mathcal{H}_2$

*Proof.* We observe that the only difference between $\mathcal{H}_1$ and $\mathcal{H}_2$ is the statements generated in $\mathsf{ONewStatement}$. We begin by proving that the distributions $\{x \leftarrow \mathsf{ONewStatement}()\}$ and $\{x \mid (x, _) \leftarrow \mathsf{GenR}(n)\}$ are indistinguishable.

In $\mathsf{GenR}$ as described in Figure 9, we sample $\mathsf{sk}_\mathsf{R}$ at random and then set the verification key $\mathsf{vk}_\mathsf{R}$ as $g^{\mathsf{sk}_\mathsf{R}}$. Thus, $\mathsf{vk}_\mathsf{R}$ follows the uniform distribution. The message $m_\mathsf{R}$ is uniformly sampled. Therefore, the statement $x := (\mathsf{vk}_\mathsf{R}, m_\mathsf{R})$ follows a distribution $\mathcal{U} \times \mathcal{U}$ where $\mathcal{U}$ is the uniform distribution.

On the other hand, in $\mathsf{ONewStatement}$ as defined in Figure 20 the verification key $\mathsf{vk}_\mathsf{R}$ is defined as $\mathrm{Agg}(\mathsf{vk}_A, g^r)$ where $r$ was sampled uniformly at random and thus both $\mathsf{vk}_A$ and $g^r$ follow the uniform distribution. Since $\mathsf{vk}_A$ and $g^r$ are elements of a cyclic group, then $\mathsf{vk}_\mathsf{R}$ is uniformly distributed. The message $m_\mathsf{R}$ is uniformly sampled. Thus, the statement $x := (\mathsf{vk}'_\mathsf{R}, m_\mathsf{R})$ also follows a distribution $\mathcal{U} \times \mathcal{U}$.

Thus, both $\{x \leftarrow \mathsf{ONewStatement}()\}$ and $\{x \mid (x, _) \leftarrow \mathsf{GenR}(n)\}$ follow the same distribution and thus are indistinguishable. The rest of $\mathcal{H}_2$ is a faithful computation of $\mathcal{H}_1$, therefore $\mathcal{H}_1 \approx \mathcal{H}_2$.

**Hybrid $\mathcal{H}_3$** This hybrid is identical to $\mathcal{H}_2$ with the exception that we change oracle $\mathsf{OPreSign}_A$. In $\mathsf{OPreSign}_A$ we will sample a new message $m'$ uniformly at random. Then, we program the random oracle at $m'$ to be $\mathsf{H}[m'] := \mathsf{H}(m) \cdot \mathsf{H}(m_\mathsf{R})$. We store $m'$ into a set $\mathcal{M}'$. We also store the output $pm$ into a table indexed by the pair of message and statement $\mathcal{PM}[(m, x)] := pm$ to be used in future hybrids. We abort if the adversary queries the random oracle for a message $m \in \mathcal{M}'$. In Figure 21 we include the new $\mathsf{OPreSign}_A$ oracle.

*Claim.* If the size of the message space $\mathcal{M}$ is exponential in the security parameter, then $\mathcal{H}_2 \approx \mathcal{H}_3$.

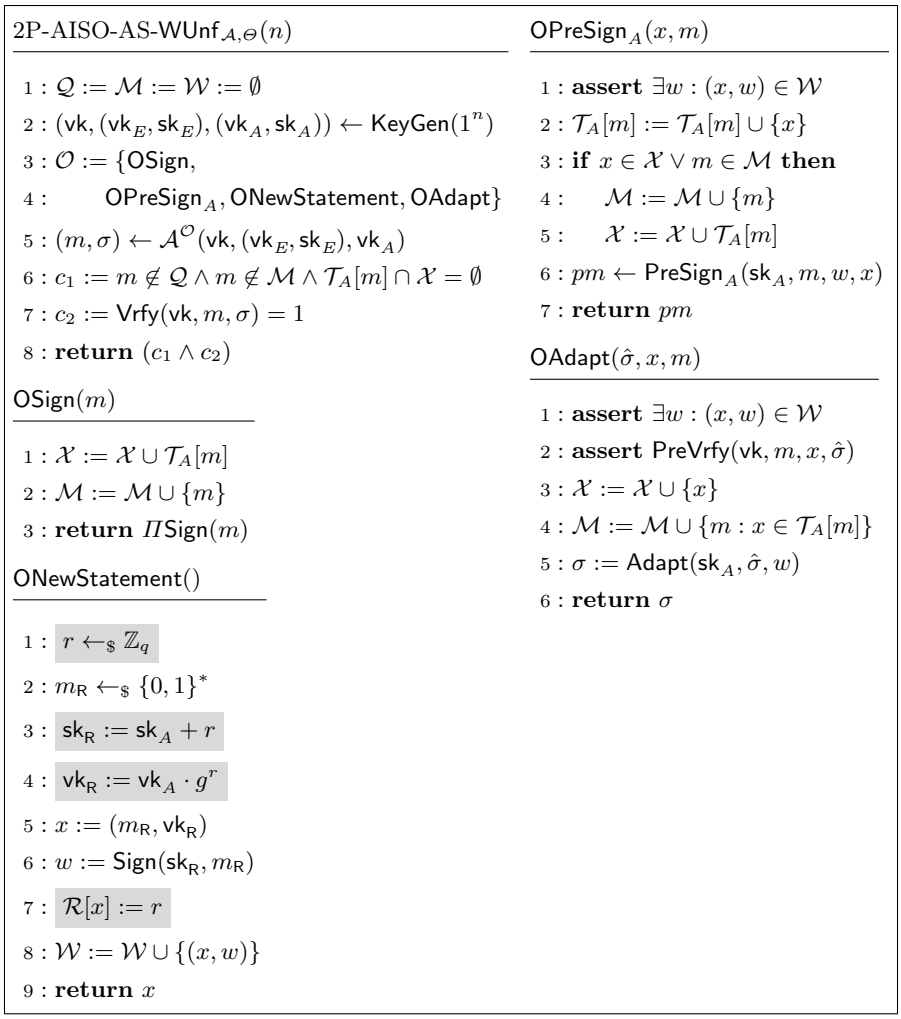



Fig. 20: Definition for $\mathcal{H}_2$ used in the proof of WUnf of Figure 9. Changes w.r.t. the previous hybrid are highlighted in gray.

*Proof.* The only difference with $\mathcal{H}_3$ is that in $\mathcal{H}_2$ we are programming the random oracle on queries to $\mathsf{OPreSign}_A$. Therefore, the only possibility to distinguish between $\mathcal{H}_3$ and $\mathcal{H}_2$ is for $\mathcal{A}$ find out the programming of the random oracle. In order to find out, $\mathcal{A}$ would need to query $\mathsf{H}$ on one of the messages $m'$ randomly sampled in each query to $\mathsf{PreSign}_A$. Let $q_H$ and $q_P$ denote the number queries that $\mathcal{A}$ makes to $\mathsf{H}$ and $\mathsf{OPreSign}_A$ respectively. The distinguishing probability for a single $\mathsf{OPreSign}_A$ query would be $\frac{q_H}{2^\lambda}$. After $\mathcal{A}$ makes $q_P$ queries to $\mathsf{OPreSign}_A$, the distinguishing probability becomes $\frac{q_H \cdot q_P}{2^\lambda}$. This probability is still negligible, thus $\mathcal{H}_2 \approx \mathcal{H}_3$.

**Hybrid $\mathcal{H}_4$** This hybrid is identical to $\mathcal{H}_3$ with the exception that we change oracle $\mathsf{OPreSign}_A$. In $\mathsf{OPreSign}_A$, after programming the random oracle at $\mathsf{H}[m'] := \mathsf{H}(m) \cdot \mathsf{H}(m_\mathsf{R})$, we generate a signature on $m'$ using $\mathsf{sk}_A$ as signing key: $\sigma' := \mathsf{H}(m')^{\mathsf{sk}_A}$. Then we generate another signature on $m_\mathsf{R}$ using $r$ as signing key: $\sigma_r := \mathsf{H}(m_\mathsf{R})^r$. Finally, the output of the oracle is set to $pm := \sigma' \cdot \sigma_r$. The description of the oracle is in Figure 22.

*Claim.* $\mathcal{H}_3 \approx \mathcal{H}_4$.

2P-AISO-AS-WUnf$_{\mathcal{A},\Theta}(n)$

$\mathcal{Q} := \mathcal{M} := \mathcal{W} := \emptyset$
$(\mathsf{vk}, (\mathsf{vk}_E, \mathsf{sk}_E), (\mathsf{vk}_A, \mathsf{sk}_A)) \leftarrow \mathsf{KeyGen}(1^n)$
$\mathcal{O} := \{\mathsf{OSign},$
$\quad \mathsf{OPreSign}_A, \mathsf{ONewStatement}, \mathsf{OAdapt}\}$
$(m, \sigma) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{vk}, (\mathsf{vk}_E, \mathsf{sk}_E), \mathsf{vk}_A)$
$c_1 := m \notin \mathcal{Q} \wedge m \notin \mathcal{M} \wedge \mathcal{T}_A[m] \cap \mathcal{X} = \emptyset$
$c_2 := \mathsf{Vrfy}(\mathsf{vk}, m, \sigma) = 1$
**return** $(c_1 \wedge c_2)$

$\mathsf{OSign}(m)$

$\mathcal{X} := \mathcal{X} \cup \mathcal{T}_A[m]$
$\mathcal{M} := \mathcal{M} \cup \{m\}$
**return** $\Pi\mathsf{Sign}(m)$

$\mathsf{ONewStatement}()$

$r \leftarrow_\$ \mathbb{Z}_q$
$m_\mathsf{R} \leftarrow_\$ \{0,1\}^*$
$\mathsf{sk}_\mathsf{R} := \mathsf{sk}_A + r$
$\mathsf{vk}_\mathsf{R} := \mathsf{vk}_A \cdot g^r$
$x := (m_\mathsf{R}, \mathsf{vk}_\mathsf{R})$
$w := \mathsf{Sign}(\mathsf{sk}_\mathsf{R}, m_\mathsf{R})$
$\mathcal{R}[x] := r$
$\mathcal{W} := \mathcal{W} \cup \{(x, w)\}$
**return** $x$

$\mathsf{OPreSign}_A(m, x)$

**assert** $\exists w \ : \ (x, w) \in \mathcal{W}$
$\mathcal{T}_A[m] := \mathcal{T}_A[m] \cup \{x\}$
**if** $x \in \mathcal{X} \vee m \in \mathcal{M}$ **then**
$\quad \mathcal{M} := \mathcal{M} \cup \{m\}$
$\quad \mathcal{X} := \mathcal{X} \cup \mathcal{T}_A[m]$
$(m_\mathsf{R}, \mathsf{vk}_\mathsf{R}) := x$
$m' \leftarrow_\$ \{0,1\}^*$
$\mathsf{H}[m'] := \mathsf{H}(m) \cdot \mathsf{H}(m_\mathsf{R})$
$\mathcal{M}' := \mathcal{M}' \cup m'$
$pm := \mathsf{PreSign}_A(\mathsf{sk}_A, w, m, x)$
**return** $pm$

$\mathsf{OAdapt}(\hat{\sigma}, x, m)$

**assert** $\exists w : (x, w) \in \mathcal{W}$
**assert** $\mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma})$
$\mathcal{X} := \mathcal{X} \cup \{x\}$
$\mathcal{M} := \mathcal{M} \cup \{m : x \in \mathcal{T}_A[m]\}$
$\sigma := \mathsf{Adapt}(\mathsf{sk}_A, \hat{\sigma}, w)$
**return** $\sigma$

Fig. 21: Definition for $\mathcal{H}_3$ used in the proof of WUnf of Figure 9. Changes w.r.t. the previous hybrid are highlighted in gray.

*Proof.* In $\mathcal{H}_3$, the output of $\mathsf{OPreSign}_A$ was $pm' = \mathsf{H}(m)^{\mathsf{sk}_A} \cdot w = \mathsf{H}(m)^{\mathsf{sk}_A} \cdot \mathsf{H}(m_\mathsf{R})^{\mathsf{sk}_\mathsf{R}}$. Due to $\mathsf{ONewStatement}$ as defined in Figure 20, it holds that $\mathsf{sk}_\mathsf{R} = \mathsf{sk}_A + r$, therefore $pm' = \mathsf{H}(m)^{\mathsf{sk}_A} \cdot \mathsf{H}(m_\mathsf{R})^{\mathsf{sk}_A + r}$ In $\mathcal{H}_4$ notice that the output of $\mathsf{OPreSign}_A$ is such that $pm = \mathsf{H}(m')^{\mathsf{sk}_A} \cdot \mathsf{H}(m_\mathsf{R})^r$. Due to the programming of the random oracle, this output is $pm = (\mathsf{H}(m) \cdot \mathsf{H}(m_\mathsf{R}))^{\mathsf{sk}_A} \cdot \mathsf{H}(m_\mathsf{R})^r = \mathsf{H}(m)^{\mathsf{sk}_A} \cdot \mathsf{H}(m_\mathsf{R})^{\mathsf{sk}_A + r} = pm'$. Thus, the output of both oracles in $\mathcal{H}_3$ and $\mathcal{H}_4$ is identical.

**Hybrid** $\mathcal{H}_5$ This hybrid is identical to $\mathcal{H}_4$ with the exception that we change oracle $\mathsf{OAdapt}$. In $\mathsf{OAdapt}$ we recompute the protocol message $pm$ by running $\mathsf{PreSign}_A$. Then, we compute the signature of the *Adaptor* $\sigma_A$ using $\mathsf{sk}_A$, and then we compute the signature of *Extractor* as $\sigma_E := \hat{\sigma} \cdot pm^{-1}$. The output of the oracle is the signature $\sigma := \sigma_E \cdot \sigma_A$. The description of the oracle is in Figure 23

*Claim.* If 2P-AISO-AS satisfies adaptability and if BLS signatures are unique, then $\mathcal{H}_4 \approx \mathcal{H}_5$.

*Proof.* Let $\sigma$ be the signature computed in oracle $\mathsf{OAdapt}$ of $\mathcal{H}_4$ and let $\sigma$ be the signature computed in oracle $\mathsf{OAdapt}$ of $\mathcal{H}_5$. By the uniqueness of BLS signatures we know that if $\mathsf{Vrfy}(\mathsf{vk}, m, \sigma) = 1$ and $\mathsf{Vrfy}(\mathsf{vk}, m, \sigma') = 1$, then $\sigma = \sigma'$. By adaptability, we know that if $\mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma}) = 1$, then if $(x, w) \in \mathsf{R}$ it holds that $\mathsf{Vrfy}(\mathsf{vk}, m, \mathsf{Adapt}(\hat{\sigma}, w)) = 1$. Since in $\mathcal{H}_4$ the signature in oracle $\mathsf{OAdapt}$ is computed as $\mathsf{Adapt}(\hat{\sigma}, w)$, we know

2P-AISO-AS-WUnf$_{\mathcal{A},\Theta}(n)$

$\mathcal{Q} := \mathcal{M} := \mathcal{W} := \emptyset$
$(\mathsf{vk}, (\mathsf{vk}_E, \mathsf{sk}_E), (\mathsf{vk}_A, \mathsf{sk}_A)) \leftarrow \mathsf{KeyGen}(1^n)$
$\mathcal{O} := \{\mathsf{OSign},$
$\quad \mathsf{OPreSign}_A, \mathsf{ONewStatement}, \mathsf{OAdapt}\}$
$(m, \sigma) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{vk}, (\mathsf{vk}_E, \mathsf{sk}_E), \mathsf{vk}_A)$
$c_1 := m \notin \mathcal{Q} \wedge m \notin \mathcal{M} \wedge \mathcal{T}_A[m] \cap \mathcal{X} = \emptyset$
$c_2 := \mathsf{Vrfy}(\mathsf{vk}, m, \sigma) = 1$
**return** $(c_1 \wedge c_2)$

$\mathsf{OSign}(m)$

$\mathcal{X} := \mathcal{X} \cup \mathcal{T}_A[m]$
$\mathcal{M} := \mathcal{M} \cup \{m\}$
**return** $\Pi\mathsf{Sign}(m)$

$\mathsf{ONewStatement}()$

$r \leftarrow_{\$} \mathbb{Z}_q$
$m_{\mathsf{R}} \leftarrow_{\$} \{0,1\}^*$
$\mathsf{sk}_{\mathsf{R}} := \mathsf{sk}_A + r$
$\mathsf{vk}_{\mathsf{R}} := \mathsf{vk}_A \cdot g^r$
$x := (m_{\mathsf{R}}, \mathsf{vk}_{\mathsf{R}})$
$w := \mathsf{Sign}(\mathsf{sk}_{\mathsf{R}}, m_{\mathsf{R}})$
$\mathcal{R}[x] := r$
$\mathcal{W} := \mathcal{W} \cup \{(x, w)\}$
**return** $x$

$\mathsf{OPreSign}_A(m, x)$

**assert** $\exists w \ : \ (x, w) \in \mathcal{W}$
$\mathcal{T}_A[m] := \mathcal{T}_A[m] \cup \{x\}$
**if** $x \in \mathcal{X} \vee m \in \mathcal{M}$ **then**
$\quad \mathcal{M} := \mathcal{M} \cup \{m\}$
$\quad \mathcal{X} := \mathcal{X} \cup \mathcal{T}_A[m]$
$(m_{\mathsf{R}}, \mathsf{vk}_{\mathsf{R}}) := x$
$m' \leftarrow_{\$} \{0,1\}^*$
$\mathsf{H}[m'] := \mathsf{H}(m) \cdot \mathsf{H}(m_{\mathsf{R}})$
$\mathcal{M}' := \mathcal{M}' \cup m'$
$\sigma' := \mathsf{Sign}(\mathsf{sk}_A + \mathsf{sk}_E, m')$
$r := \mathcal{R}[x]$
$\sigma_r := \mathsf{Sign}(r, m_{\mathsf{R}})$
$pm := \sigma' \cdot \sigma_r$
**return** $pm$

$\mathsf{OAdapt}(\hat{\sigma}, x, m)$

**assert** $\exists w : (x, w) \in \mathcal{W}$
**assert** $\mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma})$
$\mathcal{X} := \mathcal{X} \cup \{x\}$
$\mathcal{M} := \mathcal{M} \cup \{m : x \in \mathcal{T}_A[m]\}$
$\sigma := \mathsf{Adapt}(\mathsf{sk}_A, \hat{\sigma}, w)$
**return** $\sigma$

Fig. 22: Definition for $\mathcal{H}_4$ used in the proof of WUnf of Figure 9. Changes w.r.t. the previous hybrid are highlighted in gray.

that $\mathsf{Vrfy}(\mathsf{vk}, m, \sigma) = 1$. We just need to prove that $\mathsf{Vrfy}(\mathsf{vk}, m, \sigma') = 1$. Observe that $\mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma}) = 1$ implies that $\mathsf{e}(\hat{\sigma}, g) = \mathsf{e}(\mathsf{H}(m), \mathsf{vk}) \cdot \mathsf{e}(\mathsf{H}(m_{\mathsf{R}}), \mathsf{vk}_{\mathsf{R}})$. We know that the output $pm$ of $\mathsf{PreSign}_A$ must satisfy $\mathsf{e}(pm, g) = \mathsf{e}(\mathsf{H}(m), \mathsf{vk}_A) \cdot \mathsf{e}(\mathsf{H}(m_{\mathsf{R}}), \mathsf{vk}_{\mathsf{R}})$. Therefore, the output of line 4 of $\mathsf{OAdapt}$ as presented in Figure 22 is such that: $\mathsf{e}(\hat{\sigma} \cdot pm^{-1}, g) = \mathsf{e}(\hat{\sigma}, g) \cdot \mathsf{e}(pm^{-1}, g) = \mathsf{e}(\hat{\sigma}, g) \cdot \mathsf{e}(pm, g)^{-1}$. By the previous observations we obtain that

$$\begin{gathered}\mathsf{e}(\hat{\sigma}, g) \cdot \mathsf{e}(pm, g)^{-1} = \\ \mathsf{e}(\mathsf{H}(m), \mathsf{vk}) \cdot \mathsf{e}(\mathsf{H}(m_{\mathsf{R}}), \mathsf{vk}_{\mathsf{R}}) \cdot (\mathsf{e}(\mathsf{H}(m), \mathsf{vk}_A) \cdot \mathsf{e}(\mathsf{H}(m_{\mathsf{R}}), \mathsf{vk}_{\mathsf{R}}))^{-1} = \\ \mathsf{e}(\mathsf{H}(m), \mathsf{vk}) \cdot \mathsf{e}(\mathsf{H}(m_{\mathsf{R}}), \mathsf{vk}_{\mathsf{R}}) \cdot \mathsf{e}(\mathsf{H}(m), \mathsf{vk}_A)^{-1} \cdot \mathsf{e}(\mathsf{H}(m_{\mathsf{R}}), \mathsf{vk}_{\mathsf{R}})^{-1} = \\ \mathsf{e}(\mathsf{H}(m), \mathsf{vk}_A) \cdot \mathsf{e}(\mathsf{H}(m), \mathsf{vk}_E) \cdot \mathsf{e}(\mathsf{H}(m_{\mathsf{R}}), \mathsf{vk}_{\mathsf{R}}) \cdot \mathsf{e}(\mathsf{H}(m), \mathsf{vk}_A)^{-1} \cdot \mathsf{e}(\mathsf{H}(m_{\mathsf{R}}), \mathsf{vk}_{\mathsf{R}})^{-1} = \\ \mathsf{e}(\mathsf{H}(m), \mathsf{vk}_E)\end{gathered}$$

Then the signature $\sigma'$ computed as $\sigma' := \sigma_E \cdot \sigma_A$ satisfies $\mathsf{e}(\mathsf{H}(m), \mathsf{vk}_E) \cdot \mathsf{e}(\mathsf{H}(m), \mathsf{vk}_A) = \mathsf{e}(\mathsf{H}(m), \mathsf{vk})$ and thus $\mathsf{Vrfy}(\mathsf{vk}, m, \sigma') = 1$.

**Hybrid** $\mathcal{H}_6$ This hybrid is identical to $\mathcal{H}_5$ with the exception that we change oracle $\mathsf{OAdapt}$. In $\mathsf{OAdapt}$, instead of computing $pm$ by running $\mathsf{PreSign}_A$, we will do the same trick we did in Figure 22. We sample a

2P-AISO-AS-WUnf$_{\mathcal{A},\Theta}(n)$

$\mathcal{Q} := \mathcal{M} := \mathcal{W} := \emptyset$
$(\mathsf{vk}, (\mathsf{vk}_E, \mathsf{sk}_E), (\mathsf{vk}_A, \mathsf{sk}_A)) \leftarrow \mathsf{KeyGen}(1^n)$
$\mathcal{O} := \{\mathsf{OSign},$
$\quad \mathsf{OPreSign}_A, \mathsf{ONewStatement}, \mathsf{OAdapt}\}$
$(m, \sigma) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{vk}, (\mathsf{vk}_E, \mathsf{sk}_E), \mathsf{vk}_A)$
$c_1 := m \notin \mathcal{Q} \wedge m \notin \mathcal{M} \wedge \mathcal{T}_A[m] \cap \mathcal{X} = \emptyset$
$c_2 := \mathsf{Vrfy}(\mathsf{vk}, m, \sigma) = 1$
**return** $(c_1 \wedge c_2)$

$\mathsf{OSign}(m)$

$\mathcal{X} := \mathcal{X} \cup \mathcal{T}_A[m]$
$\mathcal{M} := \mathcal{M} \cup \{m\}$
**return** $\Pi\mathsf{Sign}(m)$

$\mathsf{ONewStatement}()$

$r \leftarrow_{\$} \mathbb{Z}_q$
$m_{\mathsf{R}} \leftarrow_{\$} \{0,1\}^*$
$\mathsf{sk}_{\mathsf{R}} := \mathsf{sk}_A + r$
$\mathsf{vk}_{\mathsf{R}} := \mathsf{vk}_A \cdot g^r$
$x := (m_{\mathsf{R}}, \mathsf{vk}_{\mathsf{R}})$
$w := \mathsf{Sign}(\mathsf{sk}_{\mathsf{R}}, m_{\mathsf{R}})$
$\mathcal{R}[x] := r$
$\mathcal{W} := \mathcal{W} \cup \{(x, w)\}$
**return** $x$

$\mathsf{OPreSign}_A(m, x)$

**assert** $\exists w \ : \ (x, w) \in \mathcal{W}$
$\mathcal{T}_A[m] := \mathcal{T}_A[m] \cup \{x\}$
**if** $x \in \mathcal{X} \vee m \in \mathcal{M}$ **then**
$\quad \mathcal{M} := \mathcal{M} \cup \{m\}$
$\quad \mathcal{X} := \mathcal{X} \cup \mathcal{T}_A[m]$
$(m_{\mathsf{R}}, \mathsf{vk}_{\mathsf{R}}) := x$
$m' \leftarrow_{\$} \{0,1\}^*$
$\mathsf{H}[m'] := \mathsf{H}(m) \cdot \mathsf{H}(m_{\mathsf{R}})$
$\mathcal{M}' := \mathcal{M}' \cup m'$
$\sigma' := \mathsf{Sign}(\mathsf{sk}_A + \mathsf{sk}_E, m')$
$r := \mathcal{R}[x]$
$\sigma_r := \mathsf{Sign}(r, m_{\mathsf{R}})$
$pm := \sigma' \cdot \sigma_r$
**return** $pm$

$\mathsf{OAdapt}(\hat{\sigma}, x, m)$

**assert** $\exists w \ : \ (x, w) \in \mathcal{W}$
**assert** $\mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma}) = 1$
$pm := \mathsf{PreSign}_A(m, x, w)$
$\sigma_E := \hat{\sigma} \cdot pm^{-1}$
$\sigma_A := \mathsf{Sign}(\mathsf{sk}_A, m)$
$\mathcal{M} := \mathcal{M} \cup \{m : x \in \mathcal{T}_A[m]\}$
$\sigma := \sigma_E \cdot \sigma_A$
**return** $\sigma$

Fig. 23: Definition for $\mathcal{H}_5$ used in the proof of $\mathsf{WUnf}$ of Figure 9. Changes w.r.t. the previous hybrid are highlighted in gray.

new random message $m'$ and program the random oracle at this message to be $\mathsf{H}(m') := \mathsf{H}(m) \cdot \mathsf{H}(m_{\mathsf{R}})$. The rest of the oracle remains the same. In Figure 24 we present the oracle.

*Claim.* $\mathcal{H}_5 \approx \mathcal{H}_6$

*Proof.* The proof is the same as the proof for $\mathcal{H}_3 \approx \mathcal{H}_4$.

**Reduction to EUF-CMA of BLS signatures**.

*Claim.* If BLS signatures satisfy EUF-CMA, then for all PPT adversaries $\mathcal{A}$ it holds that $\Pr[\mathsf{WUnf}_{\mathcal{A},\Theta}(n) = 1] \leq \mathsf{negl}(n)$.

*Proof.* Assume that there exists a PPT adversary $\mathcal{A}$ that wins $\mathcal{H}_5$ with non-negligible probability. Then we can create a PPT adversary that wins EUF-CMA against the BLS signature scheme.

First, the challenger of the EUF-CMA computes keys $(\mathsf{vk}, (\mathsf{vk}_A, \mathsf{sk}_A), (\mathsf{vk}_E, \mathsf{sk}_E)) \leftarrow \mathsf{KeyGen}(n)$ and sends $(\mathsf{vk}, \mathsf{vk}_E, \mathsf{vk}_A, \mathsf{sk}_E)$ to $\mathcal{B}$. The adversary $\mathcal{B}$ forwards $(\mathsf{vk}, \mathsf{vk}_E, \mathsf{vk}_A, \mathsf{sk}_E)$ to $\mathcal{A}$. Whenever $\mathcal{A}$ makes a query to the

2P-AISO-AS-WUnf$_{\mathcal{A},\Theta}(n)$

$\mathcal{Q} := \mathcal{M} := \mathcal{W} := \emptyset$
$(\mathsf{vk}, (\mathsf{vk}_E, \mathsf{sk}_E), (\mathsf{vk}_A, \mathsf{sk}_A)) \leftarrow \mathsf{KeyGen}(1^n)$
$\mathcal{O} := \{\mathsf{OSign},$
$\quad \mathsf{OPreSign}_A, \mathsf{ONewStatement}, \mathsf{OAdapt}\}$
$(m, \sigma) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{vk}, (\mathsf{vk}_E, \mathsf{sk}_E), \mathsf{vk}_A)$
$c_1 := m \notin \mathcal{Q} \wedge m \notin \mathcal{M} \wedge \mathcal{T}_A[m] \cap \mathcal{X} = \emptyset$
$c_2 := \mathsf{Vrfy}(\mathsf{vk}, m, \sigma) = 1$
**return** $(c_1 \wedge c_2)$

$\mathsf{OSign}(m)$

$\mathcal{X} := \mathcal{X} \cup \mathcal{T}_A[m]$
$\mathcal{M} := \mathcal{M} \cup \{m\}$
**return** $\Pi\mathsf{Sign}(m)$

$\mathsf{ONewStatement}()$

$r \leftarrow_\$ \mathbb{Z}_q$
$m_\mathsf{R} \leftarrow_\$ \{0,1\}^*$
$\mathsf{sk}_\mathsf{R} := \mathsf{sk}_A + r$
$\mathsf{vk}_\mathsf{R} := \mathsf{vk}_A \cdot g^r$
$x := (m_\mathsf{R}, \mathsf{vk}_\mathsf{R})$
$w := \mathsf{Sign}(\mathsf{sk}_\mathsf{R}, m_\mathsf{R})$
$\mathcal{R}[x] := r$
$\mathcal{W} := \mathcal{W} \cup \{(x, w)\}$
**return** $x$

$\mathsf{OPreSign}_A(m, x)$

**assert** $\exists w : (x, w) \in \mathcal{W}$
$\mathcal{T}_A[m] := \mathcal{T}_A[m] \cup \{x\}$
**if** $x \in \mathcal{X} \vee m \in \mathcal{M}$ **then**
$\quad \mathcal{M} := \mathcal{M} \cup \{m\}$
$\quad \mathcal{X} := \mathcal{X} \cup \mathcal{T}_A[m]$
$(m_\mathsf{R}, \mathsf{vk}_\mathsf{R}) := x$
$m' \leftarrow_\$ \{0,1\}^*$
$\mathsf{H}[m'] := \mathsf{H}(m) \cdot \mathsf{H}(m_\mathsf{R})$
$\mathcal{M}' := \mathcal{M}' \cup m'$
$\sigma' := \mathsf{Sign}(\mathsf{sk}_A + \mathsf{sk}_E, m')$
$r := \mathcal{R}[x]$
$\sigma_r := \mathsf{Sign}(r, m_\mathsf{R})$
$pm := \sigma' \cdot \sigma_r$
**return** $pm$

$\mathsf{OAdapt}(\hat{\sigma}, x, m)$

**assert** $\exists w : (x, w) \in \mathcal{W}$
**assert** $\mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma}) = 1$
$(m_\mathsf{R}, \mathsf{vk}_\mathsf{R}) := x$
$m' \leftarrow_\$ \{0,1\}^*$
$\mathsf{H}[m'] := \mathsf{H}(m) \cdot \mathsf{H}(m_\mathsf{R})$
$\sigma' := \mathsf{Sign}(\mathsf{sk}_A + \mathsf{sk}_E, m')$
$r := \mathcal{R}[x]$
$\sigma_r := \mathsf{Sign}(r, m_\mathsf{R})$
$pm := \sigma' \cdot \sigma_r$
$\sigma_E := \hat{\sigma} \cdot pm^{-1}$
$\sigma_A := \mathsf{Sign}(\mathsf{sk}_A, m)$
$\mathcal{M} := \mathcal{M} \cup \{m : x \in \mathcal{T}_A[m]\}$
$\sigma := \sigma_E \cdot \sigma_A$
**return** $\sigma$

Fig. 24: Definition for $\mathcal{H}_6$ used in the proof of WUnf of Figure 9. Changes w.r.t. the previous hybrid are highlighted in gray.

signing oracles, $\mathcal{B}$ simply forwards this query to the challenger of the EUF-CMA. Whenever $\mathcal{A}$ makes a query to ONewStatement, $\mathcal{B}$ runs the code as described in Figure 25 with one exception. Instead of computing $w$, the oracle sets it to 0. Thus, it does not compute a signature on $m_\mathsf{R}$ using $\mathsf{sk}_A$. As a consequence $\mathcal{B}$ will not know a valid witness for $x$. This creates a problem in $\mathsf{OPreSign}_A$ and OAdapt, as the witness is required to compute them. However, the programming of the random oracle will overcome this in both $\mathsf{OPreSign}_A$ and OAdapt. Whenever $\mathcal{A}$ makes a query to $\mathsf{OPreSign}_A$, $\mathcal{B}$ runs the oracle code described in Figure 23 replacing calls to Sign using $\mathsf{sk}_A$ with calls to the signing oracles provided by the challenger. Crucially, $\mathcal{B}$ will never query for a signature on the message $m$ queried by $\mathcal{A}$. Instead, $\mathcal{B}$ queries for a signature on a totally random new message $m'$, which gets added to the set of queried messages instead of $m$. However, as seen in $\mathcal{H}_5$, the programming on the random oracle is set up to then produce a valid $pm$. Whenever $\mathcal{A}$ makes a query to OAdapt, $\mathcal{B}$ runs the oracle code described in Figure 22, replacing calls to Sign using $\mathsf{sk}_A$ with calls to the signing oracles provided by the challenger. This causes the messages queried to OAdapt to be included in the set $\mathcal{Q}$ of the challenger. However, any message queried to OAdapt is added to the set $\mathcal{M}$ and, as we will see, the forgery message of $\mathcal{A}$ cannot be an element of $\mathcal{M}$. In other words, queries to OAdapt trigger queries to the signing oracles but only on messages that are not the forgery message.

$\mathcal{B}(\mathsf{vk}, \mathsf{vk}_A, \mathsf{vk}_E, \mathsf{sk}_A)$

$\mathcal{Q} := \mathcal{M} := \mathcal{W} := \emptyset$
$\mathcal{O} := \{\mathsf{OSign},$
$\quad \mathsf{OPreSign}_A, \mathsf{ONewStatement}, \mathsf{OAdapt}\}$
$(m, \sigma) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{vk}, (\mathsf{vk}_E, \mathsf{sk}_E), \mathsf{vk}_A)$
**assert** $m \notin \mathcal{Q} \wedge m \notin \mathcal{M} \wedge \mathcal{T}_A[m] \cap \mathcal{X} = \emptyset$
**assert** $\mathsf{Vrfy}(\mathsf{vk}, m, \sigma) = 1$
**return** $(m, \sigma)$

$\mathsf{OSign}(m)$

$\mathcal{X} := \mathcal{X} \cup \mathcal{T}_A[m]$
$\mathcal{M} := \mathcal{M} \cup \{m\}$
**return** $EUF - CMA.\mathsf{OSign}(m)$

$\mathsf{ONewStatement}()$

$r \leftarrow_\$ \mathbb{Z}_q$
$m_\mathsf{R} \leftarrow_\$ \{0,1\}^*$
$\mathsf{sk}_\mathsf{R} := \mathsf{sk}_A + r$
$\mathsf{vk}_\mathsf{R} := \mathsf{vk}_A \cdot g^r$
$x := (m_\mathsf{R}, \mathsf{vk}_\mathsf{R})$
$w := 0$
$\mathcal{R}[x] := r$
$\mathcal{W} := \mathcal{W} \cup \{(x, w)\}$
**return** $x$

$\mathsf{OPreSign}_A(m, x)$

**assert** $\exists w \ : \ (x, w) \in \mathcal{W}$
$\mathcal{T}_A[m] := \mathcal{T}_A[m] \cup \{x\}$
**if** $x \in \mathcal{X} \vee m \in \mathcal{M}$ **then**
$\quad \mathcal{M} := \mathcal{M} \cup \{m\}$
$\quad \mathcal{X} := \mathcal{X} \cup \mathcal{T}_A[m]$
$(m_\mathsf{R}, \mathsf{vk}_\mathsf{R}) := x$
$m' \leftarrow_\$ \{0,1\}^*$
$\mathsf{H}[m'] := \mathsf{H}(m) \cdot \mathsf{H}(m_\mathsf{R})$
$\mathcal{M}' := \mathcal{M}' \cup m'$
$\sigma' := EUF - CMA.\mathsf{OSign}(m')$
$r := \mathcal{R}[x]$
$\sigma_r := \mathsf{Sign}(r, m_\mathsf{R})$
$pm := \sigma' \cdot \sigma_r$
**return** $pm$

$\mathsf{OAdapt}(\hat{\sigma}, x, m)$

**assert** $\exists w \ : \ (x, w) \in \mathcal{W}$
**assert** $\mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma}) = 1$
$(m_\mathsf{R}, \mathsf{vk}_\mathsf{R}) := x$
$m' \leftarrow_\$ \{0,1\}^*$
$\mathsf{H}[m'] := \mathsf{H}(m) \cdot \mathsf{H}(m_\mathsf{R})$
$\sigma' := EUF - CMA.\mathsf{OSign}(m')$
$r := \mathcal{R}[x]$
$\sigma_r := \mathsf{Sign}(r, m_\mathsf{R})$
$pm := \sigma' \cdot \sigma_r$
$\sigma_E := \hat{\sigma} \cdot pm^{-1}$
$\sigma_A := \mathsf{Sign}(\mathsf{sk}_A, m)$
$\mathcal{M} := \mathcal{M} \cup \{m : x \in \mathcal{T}_A[m]\}$
$\sigma := \sigma_E \cdot \sigma_A$
**return** $\sigma$

Fig. 25: Definition for $\mathcal{H}_6$ used in the proof of WUnf of Figure 9. Changes w.r.t. the previous hybrid are highlighted in gray.

Finally, $\mathcal{A}$ returns a forgery $(m, \sigma)$ and $\mathcal{B}$ sends this as a forgery to the challenger of the EUF-CMA experiment of BLS. By the assumption that $\mathcal{A}$ wins $\mathcal{H}_5$, we know that (i) $\mathsf{Vrfy}(\mathsf{vk}, m, \sigma) = 1$ and (ii) $m \notin \mathcal{Q} \wedge m \notin \mathcal{M} \wedge \mathcal{T}_A[m] \cap \mathcal{X} = \emptyset$. From these facts we make the following observations. If $\mathcal{T}_A[m] = \emptyset$, then the adversary never queried $\mathsf{OPreSign}_A$ on the forgery message. Since $m \notin \mathcal{M}$, then we know that $\mathcal{A}$ did not query the forgery message to OAdapt nor to OSign. Thus, The forgery message was not added to the set $\mathcal{Q}$ of the challenger. Therefore, if $\mathcal{A}$ wins $\mathcal{H}_6$ with non-negligible probability, $\mathcal{B}$ wins the EUF-CMA of BLS signatures with non-negligible probability.

## G Security Analysis Of BLS-Based 2P-AISO-AS Construction

In this section we prove Theorem 2 about the security of the BLS-based construction of 2P-AISO-AS from Figure 10. To this end, we enunciate a lemma about each of the security properties defined in Section 4 and prove those lemmas.

**Lemma 7 (Weak unforgeability of Figure 10).** *If* DS *is unforgeable, then the construction of Figure 10 satisfies weak unforgeability.*

*Proof.* The proof follows in a series of hybrids, which we describe below. Let $\mathcal{H}_i \approx \mathcal{H}_{i+1}$ denote $|\Pr[\mathcal{H}_i = 1] - \Pr[\mathcal{H}_{i+1} = 1]| \leq \mathsf{negl}(n)$.

**Hybrid** $\mathcal{H}_0$: This hybrid corresponds to the original experiment 2P-AISO-AS-$WUnf_{\mathcal{A},\Theta}(n)$ (Figure 6).

**Hybrid** $\mathcal{H}_1$: In this hybrid, we modify the oracles $\mathsf{OPreSign}_A$ and $\mathsf{OAdapt}$. Whenever the oracle $\mathsf{OPreSign}_A$ has to compute the signature $\sigma_A$ to store it in the internal protocol state $st$, $\mathcal{H}_1$ instead does not compute this signature and stores the signing key in the protocol state. Then, on queries to $\mathsf{OAdapt}$, 2hyb2 uses the signing key stored in $st$ to compute at this time $\sigma_A$. The rest remains unchanged.

*Claim.* $\mathcal{H}_0 \approx \mathcal{H}_1$

*Proof.* The only changes in $\mathcal{H}_1$ are in the order in which $\mathcal{H}_1$ computes an internal value, but the behaviour of $\mathcal{H}_1$ is trivially the same as $\mathcal{H}_0$.

**Reduction to** $\mathsf{EUF-CMA}$:

*Claim.* If there exists a $\mathsf{PPT}$ adversary that wins $\mathcal{H}_1$ with non-negligible probability, then we can construct a $\mathsf{PPT}$ adversary $\mathcal{B}$ that wins the $\mathsf{EUF-CMA}$of BLS signatures with non-negligible probability.

*Proof.* the challenger plays $\mathsf{EUF-CMA}$against $\mathcal{B}$ and $\mathcal{B}$ internally plays $\mathcal{H}_1$ against $\mathcal{A}$. In the reduction, first the challenger generates the keys and gives $(\mathsf{vk}, \mathsf{vk}_A, \mathsf{vk}_E, \mathsf{sk}_E)$ to $\mathcal{B}$ and $\mathcal{B}$ forwards these keys to $\mathcal{A}$. Whenever $\mathcal{A}$ queries $\mathsf{OSign}$, $\mathcal{B}$ forwards the query to the oracle $\mathsf{OSign}$ that the challenger provides. Whenever $\mathcal{A}$ queries $\mathsf{ONewStatement}$, $\mathsf{OPreSign}_A$, $\mathcal{B}$ behaves honestly as in $\mathcal{H}_1$. Whenever $\mathcal{A}$ queries $\mathsf{OAdapt}$, $\mathcal{B}$has to query the signing oracle of the challenger to obtain $\sigma_A$, but behaves as in $\mathcal{H}_1$ for the rest. Finally, when $\mathcal{A}$ outputs a forgery $(m^*, \sigma^*)$, $\mathcal{B}$ copies this forgery as their own output.

By assumption, $\mathcal{A}$ wins $\mathcal{H}_1$, thus the forgery $(m^*, \sigma^*)$ satisfies: (i) $\mathsf{Vrfy}(\mathsf{vk}, m^*, \sigma^*) = 1$, (ii) $m^*$ was never queried to oracles $\mathsf{OSign}$ and $\mathsf{OAdapt}$ from $\mathcal{B}$ ($m^* \notin \mathcal{M}$). Therefore, by our reduction, we know that the output forgery was never queried to $\mathsf{OSign}$ of the challenger and that the forgery verifies. Thus, $\mathcal{B}$ wins the $\mathsf{EUF-CMA}$ game with the same probability that $\mathcal{A}$ triggers $\mathsf{Bad}_1$.

**Lemma 8 (Full Extractability Of Figure 10).** *If* $\mathsf{VWER}$ *is verifiable and* $\mathsf{DS}$ *is unforgeable, then the construction of Figure 10 satisfies full extractability.*

*Proof.* The proof follows in a series of hybrids, which we describe below.

**Hybrid** $\mathcal{H}_0$: This hybrid corresponds to the original experiment 2P-AISO-AS-$\mathsf{FExt}_{\mathcal{A},\Theta}(n)$ (Figure 5).

**Hybrid** $\mathcal{H}_1$: In this hybrid, the challenger stores messages queried to the presign oracle into a set $\hat{\mathcal{Q}}$ if the presignature preverifies. If the forgery message $m$ is an element of $\hat{\mathcal{Q}}$, the hybrid aborts.

*Claim.* By the verifiability of $\mathsf{VWER}$, for all $\mathsf{PPT}$ adversaries $\mathcal{A}$, it holds that $\Pr[\mathcal{H}_1^{\mathcal{A}}(n) = 1] = \Pr[\mathcal{H}_0^{\mathcal{A}}(n) = 1]$.

*Proof.* If there exists a $\mathsf{PPT}$ adversary $\mathcal{A}$ that wins $\mathcal{H}_0$ but does not win $\mathcal{H}_1$, then we can find an adversary that breaks the verifiability of $\mathsf{VWER}$. First, note that condition $c_3$ is satisfied if either $\mathcal{A}$ never queries the presign oracle or if none of the presignatures for the forgery message succeed extracting a witness from the forgery signature. If $\mathcal{A}$ does not query the presign oracle, then $\mathcal{H}_1$ is a faithful computation of $\mathcal{H}_0$. If an adversary wins in $\mathcal{H}_0$ but not in $\mathcal{H}_1$ it must be the case that they have queried the presign oracle on the forgery message. The presign oracle aborts if the resulting presignature does not preverify. This implies that all presignatures stored in $\mathcal{T}$ must preverify. A presignature is a tuple of (i) a $\mathsf{VWER}$ ciphertext and proof, and (ii) a $\mathsf{DS}$ signature. Component (i) is provided by the adversary. If the presignature preverifies, it implies that the proof of component (i) verifies. However, if there is no presignature for which extraction of a valid witness succeeds, then it must be the case that, for one of the presignatures on the forgery message

the adversary was able to find a VWER proof that verifies while the VWER ciphertext does not encrypt the witness.

Therefore we can create a PPT adversary $\mathcal{B}$ such that, using $\mathcal{A}$, $\mathcal{B}$ wins the verifiability of the VWER. Adversaries $\mathcal{B}$ starts by generating keys on their own and sending them to $\mathcal{A}$. Whenever $\mathcal{A}$ queries the signing oracles and presigning oracles, $\mathcal{B}$ behaves honestly and interact in the protocols. Finally, when $\mathcal{A}$ presents the forgery, $\mathcal{B}$ picks a presignature from $\mathcal{T}[m]$ at random. From this presignature, $\mathcal{B}$ obtains the VWER ciphertext and proof, which is the output to its challenger. If $\mathcal{A}$ wins $\mathcal{H}_0$ with non-negligible probability by querying a presignature on the forgery message, then either $\mathcal{B}$ breaks the verifiability of VWER with non-negligible probability.

**Reduction to unforgeability of $\mathsf{DS}_2$:** We want to show that, if there exists an adversary $\mathcal{A}$ that wins in $\mathcal{H}_1$, then we can use this adversary to create another PPT adversary $\mathcal{B}$ that wins the unforgeability game of the underlying signature scheme $\mathsf{DS}_2$. To this end, $\mathcal{B}$ needs to provide $\mathcal{A}$ with oracles for computing signatures and presignatures, while $\mathcal{B}$ only has access to a signing oracle themself. Then, we will argue about the winning conditions $c_1$, $c_2$ and $c_3$. We will show that if $\mathcal{A}$ finds $\sigma$ that satisfies these conditions, then $\sigma$ is a valid forgery that $\mathcal{B}$ can use to win the unforgeability game.

*Claim.* If there exists a PPT adversary $\mathcal{A}$ such that $\Pr[\mathcal{H}_1^{\mathcal{A}}(n) = 1] > \mathsf{negl}(n)$, then we can construct a PPT adversary $\mathcal{B}$ such that $\Pr[\mathsf{Sig\text{-}Forge}_{\mathcal{B},\mathsf{DS}_2}(n) = 1] > \mathsf{negl}(n)$.

*Proof.* The adversary $\mathcal{B}$ begins by generating the keys $(\mathsf{vk}, (\mathsf{vk}_A, \mathsf{sk}_A))$ and forwarding these keys to $\mathcal{A}$. Whenever $\mathcal{A}$ queries the oracle $\mathsf{OSign}$, $\mathcal{B}$ forwards the query to their challenger and forwards the result of the query back to $\mathcal{A}$. If $\mathcal{A}$ queries $(m, x, pm)$ to oracle $\mathsf{OPreSign}$, $\mathcal{B}$ starts by parsing $(c, \pi) := pm$ and asserting whether the proof verifies $\mathsf{VfEncR}(c, \pi, (\mathsf{vk}, m), x) = 1$. If the assertion does not pass, then we know that the assertion from line 2 of $\mathsf{OPreSign}_E$ in Figure 5 would also fail. If the assertion passes, $\mathcal{B}$ queries the signing oracle provided by its challenger in order to obtain a signature. From this signature, $\mathcal{B}$ computes the presignature honestly and returns it to $\mathcal{A}$. Finally, $\mathcal{A}$ presents a forgery, which $\mathcal{B}$ forwards to its challenger. If $\mathcal{A}$ wins with non-negligible probability, it must be the case that (i) $\mathcal{A}$ did not query the forgery message to the signing oracles, (ii) the forgery verifies and (iii) $\mathcal{A}$ did not query the presign oracle. This last statement ensures that the forgery message was never added to $\mathcal{Q}$ by means of $\mathcal{B}$ querying the signing oracles in order to compute a presignature. Therefore, if $\mathcal{A}$ wins with non-negligible probability, then $\mathcal{B}$ also wins with non-negigible probability.

**Lemma 9 (Designated Adaptability Of Figure 10).** *If BLS signatures are correct then the construction of Figure 10 satisfies designated adaptability.*

*Proof.* Assume that, for a given message $m$, an instance $(x, w) \in \mathsf{R}$ and a presignature $\hat{\sigma}$, it holds that $\mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma}) = 1$ but for all protocol states $st$ it holds that $\mathsf{Vrfy}(\mathsf{vk}, m, \mathsf{Adapt}(\mathsf{vk}, \hat{\sigma}, w)) \neq 1$. Then the construction of Figure 10 would not be correct.

The presignature $\hat{\sigma}$ is a tuple $(c, \pi, \sigma_E)$. From $\mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma}) = 1$ we know that the following holds: (i) the VWER proof verifies $\mathsf{VfEncR}((c, \pi), (\mathsf{vk}, m), x) = 1$; and (ii) the signature $\sigma_E$ is a valid signature of $m$ under $\mathsf{vk}_E$ $\mathsf{Vrfy}(\mathsf{vk}_E, \sigma_E, m)$.

The $\mathsf{Adapt}$ algorithm uses $\sigma_E$ from the presignature and $\sigma_A$ stored in the internal state $st$ to compute $\mathrm{Agg}(\sigma_A, \sigma_E)$. If or all protocol states $st$ it holds that $\mathsf{Vrfy}(\mathsf{vk}, m, \mathsf{Adapt}(\mathsf{vk}, \hat{\sigma}, w)) \neq 1$, it means that for all possible signatures $\sigma_A$ it holds that $\mathsf{Vrfy}(\mathsf{vk}, m, \mathrm{Agg}(\sigma_A, \sigma_E)) \neq 1$.

The BLS signatures have the property that it $\mathsf{Vrfy}(\mathsf{vk}_1, m, \sigma_1) = 1$ and $\mathsf{Vrfy}(\mathsf{vk}_2, m, \sigma_2) = 1$, then $\mathsf{Vrfy}(\mathrm{Agg}(\mathsf{vk}_1, \mathsf{vk}_2), m, \mathrm{Agg}(\sigma_1, \sigma_2)) = 1$. We can write $\mathsf{vk} := \mathrm{Agg}(\mathsf{vk}_E, \mathsf{vk}_A)$ for some verification key $\mathsf{vk}_A$ with a corresponding signing key $\mathsf{sk}_A$. By the correctness of BLS signatures, we know that in fact there exists a signature $\sigma_A$ such that $\mathsf{Vrfy}(\mathsf{vk}_A, m, \sigma_A)$. Therefore, if $st := \sigma_A$, we have found a protocol state that satisfies $\mathsf{Vrfy}(\mathsf{vk}, m, \mathsf{Adapt}(\mathsf{vk}, \hat{\sigma}, w)) = 1$.

**Lemma 10 (Unlinkability Of Figure 10).** *If BLS signatures are unique and the construction of Figure 10 is correct, then the construction of Figure 10 satisfies unlinkability.*

*Proof.* Assume that there exists PPT adversary $\mathcal{A}$ such that $|\Pr[\mathsf{Unlinkability}_{\mathcal{A},\Theta}(n,0)] - \Pr[\mathsf{Unlinkability}_{\mathcal{A},\Theta}(n,1)]| > \mathsf{negl}(n)$. This implies that $\mathcal{A}$ can distinguish between the distribution of signatures computed directly with a sign algorithm: $D_1 := \{\mathsf{Agg}(\mathsf{Sign}(\mathsf{sk}_A, m), \mathsf{Sign}(\mathsf{sk}_E, m))\}$ and the distribution of adapted signatures $D_2 := \{\mathsf{Adapt}(\mathsf{vk}, \hat{\sigma}, w)\}$ where $(x, w) \leftarrow \mathsf{GenR}()$ and $\hat{\sigma} := \Pi\mathsf{PreSign}_{\langle(\mathsf{sk}_A, w), \mathsf{sk}_E\rangle}(\mathsf{vk}, m, x)$.

By the uniqueness of BLS signatures, we get that for any message $m$ and any verification key $\mathsf{vk}$ if we get two signatures $\sigma_1$ and $\sigma_2$ such that $\mathsf{Vrfy}(\mathsf{vk}, m, \sigma_1) = \mathsf{Vrfy}(\mathsf{vk}, m, \sigma_2) = 1$ then we have that $\sigma_1 = \sigma_2$.

Therefore the signatures generated in both $D_1$ and $D_2$ are identical. Thus both distributions are also identical.

**Lemma 11 (Computational Preverify Soundness Of Figure 10).** *If* VWER *is verifiable and* DS *is correct, then the construction of Figure 10 satisfies computational preverify soundness.*

*Proof.* Assume that there exists a polynomially bounded $x \notin L_{\mathsf{R}}$ and a PPT adversary $\mathcal{A}$ that on input honestly generated keys outputs a pair $(m, \hat{\sigma})$ such that $\mathsf{PreVrfy}(\mathsf{vk}, \hat{\sigma}, m, x) = 1$ with non-negligible probability. Then we can find an adversary $\mathcal{B}$ that using $\mathcal{A}$ breaks the verifiability of the VWER scheme. The adversary $\mathcal{B}$ generates keys $(\mathsf{vk}, (\mathsf{vk}_E, \mathsf{sk}_E), (\mathsf{vk}_A, \mathsf{sk}_E))$ and forwards them to $\mathcal{A}$. The adversary $\mathcal{A}$ returns a tuple $(m, \hat{\sigma})$. Next, $\mathcal{B}$ computes $\sigma := \mathsf{Sign}(\mathsf{sk}_A, m) \cdot \mathsf{Sign}(\mathsf{sk}_E, m)$. Finally, $\mathcal{B}$ outputs the tuple $(m, \mathsf{vk}, \sigma, c, \pi, x)$.

The presignature is a tuple $\hat{\sigma} := (c, \pi, \sigma_E)$. This presignature, by our assumption, satisfies $\mathsf{PreVrfy}(\mathsf{vk}, m, \hat{\sigma}, x) = 1$ with non-negligible probability. This last fact, by the definition of the PreVrfy algorithm in Figure 10, implies that $\mathsf{VfEncR}((c, \pi), (\mathsf{vk}, m), x) = 1$. The signature $\sigma$ satisfies $\mathsf{Vrfy}(\mathsf{vk}, m, \sigma) = 1$. Finally, by our assumption, $x \notin L_{\mathsf{R}}$ and thus $(x, \mathsf{DecR}(\sigma, (c, \pi))) \notin \mathsf{R}$. Therefore, $\mathcal{B}$ wins the verifiability game of the VWER scheme with the same probability that $\mathcal{A}$ breaks the pre-verify soundness.

**Lemma 12 (One-way presignatures of Figure 10).** *If* VWER *is zero-knowledge and* R *is $q$-hard, then the construction of Figure 10 has one-way presignatures.*

*Proof.* The proof follows in a series of hybrids, which we describe below. Let $\mathcal{H}_i \approx \mathcal{H}_{i+1}$ denote $|\Pr[\mathcal{H}_i = 1] - \Pr[\mathcal{H}_{i+1} = 1]| \leq \mathsf{negl}(n)$.

**Hybrid** $\mathcal{H}_0$: This hybrid corresponds to the original experiment $\mathsf{OW\text{-}PreSign}_{\mathcal{A},\Theta}(n)$ (Figure 8).

**Hybrid** $\mathcal{H}_1$: In this hybrid, whenever the adversary makes a query $(x, m)$ to $\mathsf{OPreSign}_A$ we replace $\mathsf{EncR}((\mathsf{vk}, m), w)$ with the simulator $\mathsf{EncRSim}$ of the VWER: $\mathsf{EncRSim}((\mathsf{vk}, m))$.

*Claim.* If VWER is zero-knowledge, then $\mathcal{H}_0 \approx \mathcal{H}_1$.

*Proof.* If there exists a PPT adversary $\mathcal{A}$ that distinguishes between $\mathcal{H}_0$ and $\mathcal{H}_1$ then we can create a reduction $\mathcal{B}$ that using $\mathcal{A}$ breaks the zero-knowledge of VWER. The challenger of the zero-knowledge game starts by generating keys $(\mathsf{vk}, (\mathsf{vk}_A, \mathsf{sk}_A), (\mathsf{vk}_E, \mathsf{sk}_E))$ ans sends $(\mathsf{vk}, \mathsf{vk}_A, \mathsf{vk}_E, \mathsf{sk}_A)$ to $\mathcal{B}$. Then $\mathcal{B}$ behaves just as $\mathcal{H}_0$ with the exception that, whenever $\mathcal{A}$ makes a query $(x, m)$ to $\mathsf{OPreSign}_A$, instead of running $\mathsf{EncR}((\mathsf{vk}, m), w)$, $\mathcal{B}$ will query the oracle $\mathsf{OEncR}$ of the challenger on input $(m)$. The oracle will reply with $(c, \pi)$ and $\mathcal{B}$ will use these to run the rest of $\mathsf{PreSign}_A$ honestly.

Notice that, in the zero-knowledge game of VWER, if $b = 0$ then $\mathcal{B}$ is running a faithful computation of $\mathcal{H}_0$, otherwise, $\mathcal{B}$ would be running a faithful computation of $\mathcal{H}_1$. If $\mathcal{A}$ can distinguish between $\mathcal{H}_0$ and $\mathcal{H}_1$ then $\mathcal{B}$ can use the output of $\mathcal{A}$ to distinguish between $b = 0$ or $b = 1$ breaking the zero-knowledge of VWER.

**Reduction to $q$-hardness of** R: By this point, we have removed all information about the witnesses during presignature generation

*Claim.* If R satisfies $q$-hardness, then for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(n)$ such that $\Pr[\mathcal{H}_1^{\mathcal{A}}(n) = 1] \leq \mathsf{negl}(n)$.

*Proof.* The challenger of the $q$-hardness begins by running $q$ times $(x, w) \leftarrow \mathsf{GenR}(n)$ and sends all statements $x_i$ for $i \in [q]$ to $\mathcal{B}$. Internally, $\mathcal{B}$ plays $\mathcal{H}_1$ honestly with $\mathcal{A}$. Whenever $\mathcal{A}$ makes the $i$'th query to

$\mathsf{ONewStatement}$, $\mathcal{B}$ replies with the $i$'th statement that the challenger sent at the beginning of the reduction. At the end of $\mathcal{H}_1$, $\mathcal{A}$ outputs $w^*$ and $\mathcal{B}$ sets this witness as their output as well.

If $\mathcal{A}$ wins the one-way presignatures game with non-negligible probability, then it must be the case that for one of the statements $x^*$ responded by $\mathsf{ONewStatement}$ it holds that $(x^*, w^*) \in \mathsf{R}$, and therefore $\mathcal{B}$ also wins the $q$-hardness with the same probability of $\mathcal{A}$.

# H Generic Construction for 2P-AISO-AS

Here, we describe our generic construction for 2P-AISO-AS. We first shortly overview the required cryptographic building blocks and defer full descriptions

## H.1 Construction Details

Our construction is presented in Figure 26. It is defined w.r.t. a signature scheme $\mathsf{DS}_2$, a hard relation $\mathsf{R}$, and the identity function $f(w) := w$. It relies on a public key encryption scheme $\mathsf{PKE}$, a secret key encryption scheme $\mathsf{SKE}$, a hash function $\mathsf{H}$, a UC-secure two-party secure computation protocol $\Pi_{\mathsf{2PC}}$ (as given in [37]), and a non-interactive zero-knowledge scheme $\mathsf{NIZK}$. We use the NIZK scheme to prove statements in the following language $L^*$:

$$L^* := \left\{ \begin{array}{l} x^* := (ct_\sigma, ct_w, \mathsf{vk}, m, x), w^* := (\sigma, h, w, r_1, r_2) \ \text{ s.t.} \\ ct_\sigma := \mathsf{PKE.Enc}(x, \sigma; r_1) \ \wedge \ ct_w := \mathsf{SKE.Enc}(h, w; r_2) \\ \wedge \ \mathsf{DS}_2.\mathsf{Vrfy}(\mathsf{vk}, m, \sigma) = 1 \ \wedge \ (x, w) \in R \ \wedge \ h = \mathsf{H}(\sigma) \end{array} \right\}$$

The presignature protocol is a 2PC for the following functionality. First, it computes a fresh signature on the message. Then, it encrypts the signature into a ciphertext $ct_\sigma$ using the statement $x$[8] as the encryption key. Next, the protocol encrypts the witness into a ciphertext $ct_w$. The symmetric key for this encryption is the hash of the signature $h := \mathsf{H}(\sigma)$. Finally, the protocol computes a proof for the language described above to ensure that the computation is correct. The presignature is the tuple $(ct_\sigma, ct_w, \pi)$. The preverify algorithm simply verifies the proof $\pi$. The adapt algorithm uses the witness $w$ as the decryption key of the public-key encryption in order to decrypt $ct_\sigma$ into $\sigma$. Finally, the extract algorithm, computes the hash of the signature $h := \mathsf{H}(\sigma)$. Then, it uses the hash as the symmetric key decryption key in order to decrypt $ct_w$ into $w$.

**Theorem 4.** *If $\Pi_{\mathsf{2PC}}$ is UC-secure, $\mathsf{DS}_2$ is unforgeable, $\mathsf{PKE}$ and $\mathsf{SKE}$ are IND-CPA secure, $\mathsf{R}$ satisfies q-hardness and $\mathsf{NIZK}$ has computational zero-knowledge and is knowledge-sound, then the construction in Figure 26 satisfies full extractability against Adaptor, weak extractability against Extractor, unlinkability and computational preverify soundness.*

## H.2 Security Analysis

In this section we prove Theorem 4 about the security of the generic construction of 2P-AISO-AS from Figure 26. To this end, we enunciate a lemma about each of the security properties defined in Section 4 and prove those lemmas.

**Lemma 13 (Presignature Correctness Of Figure 26).** *If $\Pi_{\mathsf{2PC}}$ is UC-secure, $\mathsf{DS}_2$, $\mathsf{SKE}$ and $\mathsf{PKE}$ are corect and $\mathsf{NIZK}$ is complete, then the construction in Figure 26 satisfies presignature correctness.*

[8] We note that we consider plain PKE scheme here, and hence, the relation becomes $(x := \mathsf{ek}, w := \mathsf{dk}) \in R \iff (\mathsf{ek}, \mathsf{dk}) \in [\mathsf{PKE.KeyGen}(1^n)]$. We can make use of more expressive relations by replacing the PKE scheme with witness encryption.

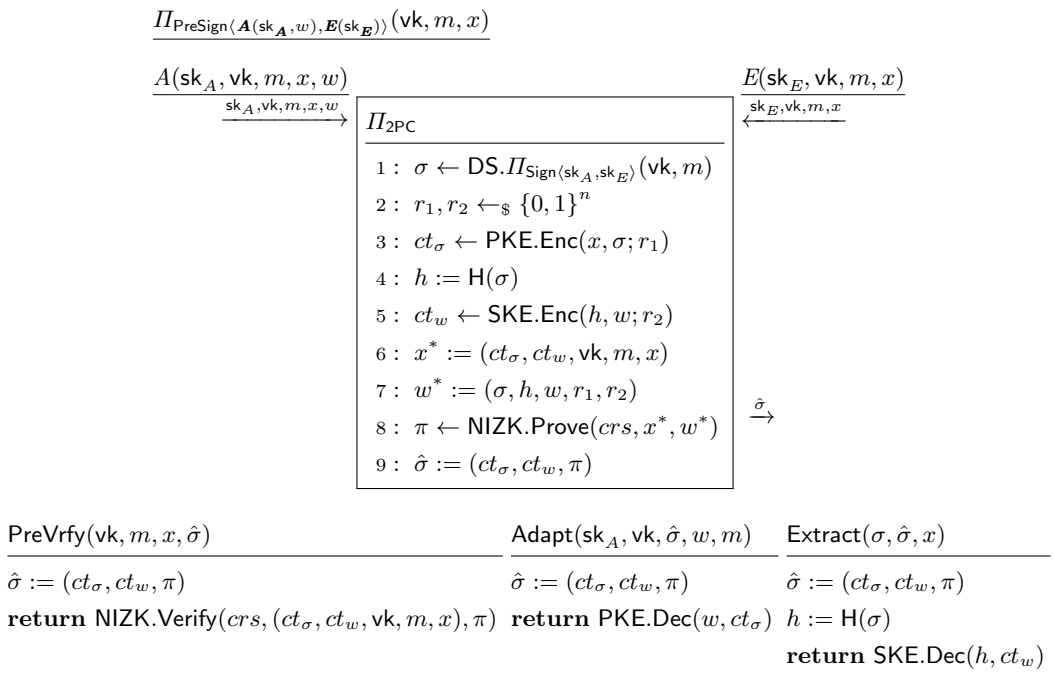



Fig. 26: Our construction for 2P-AISO-AS.

*Proof.* From honestly generated keys $(\mathsf{vk}, (\mathsf{vk}_A, \mathsf{sk}_A)(\mathsf{vk}_E, \mathsf{sk}_E))$ and honestly generated statement and witness $(x, w)$, the presignature $\hat{\sigma}$ resulting from honestly computing a presign protocol is a tuple $(ct_\sigma, ct_w, \pi)$. By the correctness of the two-party computation. $ct_\sigma$ is a ciphertext to the signature $\sigma$ under the public-key $x$; $ct_w$ is a ciphertext to the witness $w$ under the symmetric key $h := \mathsf{H}(\sigma)$; and $\pi$ is a proof of knowledge of the witness $(\sigma, h, w, r_1, r_2)$ satisfying the properties of the language abovementioned. By the completeness of the NIZK, $b_0$ is satisfied; by the correctness of the public-key encryption scheme and the signature scheme, $b_1$ is satisfied; and by the correctness of symmetric-key encryption scheme and the hard relation, $b_2$ is satisfied.

**Lemma 14 (Weak Unforgeability Of Figure 26).** *If $\Pi_{\mathsf{2PC}}$ is UC-secure, $\mathsf{DS}_2$ is unforgeable, PKE and SKE are IND-CPA secure, R satisfies q-hardness and NIZK has computational zero-knowledge, then the construction in Figure 26 satisfies weak unforgeability.*

*Proof.* The proof follows in a series of hybrids, which we describe below. Let $\mathcal{H}_i \approx \mathcal{H}_{i+1}$ denote $|\Pr[\mathcal{H}_i = 1] - \Pr[\mathcal{H}_{i+1} = 1]| \leq \mathsf{negl}(n)$.

**Hybrid $\mathcal{H}_0$**: This hybrid corresponds to the original experiment $\text{2P-AISO-AS-WUnf}_{\mathcal{A},\Theta}(n)$ (Figure 6). Note that the protocol in Figure 26 only requires two interactions, therefore $\overline{k} = 2$.

**Hybrid $\mathcal{H}_1$**: In this hybrid, we change the way we answer $\mathsf{OPreSignInit}$ queries, by replacing the execution of the protocol $\Pi_{\mathsf{PreSign}}$ with an ideal execution of its functionality $\mathcal{F}_{\mathsf{PreSign}}$.

*Claim.* If $\Pi_{\mathsf{PreSign}}$ is a UC-secure two-party computation protocol, then $\mathcal{H}_0 \approx \mathcal{H}_1$.

*Proof.* We note that we answer $\mathsf{OPreSignInit}$ and $\mathsf{ONewStatement}$ queries as in hybrid $\mathcal{H}_0$, and the only change here is to the $\Pi_{\mathsf{PreSign}}$ protocol, which is used during answering of $\mathsf{OPreSignInit}$ queries. Let $\Pi_{\mathsf{PreSign}}$ be a UC-secure two-party computation protocol. This implies that $\Pi_{\mathsf{PreSign}}$ UC-realizes $\mathcal{F}_{\mathsf{PreSign}}$, hence, the claim follows from the UC composition theorem [37]. This is because our security experiment acts as an environment for $\Pi_{\mathsf{PreSign}}$, and respectively for $\mathcal{F}_{\mathsf{PreSign}}$, for computing the function described in Figure 26.

**Hybrid** $\mathcal{H}_2$: In this hybrid, we switch from an honestly generated CRS and proof $(crs, \pi)$ to simulated $(crs^*, \pi^*)$.

*Claim.* If $\mathsf{NIZK}$ is computationally zero-knowledge, then $\mathcal{H}_1 \approx \mathcal{H}_2$.

*Proof.* In order to prove this claim we construct a reduction $\mathcal{B}$ to the computational zero-knowledge property of $\mathsf{NIZK}$. We note that all queries are answered as in the previous hybrid. The only difference here is that the reduction $\mathcal{B}$ receives a CRS $crs^*$ from the zero-knowledge challenger, and sets $crs := crs^*$, instead of $\mathcal{B}$ generating the $crs$ itself. Since we are in an ideal execution $\mathcal{F}_{\mathsf{PreSign}}$, we have that the inputs of both $A$ and $E$ for the 2PC must be provided to the functionality $\mathcal{F}_{\mathsf{PreSign}}$ (either by the reduction/environment $\mathcal{B}$ or the adversary $\mathcal{A}$ for the corrupted party). Let $(\mathsf{sk}_0, \mathsf{vk}, \mathsf{aux}, m, w)$ be the input of $A$ and $(\mathsf{sk}_1, \mathsf{vk}, \mathsf{aux}, m, x)$ input of $E$. Using these inputs $\mathcal{B}$ can evaluate the function given in Figure 26, and then forward the statement/witness pair $((ct_\sigma, ct_w, \mathsf{vk}, m, x), (\hat{\sigma}, h, w, r_1, r_2))$ to the oracle of the underlying zero-knowledge challenger, which replies with a proof $\pi^*$. $\mathcal{B}$ sets $\pi := \pi^*$ as part of the output $(ct_\sigma, ct_w, \pi)$ of 2PC.

If the zero-knowledge challenger of $\mathsf{NIZK}$ used the honest setup and prover algorithms to generate $crs^*$ and $\pi^*$, then we are exactly in hybrid $\mathcal{H}_1$, and if it used the simulator, then we are in hybrid $\mathcal{H}_2$. Therefore, if $\mathcal{A}$ can distinguish between the two hybrids with non-negligible advantage, then $\mathcal{B}$ can break the zero-knowledge property of $\mathsf{NIZK}$.

**Hybrid** $\mathcal{H}_3$: In this hybrid, we tag each query to the oracle $\mathsf{OPreSignInit}$. We denote each tag with an index denoted $i$. Then we guess which of these indices are for queries whose output will not be used later to call the oracle $\mathsf{OAdapt}$. Our guessing fails if the output of a presign query is later used in a query to the $\mathsf{OAdapt}$ oracle. If our guessing fails, we abort.

*Claim.* Let $Q_{\mathsf{PreSign}}$ denote the number of queries to oracle $\mathsf{OPreSignInit}$ that the adversary is allowed to make. Let $Q_{\mathsf{Adapt}}$ denote the number of queries to oracle $\mathsf{OAdapt}$ that the adversary is allowed to make. If $Q_{\mathsf{Adapt}} \leq \log(Q_{\mathsf{PreSign}})$ then $\mathcal{H}_2 \approx \mathcal{H}_3$.

*Proof.* In this game, instead of guessing directly the messages $m$ queried to the oracle $\mathsf{OPreSignInit}$, which would incur an exponential security loss, we guess the index of these queries. This guessing strategy results in a $\binom{Q_{\mathsf{PreSign}}}{Q_{\mathsf{Adapt}}}^{-1}$ security loss.

**Remark**: From now on, we will denote by $\mathcal{I}$ the set of indices that are part of our guess. If $i \notin \mathcal{I}$ then we would have guessed that the $i$th query to the presign oracle will not yield a result that will be used in a query to the $\mathsf{OAdapt}$ oracle.

**Hybrid** $\mathcal{H}_4$: Let $h = \mathsf{H}(\sigma)$ be the secret key of the $\mathsf{SKE}$ scheme. In this hybrid, in the oracle $\mathsf{OPreSignInit}$ if $i \notin \mathcal{I}$, we program the random oracle $\mathsf{H}$ to output $0^n$ instead of $h$ on input $\sigma$.

*Claim.* Let $\mathsf{Col}_i$ be the event that the adversary queries $\sigma$ to $\mathsf{H}$ in hybrid $\mathcal{H}_i$. Then, it holds that

$$|\Pr[\mathcal{H}_3 = 1] - \Pr[\mathcal{H}_4 = 1]| \leq \Pr[\mathsf{Col}_4].$$

*Proof.* Without loss of generality, we can assume that the SKE scheme $\mathsf{SKE}$ uses a uniformly random bitstring as a secret key. It follows by the properties of the random oracle $\mathsf{H}$ that the distinguishing gap between hybrids $\mathcal{H}_3$ and $\mathcal{H}_4$ is bound by the probability of the adversary $\mathcal{A}$ querying $\sigma$ to the random oracle $\mathsf{H}$. We denoted this event by $\mathsf{Col}_4$, hence, the probability follows straightforwardly. Looking ahead, we will bound the probability of this event happening after hybrid $\mathcal{H}_7$ using the unforgeability of the two-party signature scheme.

**Hybrid** $\mathcal{H}_5$: In this hybrid, in the oracle $\mathsf{OPreSignInit}$ if $i \notin \mathcal{I}$, we switch from computing $ct \leftarrow \mathsf{SKE.Enc}(h, w)$ to computing $ct \leftarrow \mathsf{SKE.Enc}(h, 0^{|w|})$. The condition $c_3$ ensures that if a statement $x$ presigns the forgery message $m^*$, then this statement is not compromised. This implies that the adversary is not allowed to use the oracle $\mathsf{OAdapt}$ to then extract the witnesses $w$ of these uncompromised statements. Therefore the adversary is not allowed to decrypt the ciphetext encrypted with uncompromissed witnesses.

*Claim.* If $\mathsf{SKE}$ is IND-CPA secure, then $\mathcal{H}_4 \approx \mathcal{H}_5$.

*Proof.* We construct a reduction $\mathcal{B}$ to the IND-CPA security of $\mathsf{SKE}$. $\mathcal{B}$ proceeds as in the previous hybrid for answering all the queries, except that when computing $ct_w$, $\mathcal{B}$ invokes the IND-CPA challenger. More precisely, $\mathcal{B}$ sets $(m_0 := w, m_1 := 0^{|w|})$ and submits this pair to the IND-CPA challenger, which in turn replies with $ct^*$. Next, $\mathcal{B}$ sets $ct_w := ct^*$, and computes a simulated proof $\pi$, which forms the part of the output $(ct_\sigma, ct_w, \pi)$ of $\mathcal{F}_{\mathsf{PreSign}}$.

If the IND-CPA challenger of $\mathsf{SKE}$ gave an encryption of $m_0$, then we are exactly in hybrid $\mathcal{H}_4$, and if it gave an encryption of $m_1$, then we are in hybrid $\mathcal{H}_5$. Therefore, if $\mathcal{A}$ can distinguish between the two hybrids with non-negligible advantage, then $\mathcal{B}$ can break the IND-CPA security of $\mathsf{SKE}$. Moreover, from IND-CPA security of $\mathsf{SKE}$ it also follows that $|\Pr[\mathsf{Col}_4] - \Pr[\mathsf{Col}_5]| \leq \mathsf{negl}(n)$.

**Hybrid** $\mathcal{H}_6$: In this hybrid, we modify the implementation of the presign ideal functionality. This modified implementation still realizes the ideal functionality. The only difference is that the signature computed on the first line of the protocol is registered in a table $\mathcal{C}[x]$. The hybrid is identical to $\mathcal{H}_5$, except that for the forgery of the adversary $(m, \sigma)$, we check whether there exists a statement $x$ such that $\sigma \in \mathcal{C}[x]$. If this is the case, then we abort.

*Claim.* Let $\mathsf{Bad}_1$ be the event that $\mathcal{H}_6$ aborts, then it holds that $\Pr[\mathsf{Bad}_1] \leq \nu_1(n)$, where $\nu_1$ is a negligible function in $n$.

*Proof.* We prove this claim via a reduction to the hardness of the relation $\mathsf{R}$. More precisely, we construct an adversary $\mathcal{B}$ that breaks the $q$-hardness of $\mathsf{R}$ using the adversary $\mathcal{A}$ that causes hybrid $\mathcal{H}_6$ to abort with non-negligible probability. Here $q$ denotes the number of queries that $\mathcal{A}$ makes to the $\mathsf{ONewStatement}$ oracle. $\mathcal{B}$ receives the challenge values $(x_1, \ldots, x_q)$ from $q$-hardness challenger of $\mathsf{R}$ and proceeds as in the previous hybrid to answer the oracle queries, with the following exceptions: (i) $\mathcal{B}$ answers the $i$-th query to $\mathsf{ONewStatement}$ with $x_i$; (ii) Upon receiving $\mathsf{OPreSignInit}$ query, $\mathcal{B}$ checks if the queried $x \in \{x_1, \ldots, x_q\}$, and if this is not the case, then $\mathcal{B}$ returns $\perp$, otherwise, $\mathcal{B}$ uses $w := 0^{|w|}$ during the execution of $\Pi^{\mathcal{A}}_{\mathsf{PreSign}}$, which is indistinguishable due to the previous hybrid.

Upon $\mathcal{A}$ outputting a forgery $(m, \sigma)$, if the event $\mathsf{Bad}_1$ happened, then there exists a statement $x$ such that $\sigma \in \mathcal{C}[x]$. For this statement and this signature, $\mathcal{B}$ can compute steps 3 to 7 of the presign protocol. This would yield a presignature $\hat{\sigma}$ that, due to presignature correctness, $\mathcal{B}$ can use to extract $w$ by running $\mathsf{Extract}(\mathsf{vk}, \sigma, \hat{\sigma}, x)$. With this, $\mathcal{B}$ obtains $(x, w) \in \mathsf{R}$.

We note that the view of $\mathcal{A}$ in this hybrid and in hybrid $\mathcal{H}_5$ are indistinguishable since the challenges $x_i$, for $i \in [q]$, are instances of the hard relation $\mathsf{R}$ and are generated using the $\mathsf{GenR}$ algorithm. Hence, the probability that $\mathcal{B}$ breaks the $q$-hardness of $\mathsf{R}$ is equal to the probability that the even $\mathsf{Bad}_1$ happens, which happens only with a negligible probability $\nu_1(n)$. It also follows that $|\Pr[\mathsf{Col}_5] - \Pr[\mathsf{Col}_6]| \leq \mathsf{negl}(n)$.

**Hybrid** $\mathcal{H}_7$: In this hybrid, in the oracle $\mathsf{OPreSignInit}$ if $i \notin \mathcal{I}$, we switch from computing $ct^* \leftarrow \mathsf{PKE.Enc}(x, \sigma)$ to computing $ct^* \leftarrow \mathsf{PKE.Enc}(x, 0^{|\sigma|})$. The condition $c_3$ ensures that if a statement $x$ presigns the forgery message $m^*$, then this statement is not compromised. This implies that the adversary is not allowed to use the oracle $\mathsf{OAdapt}$ to then extract the witnesses $w$ of these uncompromised statements. Therefore if the adversary encrypts using an uncompromised statement, the adversary will not be allowed to decrypt such ciphertext.

*Claim.* If $\mathsf{PKE}$ is IND-CPA secure, then $\mathcal{H}_6 \approx \mathcal{H}_7$.

*Proof.* We construct a reduction $\mathcal{B}$ to the IND-CPA security of $\mathsf{PKE}$, which uses the full extractability adversary $\mathcal{A}$ to break IND-CPA security of $\mathsf{PKE}$. $\mathcal{B}$ proceeds as in the previous hybrid except for computing $ct_\sigma$, for which $\mathcal{B}$ obtains $\mathsf{vk}^*$ from the IND-CPA challenger of $\mathsf{PKE}$, and sends $(m_0 := \sigma, m_1 := 0^{|\sigma|})$ to the IND-CPA challenger, which replies with a challenge ciphertext $ct^*$. $\mathcal{B}$ sets $x := \mathsf{vk}^*, ct_\sigma := ct^*$, where $ct_\sigma$ forms part of the output $(ct_\sigma, ct_w, \pi)$ of 2PC.

If the IND-CPA challenger of $\mathsf{PKE}$ gave an encryption of $m_0$, then we are exactly in hybrid $\mathcal{H}_6$, and if it gave an encryption of $m_1$, then we are in hybrid $\mathcal{H}_7$. Therefore, if $\mathcal{A}$ can distinguish between the two hybrids with non-negligible advantage, then $\mathcal{B}$ can break the IND-CPA security of $\mathsf{PKE}$. Moreover, from IND-CPA security of $\mathsf{PKE}$ it also follows that $|\Pr[\mathsf{Col}_6] - \Pr[\mathsf{Col}_7]| \leq \mathsf{negl}(n)$.

At this point all that remains is to prove that $\Pr[\mathsf{Col}_7] = \mathsf{negl}(n)$, which we can do by the unforgeability of the two-party signature scheme. More precisely, if there exists an adversary $\mathcal{A}$ that causes the event $\mathsf{Col}_7$ to trigger, then we can break the unforgeability of $\Sigma_2$. We observe that in hybrid $\mathcal{H}_7$ we have removed all the information in the ciphertexts, and more specifically, all information about $\sigma$ when querying the presignature of $m^*$. Hence, the adversary $\mathcal{A}$ needs to query $\sigma$ to the random oracle $\mathsf{H}$ without being given any information about $\sigma$. By the unforgeability of the underlying two-party signature scheme, we have that this happens with negligible probability, which concludes that $\Pr[\mathsf{Col}_7] = \mathsf{negl}(n)$.

**Lemma 15 (Full Extractability Of Figure 26).** *If $\Pi_{\mathsf{2PC}}$ is UC-secure, $\mathsf{DS}_2$ is unforgeable, $\mathsf{PKE}$ and $\mathsf{SKE}$ are IND-CPA secure, $\mathsf{R}$ satisfies q-hardness and $\mathsf{NIZK}$ has computational zero-knowledge, then the construction in Figure 26 satisfies full extractability.*

*Proof.* **Hybrid $\mathcal{H}_0$**: This hybrid corresponds to the original experiment 2P-AISO-AS-$\mathsf{FExt}_{\mathcal{A},\Theta}(n)$ Figure 6.

**Hybrid $\mathcal{H}_1$**: In this hybrid we replace $\Pi_{\mathsf{PreSign}}$ by its ideal functionality.

*Claim.* If $\Pi_{\mathsf{PreSign}}$ is a UC-secure computation protocol, then $\mathcal{H}_1 \approx \mathcal{H}_0$.

*Proof.* We note that we answer $\mathsf{OSign}$ queries as in hybrid $\mathcal{H}_0$, and the only change here is to the $\mathsf{OPreSignInit}$ oracle.

We note that we answer $\mathsf{OSign}$ queries as in hybrid $\mathcal{H}_0$, and the only change here is to the $\Pi_{\mathsf{PreSign}}$ protocol, which is used during answering of $\mathsf{OPreSignInit}$ queries. Let $\Pi_{\mathsf{PreSign}}$ be a UC-secure two-party computation protocol. This implies that $\Pi_{\mathsf{PreSign}}$ UC-realizes $\mathcal{F}_{\mathsf{PreSign}}$, hence, the claim follows from the UC composition theorem [37]. This is because our security experiment acts as an environment for $\Pi_{\mathsf{PreSign}}$, and respectively for $\mathcal{F}_{\mathsf{PreSign}}$, for computing the function described in Figure 26.

**Hybrid $\mathcal{H}_2$**: In this hybrid, we change $\mathsf{PreSign}$ in the cases where the adversary queries $\mathsf{OPreSign}_E$ with an statement such that $x \in \mathcal{X}$. In these cases, we replace the proof $\pi$ generated in the last line of the protocol with a simulated proof.

*Claim.* If $\mathsf{NIZK}$ is zero-knowledge, then $\mathcal{H}_2 \approx \mathcal{H}_1$.

*Proof.* If there exists a $\mathsf{PPT}$ adversary $\mathcal{A}$that distinguishes between $\mathcal{H}_2$ and $\mathcal{H}_1$ with non-negligible probability, then we can create an adversary $\mathcal{B}$ that breaks the zero-knowledge of $\mathsf{NIZK}$. Adversary $\mathcal{B}$ the zero-knowledge game with a challenger. At the same time, $\mathcal{B}$ plays the role of the challenger of $\mathcal{H}_1$ with $\mathcal{A}$, behaving honestly for all queries from $\mathcal{A}$. The only exception is that, whenever the proof $\pi$ is to be computed, $\mathcal{B}$ queries the proof oracle of the zero-knowledge. The output of this query is set as the proof of the $\mathsf{PreSign}$ protocol. Note that if the proof is simulated we would be in $\mathcal{H}_1$, otherwise, we would be in $\mathcal{H}_2$. If $\mathcal{A}$ is able to distinguish between these two hybrids with non-negligible probability, then $\mathcal{B}$ learns whether the challenger of the zero-knowledge is simulating the proofs or not with the same probability.

**Hybrid $\mathcal{H}_3$**: Let $h = \mathsf{H}(\sigma)$ be the secret key of the $\mathsf{SKE}$ scheme. In this hybrid, we change $\mathsf{PreSign}$ in the cases where the adversary queries $\mathsf{OPreSign}_E$ with an statement such that $x \in \mathcal{X}$. In these cases we program the random oracle $\mathsf{H}$ to output $0^n$ instead of $h$ on input $\sigma$.

*Claim.* Let $\mathsf{Col}_i$ be the event that the adversary queries $\sigma$ to $\mathsf{H}$ in hybrid $\mathcal{H}_i$. Then, it holds that

$$|\Pr[\mathcal{H}_2 = 1] - \Pr[\mathcal{H}_3 = 1]| \leq \Pr[\mathsf{Col}_3].$$

*Proof.* Without loss of generality, we can assume that the SKE scheme $\mathsf{SKE}$ uses a uniformly random bitstring as a secret key. It follows by the properties of the random oracle $\mathsf{H}$ that the distinguishing gap between hybrids $\mathcal{H}_2$ and $\mathcal{H}_3$ is bound by the probability of the adversary $\mathcal{A}$ querying $\sigma$ to the random oracle $\mathsf{H}$. We denoted this event by $\mathsf{Col}_3$, hence, the probability follows straightforwardly. Looking ahead, we will bound the probability of this event happening after hybrid $\mathcal{H}_7$ using the unforgeability of the two-party signature scheme.

**Hybrid** $\mathcal{H}_4$: In this hybrid, we change $\mathsf{PreSign}$ in the cases where the adversary queries $\mathsf{OPreSign}_E$ with an statement such that $x \in \mathcal{X}$. In these cases, we replace the $\mathsf{SKE}$ encryption to encrypt $0^{|w|}$.

*Claim.* If $\mathsf{SKE}$ is IND-CPA then $\mathcal{H}_4 \approx \mathcal{H}_3$.

*Proof.* If there exists a $\mathsf{PPT}$ adversary $\mathcal{A}$that distinguishes between $\mathcal{H}_3$ and $\mathcal{H}_4$ with non-negligible probability, then we can create an adversary $\mathcal{B}$ that breaks the IND-CPA of $\mathsf{SKE}$. Adversary $\mathcal{B}$ plays the role of the challenger of $\mathcal{H}_3$ honestly with $\mathcal{A}$. Whenever the $\mathsf{PreSign}$ protocol computes the $\mathsf{SKE}$ encryption, $\mathcal{B}$ queries the challenger of the IND-CPA for an encryption of either $m_0 := w$ or $m_1 := 0^{|w|}$. The result of this query is set as $ct_w$ in the $\mathsf{PreSign}$ protocol. Depending on which message is encrypted by the oracle, $\mathcal{B}$ would be reproducing $\mathcal{H}_3$ or $\mathcal{H}_4$. If $\mathcal{A}$ can distinguish between $\mathcal{H}_3$ and $\mathcal{H}_4$ with non-negligible probability, then $\mathcal{B}$ learns whether the challenger is encrypting messages $m_0$ or $m_1$ with the same probability.

**Hybrid** $\mathcal{H}_5$: In this hybrid, we change $\mathsf{PreSign}$ in the cases where the adversary queries $\mathsf{OPreSign}_E$ with an statement such that $x \in \mathcal{X}$. In these cases, we replace the $\mathsf{PKE}$ encryption to encrypt $0^{|\sigma|}$.

*Claim.* If $\mathsf{PKE}$ is IND-CPA then $\mathcal{H}_5 \approx \mathcal{H}_4$.

*Proof.* If there exists a $\mathsf{PPT}$ adversary $\mathcal{A}$that distinguishes between $\mathcal{H}_4$ and $\mathcal{H}_5$ with non-negligible probability, then we can create an adversary $\mathcal{B}$ that breaks the IND-CPA of $\mathsf{PKE}$. Adversary $\mathcal{B}$ plays the role of the challenger of $\mathcal{H}_4$ honestly with $\mathcal{A}$. Whenever the $\mathsf{PreSign}$ protocol computes the $\mathsf{PKE}$ encryption, $\mathcal{B}$ queries the challenger of the IND-CPA for an encryption of either $m_0 := \sigma$ or $m_1 := 0^{|\sigma|}$. The result of this query is set as $ct_\sigma$ in the $\mathsf{PreSign}$ protocol. Depending on which message is encrypted by the oracle, $\mathcal{B}$ would be reproducing $\mathcal{H}_4$ or $\mathcal{H}_5$. If $\mathcal{A}$ can distinguish between $\mathcal{H}_4$ and $\mathcal{H}_5$ with non-negligible probability, then $\mathcal{B}$ learns whether the challenger is encrypting messages $m_0$ or $m_1$ with the same probability.

**Hybrid** $\mathcal{H}_6$: In this hybrid, abort if the adversary $\mathcal{A}$ can trigger the event $\mathsf{Col}_6$.

*Claim.* If $\mathsf{DS}_2$ is unforgeable, then $\Pr[\mathsf{Col}_6] = \mathsf{negl}(n)$, and thus, $\mathcal{H}_6 \approx \mathcal{H}_2$.

*Proof.* In hybrid $\mathcal{H}_6$ we have removed all the information in the ciphertexts, and more specifically, all information about $\sigma$ when querying $\mathsf{PreSign}_E$ with an statement such that $x \in \mathcal{X}$. Hence, in these cases, the adversary $\mathcal{A}$ needs to query $\sigma$ to the random oracle $\mathsf{H}$ without being given any information about $\sigma$. By the unforgeability of the underlying two-party signature scheme, we have that this happens with negligible probability, which concludes that $\Pr[\mathsf{Col}_6] = \mathsf{negl}(n)$.

**Hybrid** $\mathcal{H}_7$: In this hybrid, after the adversary presents the forgery $(m, \sigma)$ we abort if the set of $(x, \hat{\sigma}) \in \mathcal{T}_A[m]$ such that $x \notin \mathcal{X}$ is empty. Let $Bad_1$ be the event that we abort in this case.

*Claim.* If $\mathsf{DS}$ is unforgeable, then $\mathcal{H}_7 \approx \mathcal{H}_6$.

*Proof.* If the set of $(x, \hat{\sigma}) \in \mathcal{T}_A[m]$ such that $x \notin \mathcal{X}$ is empty, it means that oracle $\mathsf{OPreSign}_E$ was never queried with the forgery message $m$ and a statement $x$ not generated by oracle $\mathsf{ONewStatement}$. If this is the case, we can create an adversary $\mathcal{B}$ that breaks the unforgeability of the underlying signature scheme. Adversary $\mathcal{B}$ plays $\mathcal{H}_7$ with $\mathcal{A}$. If $\mathcal{A}$ makes a presign query with a statement such that $x \in \mathcal{X}$ we proceed as in previous hybrids, where it is not required to obtain a signature in order to compute $\hat{\sigma}$. Otherwise, in a query to $\mathsf{PreSign}_E$, $\mathcal{B}$ behaves honestly and in order to compute the $\mathsf{PreSign}$ protocol, replaces the first line with a query for a signature to the challenger of the unforgeability game. If $\mathcal{A}$ is able to trigger $Bad_1$, then we know that the forgery satisfies $\mathsf{Vrfy}(\mathsf{vk}, m, \sigma) = 1$ and. Moreover, $\mathcal{B}$ never queried the sign oracle for the forgery message when answering a query to $\mathsf{OPreSign}_E$ where $x \in \mathcal{X}$. And finally, the event $Bad_1$ implies that the set of $(x, \hat{\sigma}) \in \mathcal{T}_A[m]$ such that $x \notin \mathcal{X}$ is empty. Thus $\mathcal{B}$ never queried for a signature on the forgery message when answering to a query to $\mathsf{OPreSign}_E$ where $x \notin \mathcal{X}$. Therefore, $\mathcal{B}$ wins the unforgeability game with the same probability that $\mathcal{A}$ triggers $Bad_1$.

*Claim.* If there exists a $\mathsf{PPT}$ adversary $\mathcal{A}$ that wins $\mathcal{H}_7$ with non-negligible probability, then we can create a $\mathsf{PPT}$ adversary $\mathcal{B}$ that breaks knowledge soundness of $\mathsf{NIZK}$ with the same probability.

*Proof.* By the previous hybrids, we know that if an adversary wins $\mathcal{H}_7$ it must be that the set of $(x, \hat{\sigma}) \in \mathcal{T}_A[m]$ such that $x \notin \mathcal{X}$ is non-empty. Thus, if $c_3$ is satisfied, if must be the case that extraction fails for the forgery $\sigma$ and any of the presignatures yielded by oracle $\mathsf{OPreSign}_E$. By line 2 of the oracle $\mathsf{OPreSign}_E$, we know that all of those presignatures satisfy $\mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma}) = 1$. This implies that the NIZK verification $\mathsf{Verify}(crs, (ct_\sigma, ct_w, \mathsf{vk}, m, x), \pi)$ passes, where $(ct_\sigma, ct_w, \pi) := \hat{\sigma}$. However, by the correctness of the scheme, if extraction fails, must be the case that $((ct_\sigma, ct_w, \mathsf{vk}, m, x), (\sigma, h, w, r_1, r_2)) \notin \mathsf{R}^*$. Therefore, we found a proof that verifies for an instance not in the relation.

**Lemma 16 (Adaptability Of Figure 26).** *If* NIZK *is knowledge-sound and* PKE *is correct, then the construction of Figure 26 satisfies adaptability.*

*Proof.* Assume that, for a given message $m$, an instance $(x, w) \in \mathsf{R}$ and a presignature $\hat{\sigma}$, it holds that $\mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma}) = 1$ but $\mathsf{Vrfy}(\mathsf{vk}, m, \mathsf{Adapt}(\mathsf{sk}_A, \mathsf{vk}, \hat{\sigma}, w, m)) \neq 1$. Then we can find a statement $x^* := (ct_\sigma, ct_w, \mathsf{vk}, m, x)$ and proof $\pi$ that break knowledge-soundness of the NIZK. In other words, $\mathsf{NIZK.Verify}(crs, x^*, \pi_A) = 1$ but running the extractor algorithm yields a witness $w^* \leftarrow \mathsf{Ext}(crs, x^*, \pi_A)$ such that $(x^*, w^*) \notin \mathsf{R}^*$.

However, if $\mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma}) = 1$ holds, where the presignature $\hat{\sigma}$ is the tuple $(ct_\sigma, ct_w, \pi)$, then by the definition of the PreVrfy algorithm then we know that $\mathsf{Verify}(crs, (ct_\sigma, ct_w, \mathsf{vk}, m, x), \pi) = 1$. By the definition of the Adapt algorithm and the correctness of PKE, we know that $\sigma \leftarrow \mathsf{Adapt}(\mathsf{sk}_A, \mathsf{vk}, \hat{\sigma}, w, m)$ is the signature encrypted in $ct_\sigma$. By assumption, we know that $\mathsf{Vrfy}(\mathsf{vk}, m, \mathsf{Adapt}(\mathsf{sk}_A, \mathsf{vk}, \hat{\sigma}, w, m)) \neq 1$. Therefore, we have found a statement $x^* := (ct_\sigma, ct_w, \mathsf{vk}, m, x) \notin L^*_{\mathsf{R}}$ and proof $\pi$ that break knowledge-soundness of the NIZK.

**Lemma 17 (Unlinkability Of Figure 26).** *By presignature correctness of Figure 26, the construction of Figure 26 satisfies unlinkability.*

*Proof.* By the presignature correctness of the construction of Figure 26, we observe that adapting a presignature yields a signature computed as in a two-party signing protocol. Therefore, in the unlinkability experiment, the distribution of the signatures produced by calls to the oracle OChall is the same regardless of whether $b = 0$ or $b = 1$. Thus, an adversary distinguishing the unlinkability experiment between the two cases would be able to distinguish from exactly the same distribution with non-negligible probability.

**Lemma 18 (Computational Preverify Soundness Of Figure 26).** *If* NIZK *is knowledge-sound, then the construction in Figure 26 satisfies computational preverify soundness.*

*Proof.* Assume that there exists a polynomially bounded $x \notin L_{\mathsf{R}}$ and a PPT adversary $\mathcal{A}$ that on input honestly generated keys outputs a pair $(m, \hat{\sigma})$ such that $\mathsf{PreVrfy}(\mathsf{vk}, \hat{\sigma}, m, x) = 1$ with non-negligible probability. Then we can find an adversary $\mathcal{B}$ that using $\mathcal{A}$ breaks the knowledge-soundness of the NIZK. The adversary $\mathcal{B}$ is given a $crs$. Then they honestly generate keys $(\mathsf{vk}, (\mathsf{vk}_A, \mathsf{sk}_A)(\mathsf{vk}_E, \mathsf{sk}_E))$ and forwards everything to $\mathcal{A}$. Next, $\mathcal{A}$ returns the pair $(m, \hat{\sigma})$ that preverifies. The presignature is the tuple $(ct_\sigma, ct_w, \pi)$. If the presignature preverifies, then we have that $\mathsf{Verify}(crs, (ct_\sigma, ct_w, m, x), \pi) = 1$, even though $x \notin L_{\mathsf{R}}$. Thus, the adversary $\mathcal{B}$ returns the statement $(ct_\sigma, ct_w, m, x)$ and the proof $\pi$, which would verify even though $x \notin L_{\mathsf{R}}$. In the language $L^*$ of the NIZK the last element $x$ of the statement $x^* := (ct_\sigma, ct_w, m, x)$ must satisfy $x \in L_{\mathsf{R}}$. Therefore, $x^* \notin L^*$.

# I Definition of 2P-AISO-AS for Multiple Rounds

In this section we generalize the definition of 2P-AISO-AS in Section 4 for presign protocols that require more than two steps. We model the two-party presignature generation as an interactive protocol that takes $\overline{k}$ steps and delivers the final presignature to the *Extractor*. Both *Adaptor* and *Extractor* can abort this interactive protocol at any point. Starting with step number 1, each party takes turns in executing the steps of the presign protocol and communicating the results of each step until the last step (step number $\overline{k}$) is executed by the *Extractor*. Running step number $\overline{k}$ finalizes the protocol.

**Definition 38 (Two-Party Asymmetric-Input Solitary-Output Adaptor Signature Scheme).** *A $\overline{k}$-step Two-Party Asymmetric-Input Solitary-Output Adaptor Signature (2P-AISO-AS) scheme is defined w.r.t. a relation $R \subseteq (\mathcal{X}, \mathcal{W})$ and a two-party signature scheme $\mathsf{DS}_2 = (\mathsf{Setup}, \mathsf{KeyGen}, \{\mathsf{Sign}_i\}_{i\in[k]}, \mathsf{Vrfy})$. It is run between parties Adaptor (A) and Extractor (E), and consists of a tuple $\mathsf{AS}_2^{R,\mathsf{DS}_2} = (\{\mathsf{PreSign}_i\}_{i\in[\overline{k}]}, \mathsf{Adapt}, \mathsf{PreVrfy}, \mathsf{Extract})$ of efficient algorithms defined as:*

- *A succession of* PPT *algorithms $\{\mathsf{PreSign}_i\}_{i\in[\overline{k}]}$. Each $\mathsf{PreSign}_i$ algorithm is a single step in the $\overline{k}$-interaction presign protocol. They are defined, with respect to message $m \in \{0,1\}^n$, the keys $(\mathsf{vk}, (\mathsf{vk}_A, \mathsf{sk}_A), (\mathsf{vk}_E, \mathsf{sk}_E))$ and an instance of the hard relation $(x, w) \in \mathsf{R}$. The $\mathsf{PreSign}_i$ algorithms are defined as follows:*
  - *The output of $\mathsf{PreSign}_j$ for $j < \overline{k}$ is $(pm_j, st_j)$ where $pm_j$ is a protocol message and $st_j$ is a protocol state.*
  - *The output of $\mathsf{PreSign}_{\overline{k}}$ is $\hat{\sigma}$, where $\hat{\sigma}$ is a presignature.*
  - *If $\overline{k}$ is even, the input of $\mathsf{PreSign}_1$ is $(\mathsf{sk}_A, w, m, x)$. Otherwise, if $\overline{k}$ is odd, the input of $\mathsf{PreSign}_1$ is $(\mathsf{sk}_E, m, x)$.*
  - *If $\overline{k}$ is even, the input of $\mathsf{PreSign}_2$ is $(\mathsf{sk}_E, m, x, pm_1)$. Otherwise, if $\overline{k}$ is odd, the input of $\mathsf{PreSign}_2$ is $(\mathsf{sk}_A, w, m, x, pm_1)$.*
  - *The input of $\mathsf{PreSign}_j$ for $3 \leq j \leq \overline{k}$ is $(st_{j-2}, pm_{j-1})$.*
- *$b := \mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma})$: is a* DPT *algorithm that on input a verification key $\mathsf{vk}$, a message $m \in \mathcal{M}$, a statement $x \in L_{\mathsf{R}}$ and a presignature $\hat{\sigma}$, outputs a bit $b$.*
- *$\sigma := \mathsf{Adapt}(\mathsf{vk}, \hat{\sigma}, w, st_{\overline{k}-1})$: is a* DPT *algorithm that on input a verification key $\mathsf{vk}$, a presignature $\hat{\sigma}$, a witness $w$ and the last protocol state $st_{\overline{k}-1}$, outputs a signature $\sigma$.*
- *$w := \mathsf{Extract}(\mathsf{vk}, \sigma, \hat{\sigma}, x)$: is a* DPT *algorithm that on input a verification key $\mathsf{vk}$, a signature $\sigma$, a presignature $\hat{\sigma}$ and a statement $x \in L_{\mathsf{R}}$, outputs a witness $w$ s.t. $(x, w) \in R$, or an error symbol $\perp$.*

For ease of readability, we next define the notion of presignature protocol for two-party adaptor signatures. This encapsulates the interactions between the adaptor $A$ and the extractor $E$ to generate a presignature $\hat{\sigma}$. Either the *Adaptor* or the *Extractor* start running the first step $\mathsf{PreSign}_1$ and then they take alternating turns running the subsequent steps. By the end of the protocol, the *Extractor* runs the last step of the protocol $\mathsf{PreSign}_{\overline{k}}$ and obtains the presignature $\hat{\sigma}$. Since the *Adaptor* and the *Extractor* take alternating turns for running each step of the protocol, and since the *Extractor* must run the last step $\mathsf{PreSign}_{\overline{k}}$, we know that the *Extractor* must run the steps with the same parity as $\overline{k}$. In other words, if $\overline{k}$ is even, then the *Extractor* must run the even steps of the protocol. If $\overline{k}$ is odd instead, then the *Extractor* runs the odd steps of the protocol. Thus, the *Extractor* must run the first step of the protocol $\mathsf{PreSign}_1$ if $\overline{k}$ is odd. Otherwise, if $\overline{k}$ is even, the *Adaptor* must run the first step of the protocol. Since the *Adaptor* must use the witness $w$ in the presign protocol, the first step that *Adaptor* runs must have $w$ as an input.

**Definition 39 (Presignature protocol).** *The presignature protocol $\Pi\mathsf{PreSign}\langle A(\mathsf{sk}_A, w), E(\mathsf{sk}_E)\rangle(m, x)$ between party Adaptor (A) on input $(\mathsf{sk}_A, w, m, x)$ and party Extractor (E) on input $(\mathsf{sk}_E, m, x)$ is defined as:*

- *If the number of interactions $\overline{k}$ is odd, E starts by running $\mathsf{PreSign}_1$. Otherwise, A starts by running $\mathsf{PreSign}_1$. After the first step, the parties alternate turns running $\mathsf{PreSign}_i$ in order for $i = 2$ until $i = \overline{k}$.*
- *On each step $i < \overline{k}$ the party $P \in \{A, E\}$ runs $\mathsf{PreSign}_i$ to generate $(pm_i, st_i)$. Then $P$ stores $st_i$ and sends $pm_i$ to the other party.*
- *On the last step $i = \overline{k}$, the Extractor runs $\mathsf{PreSign}_{\overline{k}}$ to generate a presignature $\hat{\sigma}$.*

**On the different values of $\overline{k}$.** . The generic definition of 2P-AISO-AS captures many 2P-AISO-AS schemes for different values of $\overline{k}$, where $\overline{k}$ is the number of presign steps $\{\mathsf{PreSign}_i\}_{i\in[\overline{k}]}$. The particular case where $\overline{k} = 2$ must be congruent with the definition we provided in Section 4. In the case $\overline{k} = 2$, the presign protocol is a two step protocol, $\mathsf{PreSign}_1$ and $\mathsf{PreSign}_2$. A simple renaming of the algorithms, $\mathsf{PreSign}_1 = \mathsf{PreSign}_A$ and $\mathsf{PreSign}_2 = \mathsf{PreSign}_E$, shows that the two notions match. In the following we will show that also the

correctness and security definitions of the generic 2P-AISO-AS are equivalent to the security definitions of Section 4 in the case $\overline{k} = 2$.

**Correctnes.** We show the correctness of 2P-AISO-AS.

**Definition 40 (Two-Party Presignature Correctness).** *A 2P-AISO-AS scheme* $\mathsf{AS}^{\mathsf{R},\mathsf{DS}_2}$ *satisfies two-party presignature correctness if for every* $n \in \mathbb{N}$*, and every message* $m \in \mathcal{M}$*, the following holds:*

$$\Pr\left[\begin{array}{l|r} b_0 \wedge & (\mathsf{vk}, (\mathsf{vk}_A, \mathsf{sk}_A), (\mathsf{vk}_E, \mathsf{sk}_E)) \leftarrow \mathsf{DS}_2.\mathsf{KeyGen}(1^n) \\ & (x, w) \leftarrow \mathsf{GenR}(1^n) \\ b_1 \wedge & \hat{\sigma} \leftarrow \Pi_{\mathsf{PreSign}\langle(\mathsf{sk}_A, w),(\mathsf{sk}_E)\rangle}(m, x) \\ b_2 & b_0 := \mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma});\ \sigma := \mathsf{Adapt}(\mathsf{vk}, \hat{\sigma}, w, st_{\overline{k}-1}) \\ & b_1 := \mathsf{Vrfy}(\mathsf{vk}, m, \sigma);\ w' := \mathsf{Extract}(\mathsf{vk}, \sigma, \hat{\sigma}, x) \\ & b_2 := (x, w') \in R \end{array}\right] = 1.$$

### I.1 Security and Privacy Notions

In this section, we define the security and privacy notions of interest for 2P-AISO-AS.

**Definition 41 (Full extractability).** *A 2P-AISO-AS scheme is* fully extractable *if for every* PPT *adversary* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}$ *such that* $\Pr[\textit{2P-AISO-AS-}\mathsf{FExt}^{\overline{k}}_{\mathcal{A},\Theta}(n) = 1] \leq \mathsf{negl}(n)$*, where the experiment 2P-AISO-AS-*$\mathsf{FExt}^{\overline{k}}_{\mathcal{A},\Theta}$ *is defined in Figure 27.*

In the full extractability game (Figure 27) the challenger assumes the role of the *Extractor*, while the adversary plays the role of the *Adaptor*.

Due to the presign protocol having multiple steps, the game provides oracles $\mathsf{OPreSignInit}$ and $\mathsf{OPreSignInteraction}$. These oracles allow the adversary to query the challenger for an advancement in an instance of the presign protocol. More specifically, $\mathsf{OPreSignInit}$ is the first step of the presign protocol that the challenger (*Extractor*) runs, whereas $\mathsf{OPreSignInteraction}$ are the subsequent steps that the challenger runs.

The oracle $\mathsf{OPreSignInit}$ initiates an instance of the presign protocol and tags it with a unique session id *sid*. There could be multiple concurrent instances of the presign protocol. To keep track of the different instances, we store each unique session id *sid* in a table *ID*.

A query to $\mathsf{OPreSignInteraction}$ must specify both the presign protocol instance to advance (by providing a session id) and which step of the protocol to run (step number $i$). In other words, the adversary queries the oracle $\mathsf{OPreSignInteraction}$ to run step number $i$ of the presign protocol with id *sid*. We ensure that the adversary queries for steps in the correct order (without duplicating a step) with the assertions of lines 3 and 4 of $\mathsf{OPreSignInteraction}$, which we will explain in detail in the following.

The challenger uses table *St* to keep track of the states of the different presign sessions. The table maps a pair of session id *sid* and step $i$ with a protocol state $st_i$. This $st_i$ is the protocol state that the challenger stores after running $i$'th step $\mathsf{PreSign}_i$. Line 3 of oracle $\mathsf{OPreSignInteraction}$ ensures that an adversary cannot query the oracle to run a presigning step $\mathsf{PreSign}_i$ more than once for the same session. Line 4 of oracle $\mathsf{OPreSignInteraction}$ ensures that an adversary cannot query step number $i$ without first querying the previous steps of the same session, thus enforcing a correct execution order.

By the definition of the presign protocol, if the number of interactions $\overline{k}$ is odd, then the challenger (*Extractor*) must only run the odd steps ($\mathsf{PreSign}_1$, $\mathsf{PreSign}_3$, etc.) of the protocol. Otherwise, the challenger (*Extractor*) must only run the even steps of the protocol. To ensure this, the game checks the parity of $\overline{k}$ several times (assert on line 1 of $\mathsf{OPreSignInteraction}$, and branching on lines 9 and 14 of $\mathsf{OPreSignInit}$). There is a special case if the number of steps of the presign protocol $\overline{k}$ is 2. In this case, the challenger (*Extractor*) only runs one step of the protocol. Therefore, in this case, the oracle $\mathsf{OPreSignInteraction}$ is not used (there would be no steps beyond $\mathsf{PreSign}_1$ or $\mathsf{PreSign}_2$). This implies that oracle $\mathsf{OPreSignInit}$ should start and

$\mathsf{FExt}^{\overline{k}}_{\mathcal{A},\Theta}(n)$

1 : $\mathcal{Q} := \mathcal{X} := \emptyset$
2 : $St := ID := \mathcal{T}_E := \bot$
3 : $(\mathsf{vk}, (\mathsf{vk}_E, \mathsf{sk}_E), (\mathsf{vk}_A, \mathsf{sk}_A))$
4 : $\quad \leftarrow\!\!\!\!\!\!\$\ \mathsf{KeyGen}(1^n)$
5 : $\mathcal{O} := \{\mathsf{OSign}, \mathsf{ONewStatement}$
6 : $\quad \mathsf{OPreSignInit},$
7 : $\quad \mathsf{OPreSignInteraction}\}$
8 : $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{vk}, \mathsf{vk}_E, (\mathsf{vk}_A, \mathsf{sk}_A))$
9 : $c_1 := m^* \notin \mathcal{Q}$
10 : $c_2 := \mathsf{Vrfy}(\mathsf{vk}, m^*, \sigma^*) = 1$
11 : $c_3 := \forall sid \in ID[m^*]:$
12 : $\quad \textbf{let}(x, \hat{\sigma}) := \mathcal{T}_E[sid]$ s.t. $x \notin \mathcal{X}$.
13 : $\quad (x, \mathsf{Extract}(\mathsf{vk}, \sigma^*, \hat{\sigma}, x)) \notin \mathsf{R}$
14 : **return** $(c_1 \wedge c_2 \wedge c_3)$

$\mathsf{ONewStatement}()$

1 : $(x, w) \leftarrow \mathsf{GenR}(1^n)$
2 : $\mathcal{X} := \mathcal{X} \cup \{x\}$
3 : **return** $x$

$\mathsf{OPreSignInit}(x, m, pm)$

1 : $sid \leftarrow\!\!\!\!\!\!\$\ \{0,1\}^*$
2 : $ID[m] := ID[m] \cup \{sid\}$
3 : **if** $\overline{k} = 2$ // Last step
4 : $\quad \hat{\sigma} \leftarrow \mathsf{PreSign}_2(\mathsf{sk}_E, x, m, pm)$
5 : $\quad$ **assert** $\mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma})$
6 : $\quad \mathcal{T}_E[sid] := (x, \hat{\sigma})$
7 : $\quad$ **return** $\hat{\sigma}$
8 : $\mathcal{T}_E[sid] := (x, \bot)$
9 : **if** $\overline{k} \equiv 1 \bmod 2$ // *Extractor* starts
10 : $\quad (pm_1, st_1)$
11 : $\quad\ \leftarrow\!\!\!\!\!\!\$\ \mathsf{PreSign}_1(\mathsf{sk}_E, x, m)$
12 : $\quad St[sid, 1] := st_1$
13 : $\quad$ **return** $(sid, pm_1)$
14 : **else** // *Adaptor* (adversary) starts
15 : $\quad (pm_2, st_2)$
16 : $\quad\ \leftarrow\!\!\!\!\!\!\$\ \mathsf{PreSign}_2(\mathsf{sk}_E, x, m, pm)$
17 : $\quad St[sid, 2] := st_2$
18 : $\quad$ **return** $(sid, pm_2)$

$\mathsf{OPreSignInteraction}(sid, pm, i)$

1 : **assert** $i \equiv \overline{k} \bmod 2$
2 : **assert** $\exists m : sid \in ID[m]$
3 : **assert** $St[sid, i] = \bot$
4 : **assert** $St[sid, i-2] \neq \bot$
5 : $st_{i-2} := St[sid, i-2]$
6 : **if** $i \neq \overline{k}$ // Not last step
7 : $\quad (pm_i, st_i)$
8 : $\quad\ \leftarrow\!\!\!\!\!\!\$\ \mathsf{PreSign}_i(st_{i-2}, pm)$
9 : $\quad St[sid, i] := st_i$
10 : $\quad$ **return** $pm_i$
11 : **else** // Last step
12 : $\quad \hat{\sigma} \leftarrow \mathsf{PreSign}_{\overline{k}}(st_{i-2}, pm)$
13 : $\quad (x, _) := \mathcal{T}_E[sid]$
14 : $\quad$ **assert** $\mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma})$
15 : $\quad \mathcal{T}_E[sid] := (x, \hat{\sigma})$
16 : $\quad$ **return** $\hat{\sigma}$

Fig. 27: Full extractability game where the adversary corrupts the adaptor party $A$. The oracle $\mathsf{OSign}$ is defined as in two-party digital signatures, and $\mathcal{Q}$ is the set of messages queried to the signing oracles (see Section A.8).

finalize a presign protocol, returning the presignature $\hat{\sigma}$. This special case is handled by the branching on line 3 of the oracle $\mathsf{OPreSignInit}$. In fact, for $\overline{k} = 2$, the game in Figure 27 reduces to the full extractability of Section 4.

At the end of the game, the adversary presents a forgery message and signature $(m^*, \sigma^*)$. This forgery is valid if it satisfies conditions $c_1$, $c_2$ and $c_3$. Condition $c_1$ and $c_2$ checks that the forgery verifies and that the message was not queried to the signing oracles. Condition $c_3$ checks that the *Extractor* cannot extract a valid witness from the forgery signature. To be more precise, $c_3$ first collects for all the presignatures generated on the forgery message $m^*$ and any statement $x$. Then, $c_3$ checks that running $\mathsf{Extract}$ on the forgery signature and each of these presignatures yields an invalid witness. We say that the witness is invalid because it does not correspond to the statement $x$ used to generate the presignature.

**Definition 42 (Weak unforgeability).** *A 2P-AISO-AS scheme $\Theta := \mathsf{AS}^{\mathsf{R},\mathsf{DS}_2}$ is* weakly unforgeable *if for every* PPT *adversary $\mathcal{A}$ there exists a negligible function* negl *such that* $\Pr[\mathsf{WExt}^{\overline{k}}_{\mathcal{A},\Theta}(n) = 1] \leq \mathsf{negl}(n)$, *where the experiment $\mathsf{WExt}^{\overline{k}}_{\mathcal{A},\Theta}$ is defined in Figure 28.*

In weak unforgeability, the challenger assumes the role of the *Adaptor*, while the adversary plays the role of the *Extractor*. The adversary can advance steps of the presign protocol in the same way as in the full extractability game: by querying oracles $\mathsf{OPreSignInit}$ and $\mathsf{OPreSignInteraction}$. By the definition of the presign protocol (Definition 39), if the number of steps $\overline{k}$ is odd, then the challenger (*Adaptor*) must run the even steps. Otherwise, if $\overline{k}$ is even, then the challenger must run the odd steps. We ensure this with the branching on lines 8 and 13 of oracle $\mathsf{OPreSignInit}$ and line 1 of oracle $\mathsf{OPreSignInteraction}$.

$\mathsf{WUnf}^{\overline{k}}_{\mathcal{A},\Theta}(n)$

1 : $\mathcal{Q} := \mathcal{W} := \mathcal{X} := \emptyset$
2 : $St := ID := \mathcal{T}_A := \bot$
3 : $(\mathsf{vk}, (\mathsf{vk}_E, \mathsf{sk}_E), (\mathsf{vk}_A, \mathsf{sk}_A))$
4 : $\quad \leftarrow \mathsf{KeyGen}(1^n)$
5 : $\mathcal{O} := \{\mathsf{OSign}, \mathsf{ONewStatement}$
6 : $\quad \mathsf{OAdapt}, \mathsf{OPreSignInit},$
7 : $\quad \mathsf{OPreSignInteraction}\}$
8 : $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{vk}, \mathsf{vk}_A, (\mathsf{vk}_E, \mathsf{sk}_E))$
9 : $c_1 := m^* \notin \mathcal{Q} \wedge \mathcal{X} \cap \mathcal{T}_A[m^*] = \emptyset$
10 : $c_2 := \mathsf{Vrfy}(\mathsf{vk}, m^*, \sigma^*) = 1$
11 : **return** $(c_1 \wedge c_2)$

$\mathsf{OAdapt}(\hat{\sigma}, x, m, sid)$

1 : **assert** $\exists w : (x, w) \in \mathcal{W}$
2 : **assert** $\mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma})$
3 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
4 : $\mathcal{X} := \mathcal{X} \cup \{x\}$
5 : $\mathcal{Q} := \mathcal{Q} \cup \{m : x \in \mathcal{T}_A[m]\}$
6 : $st_{\overline{k}-1} := St[sid, \overline{k} - 1]$
7 : **return** $\mathsf{Adapt}(\hat{\sigma}, w, st_{\overline{k}-1})$

$\mathsf{OPreSignInit}(x, m, pm)$

1 : **assert** $\exists w : (x, w) \in \mathcal{W}$
2 : $\mathcal{T}_A[m] := \mathcal{T}_A[m] \cup \{x\}$
3 : **if** $x \in \mathcal{X} \vee m \in \mathcal{Q}$ **then**
4 : $\quad \mathcal{Q} := \mathcal{Q} \cup \{m\}$
5 : $\quad \mathcal{X} := \mathcal{X} \cup \mathcal{T}_A[m]$
6 : $sid \leftarrow_{\$} \{0,1\}^*$
7 : $ID := ID \cup \{sid\}$
8 : **if** $\overline{k} \equiv 0 \bmod 2$ // *Adaptor* starts
9 : $\quad (pm_1, st_1)$
10 : $\quad \leftarrow \mathsf{PreSign}_1(\mathsf{sk}_A, w, m, x)$
11 : $\quad St[sid, 1] := st_1$
12 : $\quad$ **return** $(sid, pm_1)$
13 : **else** // *Extractor* (adversary) starts
14 : $\quad (pm_2, st_2)$
15 : $\quad \leftarrow \mathsf{PreSign}_2(\mathsf{sk}_A, w, m, x, pm)$
16 : $\quad St[sid, 2] := st_2$
17 : $\quad$ **return** $(sid, pm_2)$

$\mathsf{OPreSignInteraction}(sid, pm, i)$

1 : **assert** $i \not\equiv \overline{k} \bmod 2$
2 : **assert** $sid \in ID$
3 : **assert** $St[sid, i] = \bot$
4 : **assert** $St[sid, i-2] \neq \bot$
5 : $st_{i-2} := St[sid, i-2]$
6 : $(pm_i, st_i)$
7 : $\quad \leftarrow \mathsf{PreSign}_i(st_{i-2}, pm)$
8 : $St[sid, i] := st_i$
9 : **return** $pm_i$

$\mathsf{ONewStatement}()$

1 : $(x, w) \leftarrow \mathsf{GenR}(1^n)$
2 : $\mathcal{W} := \mathcal{W} \cup \{(x, w)\}$
3 : **return** $x$

$\mathsf{OSign}(m)$

1 : $\mathcal{X} := \mathcal{X} \cup \mathcal{T}_A[m]$
2 : $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
3 : **return** $\Pi\mathsf{Sign}(m)$

Fig. 28: Weak unforgeability game where the adversary corrupts the *Extractor* party $E$. We denote by $\Pi\mathsf{Sign}$ the two-party signature protocol.

As in full extractability, the adversary can run multiple instances of the presign protocol. Each instance of the presign protocol is tagged with a session id $sid$. The table $St$ keeps track of the state of the different presign sessions the same way as in full extractability. We also ensure that no step is duplicated during the same session (oracle $\mathsf{OPreSignInteraction}$ line 3) and that no step can be queried before the previous step (oracle $\mathsf{OPreSignInteraction}$ line 4) as in full extractability.

**Definition 43 (Adaptability and designated adaptability).** *For any message $m \in \mathcal{M}$, any verification key $\mathsf{vk}$, any statement/witness pair $(x, w) \in \mathsf{R}$, and any presignature $\hat{\sigma}$ with $\mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma}) = 1$, we say that a 2P-AISO-AS scheme satisfies* designated adaptability *if there exists a protocol state $st$ for which it holds that $\mathsf{Vrfy}(\mathsf{vk}, m, \mathsf{Adapt}(\mathsf{vk}, \hat{\sigma}, w, st)) = 1$. Alternatively, we say that a 2P-AISO-AS scheme satisfies* adaptability *if for every protocol state $st$ it holds that $\mathsf{Vrfy}(\mathsf{vk}, m, \mathsf{Adapt}(\mathsf{vk}, \hat{\sigma}, w, st)) = 1$.*

**Definition 44 (Unlinkability).** *A 2P-AISO-AS scheme $\Theta := \mathsf{AS}^{\mathsf{R},\mathsf{DS}_2}$ is* unlinkable *if for every* PPT *adversary $\mathcal{A}$ there exists a negligible function* negl *such that:*

$$|\Pr[\mathsf{Unlinkability}^{\overline{k}}_{\mathcal{A},\Theta}(n, 0)] - \Pr[\mathsf{Unlinkability}^{\overline{k}}_{\mathcal{A},\Theta}(n, 1)]| \leq \mathsf{negl}(n)$$

*where $\mathsf{Unlinkability}^{\overline{k}}_{\mathcal{A},\Theta}$ is defined in Figure 29.*

The unlinkability notion is defined w.r.t. an adversarial eavesdropper that is neither the *Adaptor* nor the *Extractor*. This adversary is trying to distinguish between signatures that were computed in a signing protocol, and signatures that were adapted. If the eavesdropper was either the *Adaptor* or the *Extractor*, they would easily know whether a signature was adapted since they participated in the two-party protocols.

$\mathsf{Unlinkability}^{\overline{k}}_{\mathcal{A},\Theta}(n,b)$

$(\mathsf{vk},(\mathsf{vk}_E,\mathsf{sk}_E),(\mathsf{vk}_A,\mathsf{sk}_A)) \leftarrow \mathsf{KeyGen}(1^n)$
$\mathcal{O} := \{\mathsf{OSign},\mathsf{OPreSign},\mathsf{OChall}\}$
$b^* \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{vk},\mathsf{vk}_E,\mathsf{vk}_A)$
**return** $b = b^*$

$\mathsf{OChall}(m,x,w)$

**assert** $(x,w) \in \mathsf{R}$
$\hat{\sigma} \leftarrow \mathsf{PreSign}\langle(\mathsf{sk}_A,w),(\mathsf{sk}_E)\rangle(m,x)$
$\sigma_0 := \mathsf{Adapt}(\mathsf{vk},\hat{\sigma},w,st_{\overline{k}-1})$
$\sigma_1 \leftarrow \mathsf{Sign}\langle\mathsf{sk}_A,\mathsf{sk}_E\rangle(m)$
**return** $\sigma_b$

$\mathsf{OPreSign}(m,x,w)$

**assert** $(x,w) \in \mathsf{R}$
$\hat{\sigma} \leftarrow \mathsf{PreSign}\langle(\mathsf{sk}_A,w),(\mathsf{sk}_E)\rangle(m,x)$
$\sigma \leftarrow \mathsf{Adapt}(\mathsf{vk},\hat{\sigma},w,st_{\overline{k}-1})$
**return** $(\hat{\sigma},\sigma)$

$\mathsf{OSign}(m)$

$\sigma \leftarrow \mathsf{Sign}\langle\mathsf{sk}_A,\mathsf{sk}_E\rangle(m)$
**return** $\sigma$

Fig. 29: Unlinkability experiment of a 2P-AISO-AS scheme. We denote by Sign and PreSign the execution of their protocols.

**Definition 45 (Computational preverify soundness).** *A 2P-AISO-AS scheme satisfies* computational preverify soundness *if for every* PPT *adversary* $\mathcal{A}$ *and for every* $n \in \mathbb{N}$ *and polynomially bounded* $x \notin L_{\mathsf{R}}$*, it holds that:*

$$\Pr\left[b \;\middle|\; \begin{array}{c}(\mathsf{vk},(\mathsf{vk}_E,\mathsf{sk}_E),(\mathsf{vk}_A,\mathsf{sk}_A)) \leftarrow \mathsf{DS}_2.\mathsf{KeyGen}(1^n)\\ (m,\hat{\sigma}) \leftarrow \mathcal{A}(\mathsf{vk},(\mathsf{vk}_E,\mathsf{sk}_E),(\mathsf{vk}_A,\mathsf{sk}_A))\end{array}\right] \leq \mathsf{negl}(n)$$

*where* $b := \mathsf{PreVrfy}(\mathsf{vk},m,\hat{\sigma},x)$.

Computational preverify soundness models the fact that it is not possible to use a statement $x$ that is not in the language of the relation $L_{\mathsf{R}}$ in order to compute a verifying presignature. This definition prevents the usage of ill-formed statements to compute presignatures that, although verifying, would not extract a valid witness.

## J Two-Party Solitary-Output Adaptor Signature (2P-SO-AS)

In this section we present a definition of Two-Party Solitary-Output Adaptor Signature (2P-SO-AS). 2P-SO-AS is similar to 2P-AISO-AS with the only difference that the *Adaptor* does not input the witness $w$ into the presigning protocol. Thus, the interface of 2P-SO-AS resembles that of previous definitions of adaptor signatures and thus it cannot be instantiated from unique signature schemes [16]. In contrast to previous notions however, in 2P-SO-AS we explicitly include the unfair delivery guarantees of two-party protocols [18] as part of the interface and security notions. Thus, 2P-SO-AS can bridge previous notions with 2P-AISO-AS.

As in 2P-AISO-AS, in 2P-SO-AS we model the two-party presign protocol as an interactive protocol that takes $\overline{k}$ interactions or steps and delivers the final presignature to the *Extractor*. Both *Adaptor* and *Extractor* can abort this interactive protocol at any point. Starting with step number 1, each party takes turns in executing the steps of the presign protocol until the last step, step number $\overline{k}$, is executed by the *Extractor*. Running step number $\overline{k}$ finalizes the protocol. By the end of the protocol, the *Extractor* obtains the presignature $\hat{\sigma}$, which is the output of the protocol. Notice that, if $\overline{k}$ is even, then the *Extractor* must run the even steps of the protocol. If $\overline{k}$ is odd instead, then the *Extractor* runs the odd steps of the protocol. Thus, the first step of the protocol is run by the *Extractor* if $\overline{k}$ is odd. Otherwise, the first step of the protocol is run by the *Adaptor*.

**Definition 46 (Two-Party Asymmetric-Input Solitary-Output Adaptor Signature Scheme).** *A $\overline{k}$-interaction Two-Party Solitary-Output Adaptor Signature (2P-SO-AS) scheme is defined w.r.t. a relation $R \subseteq (\mathcal{X}, \mathcal{W})$ and a two-party signature scheme $\mathsf{DS}_2 = (\mathsf{Setup}, \mathsf{KeyGen}, \{\mathsf{Sign}_i\}_{i\in[k]}, \mathsf{Vrfy})$. It is run between parties Adaptor (A) and Extractor (E), and consists of a tuple $\mathsf{AS}_2^{R,\mathsf{DS}_2} = (\{\mathsf{PreSign}_i\}_{i\in[\overline{k}]}, \mathsf{Adapt}, \mathsf{PreVrfy}, \mathsf{Extract})$ of efficient algorithms defined as:*

- *A succession of* PPT *algorithms $\{\mathsf{PreSign}_i\}_{i\in[\overline{k}]}$. Each $\mathsf{PreSign}_i$ algorithm representing a single step in the $\overline{k}$-interaction presign protocol. They are defined, with respect to message $m \in \{0,1\}^n$, the keys $(\mathsf{vk}, (\mathsf{vk}_A, \mathsf{sk}_A), (\mathsf{vk}_E, \mathsf{sk}_E))$ and an instance of the hard relation $(x, w) \in \mathsf{R}$. The $\mathsf{PreSign}_i$ algorithms are defined as follows:*
  - *The output of $\mathsf{PreSign}_j$ for $j < \overline{k}$ is $(pm_j, st_j)$ where $pm_j$ is a protocol message and $st_j$ is a protocol state.*
  - *The output of $\mathsf{PreSign}_{\overline{k}}$ is $\hat{\sigma}$, where $\hat{\sigma}$ is a presignature.*
  - *The input of $\mathsf{PreSign}_1$ is $(\mathsf{sk}, m, x)$ where $\mathsf{sk}$ is the signing key of the Adaptor $\mathsf{sk}_A$ if $\overline{k}$ is even, or the signing key of the Extractor $\mathsf{sk}_E$ otherwise.*
  - *The input of $\mathsf{PreSign}_2$ is $(\mathsf{sk}, m, x, pm_1)$ where $\mathsf{sk}$ is the signing key of the Adaptor $\mathsf{sk}_A$ if $\overline{k}$ is odd, or the signing key of the Extractor $\mathsf{sk}_E$ otherwise.*
  - *The input of $\mathsf{PreSign}_j$ for $3 \leq j \leq \overline{k}$ is $(st_{j-2}, pm_{j-1})$.*
- *$b := \mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma})$: is a* DPT *algorithm that on input a verification key $\mathsf{vk}$, a message $m \in \mathcal{M}$, a statement $x \in L_{\mathsf{R}}$ and a presignature $\hat{\sigma}$, outputs a bit $b$.*
- *$\sigma := \mathsf{Adapt}(\mathsf{vk}, \hat{\sigma}, w, st_{\overline{k}-1})$: is a* DPT *algorithm that on input a verification key $\mathsf{vk}$, a presignature $\hat{\sigma}$, a witness $w$, and a protocol state, outputs a signature $\sigma$.*
- *$w := \mathsf{Extract}(\mathsf{vk}, \sigma, \hat{\sigma}, x)$: is a* DPT *algorithm that on input a verification key $\mathsf{vk}$, a signature $\sigma$, a presignature $\hat{\sigma}$ and a statement $x \in L_{\mathsf{R}}$, outputs a witness $w$ s.t. $(x, w) \in R$, or an error symbol $\perp$.*

For ease of readability, we next define the notion of presignature protocol for 2P-SO-AS. This encapsulates the interactions between the adaptor $A$ and the extractor $E$ to obtain presignature $\hat{\sigma}$. Notice that, by construction, $E$ will get the output of the presignature protocol.

**Definition 47 (Presignature protocol).** *The presignature protocol $\Pi\mathsf{PreSign}\langle A(\mathsf{sk}_A), E(\mathsf{sk}_E)\rangle(m, x)$ between party Adaptor (A) on input $(\mathsf{sk}_A, m, x)$ and party Extractor (E) on input $(\mathsf{sk}_E, m, x)$ is defined as:*

- *If the number of interactions $\overline{k}$ is odd, E starts by running $\mathsf{PreSign}_1$. Otherwise, A starts by running $\mathsf{PreSign}_1$. After the first step, the parties alternate turns running $\mathsf{PreSign}_i$ in order for $i = 2$ until $i = \overline{k}$.*
- *On each step $i < \overline{k}$ the party $P \in \{A, E\}$ runs $\mathsf{PreSign}_i$ to generate $(pm_i, st_i)$. Then $P$ stores $st_i$ and sends $pm_i$ to the other party.*
- *On the last step $i = \overline{k}$, the Extractor runs $\mathsf{PreSign}_{\overline{k}}$ to generate a presignature $\hat{\sigma}$.*

**Correctnes.** We show the correctness of 2P-SO-AS.

**Definition 48 (Two-Party Presignature Correctness).** *A 2P-SO-AS scheme $\mathsf{AS}^{\mathsf{R},\mathsf{DS}_2}$ satisfies two-party presignature correctness if for every $n \in \mathbb{N}$, and every message $m \in \mathcal{M}$, the following holds:*

$$\Pr\left[\begin{array}{l|r} & (\mathsf{vk}, (\mathsf{vk}_A, \mathsf{sk}_A), (\mathsf{vk}_E, \mathsf{sk}_E)) \leftarrow \mathsf{DS}_2.\mathsf{KeyGen}(1^n) \\ & (x, w) \leftarrow \mathsf{GenR}(1^n) \\ b_0 \wedge & \hat{\sigma} \leftarrow \Pi_{\mathsf{PreSign}\langle \mathsf{sk}_A, \mathsf{sk}_E\rangle}(m, x) \\ b_1 \wedge & b_0 := \mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma});\ \sigma := \mathsf{Adapt}(\mathsf{vk}, \hat{\sigma}, w, st_{\overline{k}-1}) \\ b_2 & b_1 := \mathsf{Vrfy}(\mathsf{vk}, m, \sigma);\ w' := \mathsf{Extract}(\mathsf{vk}, \sigma, \hat{\sigma}, x) \\ & b_2 := (x, w') \in R \end{array}\right] = 1.$$

### J.1 Security and Privacy Notions

In this section, we define the security and privacy notions of interest for 2P-SO-AS.

**Definition 49 (Full extractability).** *A 2P-SO-AS scheme is* fully extractable *if for every* PPT *adversary* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}$ *such that* $\Pr[\textit{2P-SO-AS-}\mathsf{FExt}^{\overline{k}}_{\mathcal{A},\Theta}(n) = 1] \leq \mathsf{negl}(n)$*, where the experiment* $\textit{2P-SO-AS-}\mathsf{FExt}^{\overline{k}}_{\mathcal{A},\Theta}$ *is defined in Figure 30.*

In the full extractability game (Figure 30) the challenger assumes the role of the *Extractor*, while the adversary plays the role of the *Adaptor*.

Due to the presign protocol having multiple steps, the game provides oracles $\mathsf{OPreSignInit}$ and $\mathsf{OPreSignInteraction}$. These oracles allow the adversary to query the challenger for an advancement in an instance of the presign protocol. Oracle $\mathsf{OPreSignInit}$ is the first step of the presign protocol that the challenger (*Extractor*) runs, and $\mathsf{OPreSignInteraction}$ are the subsequent steps that the challenger runs. The oracle $\mathsf{OPreSignInit}$ initiates an instance of the presign protocol, tagged by a unique session id *sid*. The adversary may run multiple instances of the presign protocol concurrently. To keep track of the different instances, the oracle $\mathsf{OPreSignInit}$ stores each unique session id *sid* in a table *ID*. A query to $\mathsf{OPreSignInteraction}$ must specify both the instance to advance (by providing the session id *sid*) and which step $i$ of the protocol to run, i.e. which of the $\{\mathsf{PreSign}_i\}_{i\in[\overline{k}]}$ algorithms to run. We ensure that the adversay queries for steps in the correct order (without duplicating a step) with the assertions of lines 3 and 4 of $\mathsf{OPreSignInteraction}$, which we will explain in detail in the following.

With table *St* we keep track of the state of the different presign sessions. The table maps a pair of session id *sid* and step $i$ with a protocol state $st_i$. This $st_i$ is the protocol state that the challenger stores after the $i$'th step of the presign protocol with session id *sid*. Line 3 of oracle $\mathsf{OPreSignInteraction}$ ensures that an adversary cannot query the oracle to run a presigning step $\mathsf{PreSign}_i$ more than once for the same session. Line 4 of oracle $\mathsf{OPreSignInteraction}$ ensures that an adversary cannot query step number $i$ without first querying the previous steps of the same session, thus enforcing a correct execution order.

Notice that, by the definition of the presign protocol (Definition 47), if the number of interactions $\overline{k}$ is odd, then the challenger (*Extractor*) must only run the odd steps ($\mathsf{PreSign}_1$, $\mathsf{PreSign}_3$, etc.) of the protocol. Otherwise, the challenger (*Extractor*) must only run the even steps of the protocol. To ensure this, the game checks the parity of $\overline{k}$ several times (assert on line 1 of $\mathsf{OPreSignInteraction}$, and branching on lines 13 and 18 of $\mathsf{OPreSignInit}$). There is a special case if the number of steps of the presign protocol $\overline{k}$ is 2. In this case, the challenger (*Extractor*) only runs one step of the protocol. Therefore, in this case, the oracle $\mathsf{OPreSignInteraction}$ is not used (there would be no steps beyond $\mathsf{PreSign}_1$ or $\mathsf{PreSign}_2$). This implies that oracle $\mathsf{OPreSignInit}$ should start and finalize a presign protocol, returning the presignature $\hat{\sigma}$. This special case is handled by the branching on line 3 of the oracle $\mathsf{OPreSignInit}$.

At the end of the game, the adversary presents a forgery message and signature $(m^*, \sigma^*)$. This forgery is valid if it satisfies conditions $c_1$, $c_2$ and $c_3$. Condition $c_1$ and $c_2$ checks that the forgery verifies and that the message was not queried to the signing oracles. Condition $c_3$ checks that the *Extractor* cannot extract a valid witness from the forgery signature. To be more precise, $c_3$ first collects for all the presignatures generated on the forgery message $m^*$ and any statement $x$. Then, $c_3$ checks that running $\mathsf{Extract}$ on the forgery signature and each of these presignatures yields an invalid witness. We say that the witness is invalid because it does not correspond to the statement $x$ used to generate the presignature.

**Definition 50 (full unforgeability).** *A 2P-SO-AS scheme* $\Theta := \mathsf{AS}^{\mathsf{R},\mathsf{DS}_2}$ *is* fully unforgeable *if for every* PPT *adversary* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}$ *such that* $\Pr[\mathsf{WExt}^{\overline{k}}_{\mathcal{A},\Theta}(n) = 1] \leq \mathsf{negl}(n)$*, where the experiment* $\mathsf{WExt}^{\overline{k}}_{\mathcal{A},\Theta}$ *is defined in Figure 31.*

In full unforgeability, the challenger assumes the role of the *Adaptor*, while the adversary plays the role of the *Extractor*. The adversary can advance steps of the presign protocol in the same way as in the full extractability game: by querying oracles $\mathsf{OPreSignInit}$ and $\mathsf{OPreSignInteraction}$. By the definition of the presign protocol (Definition 47), if the number of steps $\overline{k}$ is odd, then the challenger (*Adaptor*) must run

$\mathsf{FExt}^{\overline{k}}_{\mathcal{A},\Theta}(n)$

1 : $\mathcal{Q} := \mathcal{X} := \emptyset$
2 : $St := ID := \mathcal{T}_E := \bot$
3 : $(\mathsf{vk}, (\mathsf{vk}_E, \mathsf{sk}_E), (\mathsf{vk}_A, \mathsf{sk}_A))$
4 : $\quad \leftarrow\!\!\!\!\!\!\$\ \mathsf{KeyGen}(1^n)$
5 : $\mathcal{O} := \{\mathsf{OSign}, \mathsf{ONewStatement}$
6 : $\quad \mathsf{OPreSignInit},$
7 : $\quad \mathsf{OPreSignInteraction}\}$
8 : $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{vk}, \mathsf{vk}_E, (\mathsf{vk}_A, \mathsf{sk}_A))$
9 : $c_1 := m^* \notin \mathcal{Q}$
10 : $c_2 := \mathsf{Vrfy}(\mathsf{vk}, m^*, \sigma^*) = 1$
11 : $c_3 := \forall sid \in ID[m^*]:$
12 : $\quad \textbf{let}(x, \hat{\sigma}) := \mathcal{T}_E[sid]$ s.t. $x \notin \mathcal{X}$.
13 : $\quad (x, \mathsf{Extract}(\mathsf{vk}, \sigma^*, \hat{\sigma}, x)) \notin \mathsf{R}$
14 : **return** $(c_1 \wedge c_2 \wedge c_3)$

$\mathsf{ONewStatement}()$

1 : $(x, w) \leftarrow \mathsf{GenR}(1^n)$
2 : $\mathcal{X} := \mathcal{X} \cup \{x\}$
3 : **return** $x$

$\mathsf{OPreSignInit}(x, m, pm)$

1 : $sid \leftarrow_{\$} \{0,1\}^*$
2 : $ID[m] := ID[m] \cup \{sid\}$
3 : **if** $\overline{k} = 2$ // Last step
4 : $\quad \hat{\sigma} \leftarrow \mathsf{PreSign}_2(\mathsf{sk}_E, x, m, pm)$
5 : $\quad$ **assert** $\mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma})$
6 : $\quad \mathcal{T}_E[sid] := (x, \hat{\sigma})$
7 : $\quad$ **return** $\hat{\sigma}$
8 : $\mathcal{T}_E[sid] := (x, \bot)$
9 : **if** $\overline{k} \equiv 1 \bmod 2$ // *Extractor* starts
10 : $\quad (pm_1, st_1)$
11 : $\quad \leftarrow\!\!\!\!\!\!\$\ \mathsf{PreSign}_1(\mathsf{sk}_E, x, m)$
12 : $\quad St[sid, 1] := st_1$
13 : $\quad$ **return** $(sid, pm_1)$
14 : **else** // *Adaptor* (adversary) starts
15 : $\quad (pm_2, st_2)$
16 : $\quad \leftarrow\!\!\!\!\!\!\$\ \mathsf{PreSign}_2(\mathsf{sk}_E, x, m, pm)$
17 : $\quad St[sid, 2] := st_2$
18 : $\quad$ **return** $(sid, pm_2)$

$\mathsf{OPreSignInteraction}(sid, pm, i)$

1 : **assert** $i \equiv \overline{k} \bmod 2$
2 : **assert** $\exists m : sid \in ID[m]$
3 : **assert** $St[sid, i] = \bot$
4 : **assert** $St[sid, i-2] \neq \bot$
5 : $st_{i-2} := St[sid, i-2]$
6 : **if** $i \neq \overline{k}$ // Not last step
7 : $\quad (pm_i, st_i)$
8 : $\quad \leftarrow\!\!\!\!\!\!\$\ \mathsf{PreSign}_i(st_{i-2}, pm)$
9 : $\quad St[sid, i] := st_i$
10 : $\quad$ **return** $pm_i$
11 : **else** // Last step
12 : $\quad \hat{\sigma} \leftarrow \mathsf{PreSign}_{\overline{k}}(st_{i-2}, pm)$
13 : $\quad (x, _) := \mathcal{T}_E[sid]$
14 : $\quad$ **assert** $\mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma})$
15 : $\quad \mathcal{T}_E[sid] := (x, \hat{\sigma})$
16 : $\quad$ **return** $\hat{\sigma}$

Fig. 30: Full extractability game where the adversary corrupts the adaptor party $A$. The oracle $\mathsf{OSign}$ is defined as in two-party digital signatures, and $\mathcal{Q}$ is the set of messages queried to the signing oracle.

the even steps. Otherwise, if $\overline{k}$ is even, then the challenger must run the odd steps. We ensure this with the branching on lines 8 and 13 of oracle $\mathsf{OPreSignInit}$ and line 1 of oracle $\mathsf{OPreSignInteraction}$.

As in full extractability, the adversary can run multiple instances of the presign protocol. Each instance of the presign protocol is tagged with a session id $sid$ as in full extractability. The table $St$ keeps track of the state of the different presign sessions the same way as in full extractability. As in full extractability we also ensure that no step is duplicated during the same session (oracle $\mathsf{OPreSignInteraction}$ line 3) and that no step can be queried before the previous step (oracle $\mathsf{OPreSignInteraction}$ line 4).

One key difference between full unforgeability from 2P-SO-AS and weak unforgeability of 2P-AISO-AS, is that in weak unforgeability, the signing oracle may also leak witnesses to presginatures if the underlying signature scheme is unique. In full unforgeability from 2P-SO-AS however, this is not the case due to the impossibility of instantiating 2P-SO-AS from unique signature schemes.

**Definition 51 (Adaptability and designated adaptability).** *For any message $m \in \mathcal{M}$, any verification key $\mathsf{vk}$, any statement/witness pair $(x, w) \in \mathsf{R}$, and any presignature $\hat{\sigma}$ with $\mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma}) = 1$, we say that a 2P-AISO-AS scheme satisfies* designated adaptability *if there exists a protocol state $st$ for which it holds that $\mathsf{Vrfy}(\mathsf{vk}, m, \mathsf{Adapt}(\mathsf{vk}, \hat{\sigma}, w, st)) = 1$. Alternatively, we say that a 2P-AISO-AS scheme satisfies* adaptability *if for every protocol state $st$ it holds that $\mathsf{Vrfy}(\mathsf{vk}, m, \mathsf{Adapt}(\mathsf{vk}, \hat{\sigma}, w, st)) = 1$.*

**Definition 52 (Unlinkability).** *A 2P-SO-AS scheme $\Theta := \mathsf{AS}^{\mathsf{R},\mathsf{DS}_2}$ is* unlinkable *if for every* PPT *adversary $\mathcal{A}$ there exists a negligible function* $\mathsf{negl}$ *such that:*

$$|\Pr[\mathsf{Unlinkability}^{\overline{k}}_{\mathcal{A},\Theta}(n, 0)] - \Pr[\mathsf{Unlinkability}^{\overline{k}}_{\mathcal{A},\Theta}(n, 1)]| \leq \mathsf{negl}(n)$$

*where $\mathsf{Unlinkability}^{\overline{k}}_{\mathcal{A},\Theta}$ is defined in Figure 32.*

$\mathsf{FUnf}^{\overline{k}}_{\mathcal{A},\Theta}(n)$

$\mathcal{Q} := \mathcal{W} := ID := \emptyset$
$St := \bot$
$(\mathsf{vk}, (\mathsf{vk}_E, \mathsf{sk}_E), (\mathsf{vk}_A, \mathsf{sk}_A))$
$\quad \leftarrow \mathsf{KeyGen}(1^n)$
$\mathcal{O} := \{\mathsf{OSign}, \mathsf{ONewStatement}$
$\quad \mathsf{OAdapt}, \mathsf{OPreSignInit},$
$\quad \mathsf{OPreSignInteraction}\}$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{vk}, \mathsf{vk}_A, (\mathsf{vk}_E, \mathsf{sk}_E))$
$c_1 := m^* \notin \mathcal{Q}$
$c_2 := \mathsf{Vrfy}(\mathsf{vk}, m^*, \sigma^*) = 1$
**return** $(c_1 \wedge c_2)$

$\mathsf{OAdapt}(\hat{\sigma}, x, m, sid)$

**assert** $\exists w : (x, w) \in \mathcal{W}$
**assert** $\mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma})$
$\mathcal{Q} := \mathcal{Q} \cup \{m\}$
$\mathcal{Q} := \mathcal{Q} \cup \{m : x \in \mathcal{T}_A[m]\}$
$\mathcal{W} := \mathcal{W} \setminus \{(x, w)\}$
$st_{\overline{k}-1} := St[sid, \overline{k} - 1]$
**return** $\mathsf{Adapt}(\hat{\sigma}, w, st_{\overline{k}-1})$

$\mathsf{OPreSignInit}(x, m, pm)$

$\mathcal{T}_A[m] := \mathcal{T}_A[m] \cup \{x\}$
**if** $\nexists w : (x, w) \in \mathcal{W}$
$\quad \mathcal{Q} := \mathcal{Q} \cup \{m\}$
$sid \leftarrow_{\$} \{0,1\}^*$
$ID := ID \cup \{sid\}$
**if** $\overline{k} \equiv 0 \bmod 2$ // *Adaptor* starts
$\quad (pm_1, st_1)$
$\quad \leftarrow \mathsf{PreSign}_1(\mathsf{sk}_A, m, x)$
$\quad St[sid, 1] := st_1$
$\quad$ **return** $(sid, pm_1)$
**else** // *Extractor* (adversary) starts
$\quad (pm_2, st_2)$
$\quad \leftarrow \mathsf{PreSign}_2(\mathsf{sk}_A, m, x)$
$\quad St[sid, 2] := st_2$
$\quad$ **return** $(sid, pm_2)$

$\mathsf{OPreSignInteraction}(sid, pm, i)$

**assert** $i \not\equiv \overline{k} \bmod 2$
**assert** $sid \in ID$
**assert** $St[sid, i] = \bot$
**assert** $St[sid, i-2] \neq \bot$
$st_{i-2} := St[sid, i-2]$
$(pm_i, st_i)$
$\quad \leftarrow \mathsf{PreSign}_i(st_{i-2}, pm)$
$St[sid, i] := st_i$
**return** $pm_i$

$\mathsf{ONewStatement}()$

$(x, w) \leftarrow \mathsf{GenR}(1^n)$
$\mathcal{W} := \mathcal{W} \cup \{(x, w)\}$
**return** $x$

$\mathsf{OSign}(m)$

$\mathcal{Q} := \mathcal{Q} \cup \{m\}$
**return** $\Pi\mathsf{Sign}(m)$

Fig. 31: Full unforgeability game where the adversary corrupts the *Extractor* party $E$. We denote by $\Pi\mathsf{Sign}$ the two-party signature protocol.

The unlinkability notion is defined w.r.t. an adversarial eavesdropper that is neither the *Adaptor* nor the *Extractor*. This adversary is trying to distinguish between signatures that were computed in a signing protocol, and signatures that were adapted. If the eavesdropper was either the *Adaptor* or the *Extractor*, they would easily know whether a signature was adapted since they participated in the two-party protocols.

**Definition 53 (Computational preverify soundness).** *A 2P-SO-AS scheme satisfies* computational preverify soundness *if for every* PPT *adversary* $\mathcal{A}$ *and for every* $n \in \mathbb{N}$ *and polynomially bounded* $x \notin L_{\mathsf{R}}$*, it holds that:*

$$\Pr\left[b \;\middle|\; \begin{array}{c}(\mathsf{vk}, (\mathsf{vk}_E, \mathsf{sk}_E), (\mathsf{vk}_A, \mathsf{sk}_A)) \leftarrow \mathsf{DS}_2.\mathsf{KeyGen}(1^n) \\ (m, \hat{\sigma}) \leftarrow \mathcal{A}(\mathsf{vk}, (\mathsf{vk}_E, \mathsf{sk}_E), (\mathsf{vk}_A, \mathsf{sk}_A))\end{array}\right] \leq \mathsf{negl}(n)$$

*where* $b := \mathsf{PreVrfy}(\mathsf{vk}, m, \hat{\sigma}, x)$.

Computational preverify soundness models the fact that it is not possible to use a statement $x$ that is not in the language of the relation $L_{\mathsf{R}}$ in order to compute a verifying presignature. This definition prevents the usage of ill-formed statements to compute presignatures that, although verifying, would not extract a valid witness.

### J.2 Lifting A 2P-SO-AS Instantiation Into A 2P-AISO-AS Instantiation

Assume that we have a secure 2P-SO-AS instantiation $\Theta$. We can construct a 2P-AISO-AS scheme $H$ from $\Theta$ by simply ignoring the extra input of the *Adaptor* $w$ in the $\mathsf{PreSign}$ protocol. More

$\mathsf{Unlinkability}^{\overline{k}}_{\mathcal{A},\Theta}(n,b)$

$(\mathsf{vk},(\mathsf{vk}_E,\mathsf{sk}_E),$
$\quad(\mathsf{vk}_A,\mathsf{sk}_A)) \leftarrow \mathsf{KeyGen}(1^n)$
$\mathcal{O} := \{\mathsf{OSign},\mathsf{OPreSign},$
$\quad\mathsf{OChall}\}$
$b^* \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{vk},\mathsf{vk}_E,\mathsf{vk}_A)$
**return** $b = b^*$

$\mathsf{OChall}(m,x,w)$

**assert** $(x,w) \in \mathsf{R}$
$\hat{\sigma} \leftarrow \mathsf{PreSign}\langle \mathsf{sk}_A,\mathsf{sk}_E\rangle(m,x)$
$\sigma_0 := \mathsf{Adapt}(\mathsf{vk},\hat{\sigma},w,st_{\overline{k}-1})$
$\sigma_1 \leftarrow \mathsf{Sign}\langle \mathsf{sk}_A,\mathsf{sk}_E\rangle(m)$
**return** $\sigma_b$

$\mathsf{OPreSign}(m,x,w)$

**assert** $(x,w) \in \mathsf{R}$
$\hat{\sigma} \leftarrow \mathsf{PreSign}\langle \mathsf{sk}_A,\mathsf{sk}_E\rangle(m,x)$
$\sigma \leftarrow \mathsf{Adapt}(\mathsf{vk},\hat{\sigma},w,st)$
**return** $(\hat{\sigma},\sigma)$

$\mathsf{OSign}(m)$

$\sigma \leftarrow \mathsf{Sign}\langle \mathsf{sk}_A,\mathsf{sk}_E\rangle(m)$
**return** $\sigma$

Fig. 32: Unlinkability experiment of a 2P-SO-AS scheme. We denote by Sign and PreSign the execution of their protocols.

concretely $H.\Pi\mathsf{PreSign}\langle(\mathsf{sk}_A,w),(\mathsf{sk}_E)\rangle(m,x) := \Theta.\Pi\mathsf{PreSign}\langle(\mathsf{sk}_A),(\mathsf{sk}_E)\rangle(m,x)$. Or even more explicitlty: if $\overline{k}$ is even, then $H.\mathsf{PreSign}_1(\mathsf{sk}_A,w,m,x) := \Theta.\mathsf{PreSign}_1(\mathsf{sk}_A,m,x)$, otherwise if $\overline{k}$ is odd, then $H.\mathsf{PreSign}_2(\mathsf{sk}_A,w,m,x,pm_1) = \Theta.\mathsf{PreSign}_2(\mathsf{sk}_A,m,x,pm_1)$ The rest of the algorithms remain unchanged.

**Lemma 19.** *If $\Theta$ is correct, then $H$ is correct.*

*Proof.* Trivially, the following equality holds

$$\Pr\left[\begin{array}{l|r} & (\mathsf{vk},(\mathsf{vk}_A,\mathsf{sk}_A),(\mathsf{vk}_E,\mathsf{sk}_E)) \\ & \leftarrow \mathsf{DS}_2.\mathsf{KeyGen}(1^n) \\ & (x,w) \leftarrow \mathsf{GenR}(1^n) \\ b_0\wedge & \hat{\sigma} \leftarrow H.\Pi_{\mathsf{PreSign}\langle(\mathsf{sk}_A,w),(\mathsf{sk}_E)\rangle}(m,x) \\ b_1\wedge & b_0 := H.\mathsf{PreVrfy}(\mathsf{vk},m,x,\hat{\sigma}) \\ b_2 & \sigma := H.\mathsf{Adapt}(\mathsf{vk},\hat{\sigma},w,st_{\overline{k}-1}) \\ & b_1 := \mathsf{Vrfy}(\mathsf{vk},m,\sigma) \\ & w' := H.\mathsf{Extract}(\mathsf{vk},\sigma,\hat{\sigma},x) \\ & b_2 := (x,w') \in R \end{array}\right] = \Pr\left[\begin{array}{l|r} & (\mathsf{vk},(\mathsf{vk}_A,\mathsf{sk}_A),(\mathsf{vk}_E,\mathsf{sk}_E)) \\ & \leftarrow \mathsf{DS}_2.\mathsf{KeyGen}(1^n) \\ & (x,w) \leftarrow \mathsf{GenR}(1^n) \\ b_0\wedge & \hat{\sigma} \leftarrow \Theta.\Pi_{\mathsf{PreSign}\langle(\mathsf{sk}_A),(\mathsf{sk}_E)\rangle}(m,x) \\ b_1\wedge & b_0 := \Theta.\mathsf{PreVrfy}(\mathsf{vk},m,x,\hat{\sigma}) \\ b_2 & \sigma := \Theta.\mathsf{Adapt}(\mathsf{vk},\hat{\sigma},w,st_{\overline{k}-1}) \\ & b_1 := \mathsf{Vrfy}(\mathsf{vk},m,\sigma) \\ & w' := \Theta.\mathsf{Extract}(\mathsf{vk},\sigma,\hat{\sigma},x) \\ & b_2 := (x,w') \in R \end{array}\right] = 1.$$

**Lemma 20.** *If $\Theta$ satisfies adaptability, then $H$ satisfies adaptability.*

*Proof.* Given any message $m \in \mathcal{M}$, any verification key $\mathsf{vk}$, any statement/witness pair $(x,w) \in \mathsf{R}$, any presignature $\hat{\sigma}$ with $\Theta.\mathsf{PreVrfy}(\mathsf{vk},m,x,\hat{\sigma}) = 1$, and any protocol state $st$, since $\Theta$ satisfies adaptability, we know that $\mathsf{Vrfy}(\mathsf{vk},m,\Theta.\mathsf{Adapt}(\mathsf{vk},\hat{\sigma},w,st)) = 1$. Given that $\Theta.\mathsf{PreVrfy} = H.\mathsf{PreVrfy}$ and $\Theta.\mathsf{Adapt} = H.\mathsf{Adapt}$, then $H$ also satisfies adaptability.

**Lemma 21.** *If $\Theta$ satisfies designated adaptability, then $H$ satisfies designated adaptability.*

*Proof.* Given any message $m \in \mathcal{M}$, any verification key $\mathsf{vk}$, any statement/witness pair $(x,w) \in \mathsf{R}$, and any presignature $\hat{\sigma}$ with $\mathsf{PreVrfy}(\mathsf{vk},m,x,\hat{\sigma}) = 1$, since $\Theta$ satisfies designated adaptability, we know that there exists a protocol state $st$ for which it holds that $\mathsf{Vrfy}(\mathsf{vk},m,\mathsf{Adapt}(\mathsf{vk},\hat{\sigma},w,st)) = 1$. Given that $\Theta.\mathsf{PreVrfy} = H.\mathsf{PreVrfy}$ and $\Theta.\mathsf{Adapt} = H.\mathsf{Adapt}$, then $H$ also satisfies designated adaptability.

**Lemma 22.** *If $\Theta$ satisfies unlinkability, then $H$ satisfies unlinkability.*

*Proof.* Given that $H.\Pi\mathsf{PreSign}\langle(\mathsf{sk}_A, w), (\mathsf{sk}_E)\rangle(m, x) := \Theta.\Pi\mathsf{PreSign}\langle(\mathsf{sk}_A), (\mathsf{sk}_E)\rangle(m, x)$, we get that signing or presigning and then adapting runs the exact same procedures in both unlinkability for $\Theta$ and in unlinkability for $H$ the. Therefore if $\Theta$ is unlinkable, then $H$ is also unlinkable.

**Lemma 23.** *If $\Theta$ satisfies preverify soundness, then $H$ satisfies preverify soundness.*

*Proof.* Given that $H.\mathsf{PreVrfy} = \Theta.\mathsf{PreVrfy}$, if for every $\mathsf{PPT}$ adversary $\mathcal{A}$ and for every $n \in \mathbb{N}$ and polynomially bounded $x \notin L_{\mathsf{R}}$, it holds that:

$$\Pr\left[b \;\middle|\; \begin{array}{c}(\mathsf{vk}, (\mathsf{vk}_E, \mathsf{sk}_E), (\mathsf{vk}_A, \mathsf{sk}_A)) \leftarrow \mathsf{DS}_2.\mathsf{KeyGen}(1^n) \\ (m, \hat{\sigma}) \leftarrow \mathcal{A}(\mathsf{vk}, (\mathsf{vk}_E, \mathsf{sk}_E), (\mathsf{vk}_A, \mathsf{sk}_A))\end{array}\right] \leq \mathsf{negl}(n)$$

where $b := \Theta.\mathsf{PreVrfy}(\mathsf{vk}, m, \hat{\sigma}, x)$, then the same probability holds for $b := H.\mathsf{PreVrfy}(\mathsf{vk}, m, \hat{\sigma}, x)$.

**Lemma 24.** *If $\Theta$ satisfies full extractability, then $H$ satisfies full extractability.*

*Proof.* Notice that, by our definition of $H$, all the presign steps that the *Extractor* runs in $H$ are exactly the same as the steps that the *Extractor* runs in $\Theta$. Moreover, the full extractability games for 2P-AISO-AS (Figure 27) and 2P-SO-AS (Figure 30) are identical. Therefore if $\Theta$ satisfies full extractability, then $H$ also satisfies full extractability.

**Lemma 25.** *If $\Theta$ satisfies full unforgeability, then $H$ satisfies weak unforgeability.*

*Proof.* Assume that there exists a $\mathsf{PPT}$ adversary $\mathcal{A}$that breaks weak unforgeability of $H$, then we can create a $\mathsf{PPT}$ adversary $\mathcal{B}$that uses $\mathcal{A}$to break full unforgeability of $\Theta$. The challenger plays the full unforgeability game (2P-SO-AS) with $\mathcal{B}$and $\mathcal{B}$plays the weak unforgeability game (2P-AISO-AS) with $\mathcal{A}$. The challenger starts by sampling keys and sending $(\mathsf{vk}, \mathsf{vk}_A, \mathsf{vk}_E, \mathsf{sk}_E)$ to $\mathcal{B}$and $\mathcal{B}$forwards these keys to $\mathcal{A}$. Whenever $\mathcal{A}$makes a query to oracles $\mathsf{OPreSignInit}$, $\mathsf{OPreSignInteraction}$, $\mathsf{OAdapt}$, $\mathsf{OSign}$ or $\mathsf{ONewStatement}$, $\mathcal{B}$forwards this query to the respective oracles of the challenger. Oracles $\mathsf{OPreSignInteraction}$ and $\mathsf{ONewStatement}$ are identical in weak unforgeability (2P-AISO-AS) and in full unforgeability (2P-SO-AS). Oracles $\mathsf{OAdapt}$ and $\mathsf{OSign}$ are almost identical, with the only difference being that in weak unforgeability (2P-AISO-AS), these oracles store additional information in the sets $\mathcal{X}$ (compromised statements) and $\mathcal{Q}$ (messages on which a signature has been revealed). Thus, these oracles are a faithful computation of each other. Finally, oracle $\mathsf{OPreSignInit}$ is different in both weak unforgeability (2P-AISO-AS) and full unforgeability (2P-SO-AS). By the definition of $H$, we know that if $\overline{k}$ is even, then $H.\mathsf{PreSign}_1(\mathsf{sk}_A, w, m, x) := \Theta.\mathsf{PreSign}_1(\mathsf{sk}_A, m, x)$, otherwise if $\overline{k}$ is odd, then $H.\mathsf{PreSign}_2(\mathsf{sk}_A, w, m, x, pm_1) = \Theta.\mathsf{PreSign}_2(_A, m, x, pm_1)$. Thus the oracle $\mathsf{OPreSignInit}$ from full unforgeability (2P-SO-AS) runs a faithful computation of the expected oracle from weak unforgeability (2P-AISO-AS). Finally, $\mathcal{A}$ outputs a forgery $(m^*, \sigma^*)$ and $\mathcal{B}$ outputs this same forgery. The set of forgeries that satisfy $c_1$ and $c_2$ in weak unforgeability (2P-AISO-AS) are a subset of the forgeries that satisfy $c_1$ and $c_2$ in full unforgeability (2P-SO-AS). Therefore if $\mathcal{A}$ wins with non-negligible probability, then $\mathcal{B}$ wins with non-negligible probability.

## K Security Of Two-Party Adaptor Signatures From Identification Schemes

In previous works, Kiltz et al. [38] presented a construction for digital signatures from identification schemes. Later, Erwig et al. [16] proposed two extensions: one achieves two-party signatures from identification schemes, and the other one achieves adaptor signatures from identification schemes. When combining both extensions, Erwig et al. showed that one can achieve two-party adaptor signatures from identification schemes.

When presenting these constructions, they also proposed security notions congruent with the available notions of single-party adaptor signatures [1]. These notions security notions were *two-party existential unforgeability* and *two-party witness extractability*, and the constructions of two-party adaptor signatures from identification schemes was proven secure under them. However, these security notions inherited the

same weaknesses discussed in [21, 14]. Moreover, in the context of two-party adaptor signatures, we discussed the relevance of the unfair output delivery of the presign protocol (Section 2). In the construction of two-party adaptor signatures from identification schemes, it is assumed that the presign protocol guarantees the output delivery to both participants. In this paper we proposed definitions for $2P-AISO-AS$ and $2P-SO-AS$ that both include the insights from [21, 14] and explicitly do not assume fair output delivery on the presignature generation. In the following, we will prove that the construction of two-party adaptor signatures from identification schemes is also a secure 2P-SO-AS.

### K.1 2P-SO-AS From Identification Schemes

We first reformulate the construction of two-party adaptor signatures from identification schemes [16] as a 2P-SO-AS. For reference, we include in Figure 33 the construction of single-party signature schemes from identification schemes, in Figure 35 the construction of two-party signature scheme from identification schemes and in Figure 34 the construction of two-party adaptor signatures from identification schemes. Both constructions of two-party signatures and adaptor signatures rely on the deterministic functions $\mathcal{F}_{\mathsf{com\text{-}rand}}$, $\mathcal{F}_{\mathsf{com\text{-}sig}}$, $\mathcal{F}_{\mathsf{com\text{-}vk}}$ which combine shares of announcements, signatures and verification keys respectively. In particular, $\mathcal{F}_{\mathsf{com\text{-}vk}}$ must satisfy:

$$\Pr\left[\mathsf{Vrfy}(\mathsf{vk},m,\sigma) \;\middle|\; \begin{array}{r}(\mathsf{vk}_0,\mathsf{sk}_0)\leftarrow\mathsf{IGen}(n);\ (\mathsf{vk}_1,\mathsf{sk}_1)\leftarrow\mathsf{IGen}(n)\\ \mathsf{vk}:=\mathcal{F}_{\mathsf{com\text{-}vk}}(\mathsf{vk}_0,\mathsf{vk}_1)\\ \sigma\leftarrow\Pi\mathsf{Sign}\langle\mathsf{sk}_0,\mathsf{sk}_1\rangle(\mathsf{vk},m)\end{array}\right]=1 \tag{1}$$

Two-party adaptor signatures also rely on the deterministic functions $\mathcal{F}_{\mathsf{shift}}$, $\mathcal{F}_{\mathsf{adapt}}$, $\mathcal{F}_{\mathsf{ext}}$ and $\mathcal{F}_{\mathsf{dec\text{-}sig}}$ which must satisfy the following:

$$\mathcal{F}_{\mathsf{shift}}(\mathsf{V}_0(\mathsf{vk},h,\tilde{s}),x)=\mathsf{V}_0(\mathsf{vk},h,\mathcal{F}_{\mathsf{adapt}}(\tilde{s},w)). \tag{2}$$

$$w=\mathcal{F}_{\mathsf{ext}}(\mathcal{F}_{\mathsf{adapt}}(\tilde{s},w),\tilde{s}). \tag{3}$$

$$\Pr\left[\mathsf{Vrfy}(\mathsf{vk}_{1-i},m,\sigma_{1-i}) \;\middle|\; \begin{array}{r}(\mathsf{vk}_0,\mathsf{sk}_0)\leftarrow\mathsf{IGen}(n);\ (\mathsf{vk}_1,\mathsf{sk}_1)\leftarrow\mathsf{IGen}(n)\\ \mathsf{vk}:=\mathcal{F}_{\mathsf{com\text{-}vk}}(\mathsf{vk}_0,\mathsf{vk}_1)\\ \sigma\leftarrow\Pi\mathsf{Sign}\langle\mathsf{sk}_0,\mathsf{sk}_1\rangle(\mathsf{vk},m)\\ \sigma_{1-i}\leftarrow\mathcal{F}_{\mathsf{dec\text{-}sig}}(\mathsf{sk}_i,\mathsf{vk}_i,\sigma)\end{array}\right]=1 \tag{4}$$

Moreover, in the proofs we require

In Figure 34 we present the presign protocol by unrolling two sub-protocols used in their construction, namely $\Pi\mathsf{Rand\text{-}Exc}$ and $\Pi\mathsf{Exchange}$. Indeed we observe that, without an additional trust assumption, the $\Pi\mathsf{Exchange}$ cannot be realized fairly due to the impossibility result of [18]. This implies that one of the two participants of the presign protocol ($P_0$ and $P_1$) obtains the output of the protocol (the presignature) before the other party. This observation aligns with the discussion from Section 2 where we argue that, in applications, one of the parties (the *Extractor*) is required to obtain the output of the presign protocol before the other party.

The protocol $\Pi\mathsf{Rand\text{-}Exc}$ is a coin-flipping protocol between two parties $P_0$ and $P_1$. Roughly, $P_0$ samples a random value, then computes and sends an (extractable) commitment of this value to $P_1$. Next $P_1$ samples another random value and sends it to $P_0$. Finally, $P_0$ discloses their random value by sending the opening of the commitment to $P_1$. The hiding of the commitment ensures that both parties can sample some joint randomness in such a way that no party can choose their randomness contingent to the randomness of other party. The $\Pi\mathsf{Rand\text{-}Exc}$ is used in both two-party signatures (Figure 35) and in two-party adaptor signatures (Figure 34). In both cases, we fix that $P_0$ (resp. $P_1$) during the sign and presign protocols is also $P_0$ (resp. $P_1$) in the underlying $\Pi\mathsf{Rand\text{-}Exc}$ protocol. We want to remark that this merely a matter of notation. The parties could equivalently swap their roles in the $\Pi\mathsf{Rand\text{-}Exc}$ without affecting the result, but we chose to

| $\mathsf{KeyGen}(n)$ | $\mathsf{Sign}(\mathsf{sk}, m)$ | $\mathsf{Vrfy}(\mathsf{vk}, m, \sigma)$ |
|---|---|---|
| $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{IGen}(n)$ | $(R, St) \leftarrow \mathsf{P}_1(\mathsf{sk})$ | $(h, s) := \sigma$ |
| 1 : **return** $(\mathsf{sk}, \mathsf{vk})$ | 1 : $h := \mathsf{H}(R, m)$ | 1 : $R := \mathsf{V}_0(\mathsf{vk}, h, s)$ |
| | 2 : $s \leftarrow \mathsf{P}_2(\mathsf{sk}, R, h, St)$ | 2 : **return** $h = \mathsf{H}(R, m)$ |
| | 3 : **return** $(h, s)$ | |

Fig. 33: Digital signature scheme from identification schemes [38]

fix this assignment for the sake of simplicity. The security of the constructions would also not be affected but fixing this assignment determines where certain proof arguments are invoked.

In Figure 36 we rewrite the presign protocol from Figure 34 so that it complies with the definition of 2P-SO-AS. In a 2P-SO-AS, the presign protocol is a series of steps $\mathsf{PreSign}_1$, $\mathsf{PreSign}_2$, $\mathsf{PreSign}_3$, $\mathsf{PreSign}_4$ and $\mathsf{PreSign}_5$. The parties that participate in the presign protocol, $E$ (*Extractor*) and $A$ (*Adaptor*) take alternating turns in running these steps. Each step outputs a state and a protocol message. The final step outputs the presignature. Notice that the presign protocol in Figure 36 ultimately performs the same operations and in the same order as the presign protocol in Figure 34. Relative to Figure 34, in Figure 36 the *Extractor* was $P_0$ and the *Adaptor* was $P_1$. Algorithms $\mathsf{PreSign}_1$, $\mathsf{PreSign}_2$ and $\mathsf{PreSign}_3$ perform the corresponding steps of $\Pi\mathsf{Rand\text{-}Exc}$.

**Lemma 26 (Figure 36 is correct).** *If the identification scheme is complete and if the equations 2 and 3 hold, then the 2P-SO-AS of Figure 36 is correct.*

*Proof.* The proof follows the same steps of the proof for lemma 1 in [16].

We fix an arbitrary message $m$ and a pair of statement and witness $(x, w) \in \mathsf{R}$. Let $(\mathsf{vk}, (\mathsf{vk}_E, \mathsf{sk}_E), (\mathsf{vk}_A, \mathsf{sk}_A)) \leftarrow \mathsf{KeyGen}(n)$. Let $\hat{\sigma} \leftarrow \Pi\mathsf{PreSign}\langle \mathsf{sk}_E, \mathsf{sk}_A \rangle(m, x)$. Let $\sigma \leftarrow \mathsf{Adapt}(\mathsf{vk}, \hat{\sigma}, w, st_4)$ and let $w' \leftarrow \mathsf{Extract}(\mathsf{vk}, \sigma, \hat{\sigma}, x)$.

From Figure 36 we know that $\hat{\sigma} := (h, \tilde{s})$, $\sigma := (h, s)$ and $w' := \mathcal{F}_{\mathsf{ext}}(s, \tilde{s})$ where we have $s := \mathcal{F}_{\mathsf{adapt}}(\tilde{s}, w)$, $\tilde{s} := \mathcal{F}_{\mathsf{com\text{-}sig}}(h, \mathsf{P}_2(\mathsf{sk}_A, R_A, h, St_A), \mathsf{P}_2(\mathsf{sk}_E, R_E, h, St_E))$, $h := \mathsf{H}(R_{sign}, m)$, $R_{sign} := \mathcal{F}_{\mathsf{shift}}(R_{pre}, x)$, $R_{pre} := \mathcal{F}_{\mathsf{com\text{-}rand}}(R_A, R_E)$, $(R_A, St_A) \leftarrow \mathsf{P}_1(\mathsf{sk}_A)$ and $(R_E, St_E) \leftarrow \mathsf{P}_1(\mathsf{sk}_E)$.

From completeness of the identification scheme we get:

$$\mathsf{H}(\mathcal{F}_{\mathsf{shift}}(\mathsf{V}_0(\mathsf{vk}, h, \tilde{s}), x), m) = \mathsf{H}(\mathcal{F}_{\mathsf{shift}}(R_{pre}, x), m) = \mathsf{H}(R_{sign}, m) = h.$$

And therefore $\mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma}) = 1$.

From equation 2 and from the previous equation we observe that

$$\mathsf{H}(\mathsf{V}_0(\mathsf{vk}, h, s), m) = \mathsf{H}(\mathsf{V}_0(\mathsf{vk}, h, \mathcal{F}_{\mathsf{adapt}}(\tilde{s}, w)), m) = \mathsf{H}(\mathcal{F}_{\mathsf{shift}}(\mathsf{V}_0(\mathsf{vk}, h, \tilde{s}), x), m) = h.$$

And therefore $\mathsf{Vrfy}(\mathsf{vk}, m, \sigma) = 1$

Finally, from equation 3 we get:

$$w' = \mathcal{F}_{\mathsf{ext}}(s, \tilde{s}) = \mathcal{F}_{\mathsf{ext}}(\mathcal{F}_{\mathsf{adapt}}(\tilde{s}, w), \tilde{s}) = w.$$

And therefore $(x, w') \in \mathsf{R}$.

### K.2 Security analysis

**Lemma 27 (Figure 36 satisfies adaptability).** *If the equation 2 holds, then the 2P-SO-AS of Figure 36 satisfies adaptability.*

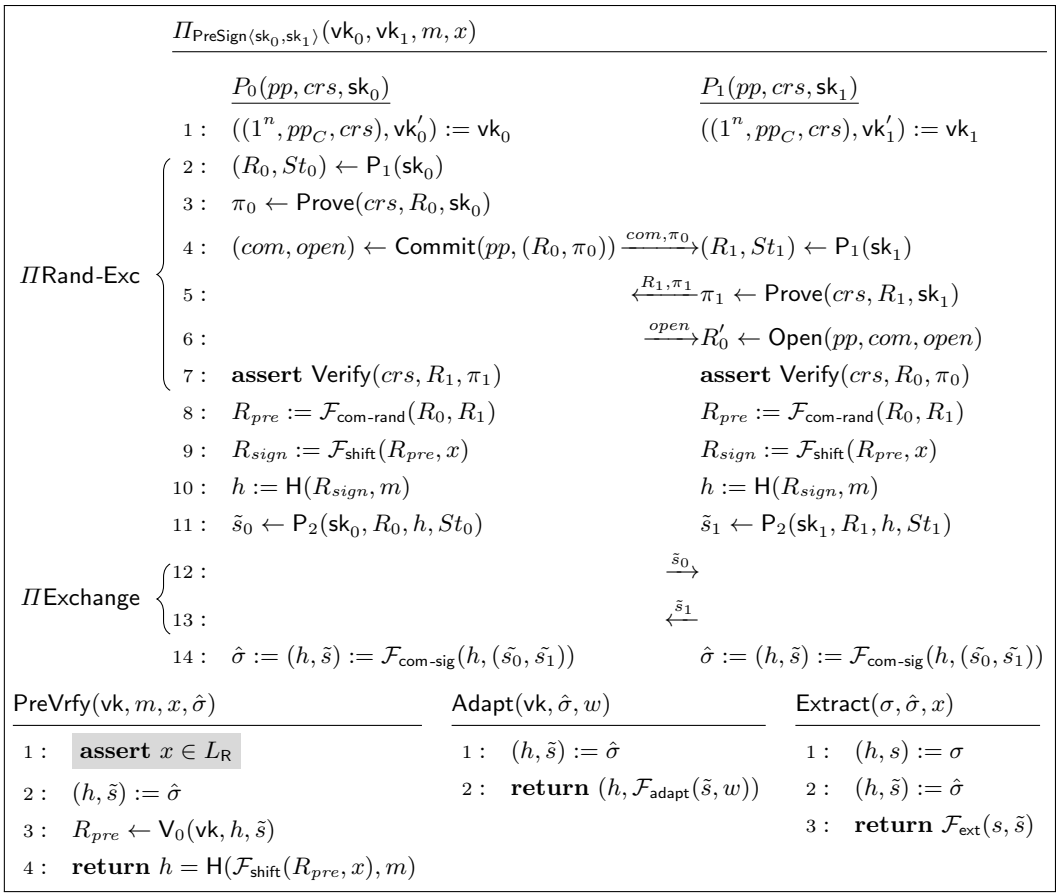



Fig. 34: Presign protocol between two parties as defined in the construction of adaptor signatures from identification schemes by Erwig et al. [16]. Sub-protocols $\Pi\mathsf{Rand\text{-}Exc}$ and $\Pi\mathsf{Exchange}$ are unrolled.

*Proof.* The proof follows the same steps of the proof for lemma 3 in [16].

Assume that $\mathsf{PreVrfy}(\mathsf{vk},m,x,\hat{\sigma}) = 1$. This implies that $h = \mathsf{H}(\mathcal{F}_{\mathsf{shift}}(\mathsf{V}_0(\mathsf{vk},h,\tilde{s}),x),m)$ where $(h,\tilde{s}) := \hat{\sigma}$. Then we use equation 2 and plug it in the previous equation. As a result we get:

$$h = \mathsf{H}(\mathsf{V}_0(\mathsf{vk},h,\mathcal{F}_{\mathsf{adapt}}(\tilde{s},w)),m).$$

This implies that $\mathsf{Vrfy}(\mathsf{vk},m,\sigma) = 1$ where $\sigma = (h,\mathcal{F}_{\mathsf{adapt}}(\tilde{s},w)) = \mathsf{Adapt}(\mathsf{vk},\hat{\sigma},w,st_4)$ for any $st_4$.

**Lemma 28 (Figure 36 satisfies full unforgeability).** *If the* NIZK *is zero-knowledge and simulation-extractable, the commitment scheme is extractable, the signature scheme from Figure 33 is SUF-CMA and the hard relation satisfies one-more-hardness, then the construction from Figure 36 satisfies full unforgeability.*

*Proof.* The proof follows in a series of hybrids, which we describe below. Let $\mathcal{H}_i \approx \mathcal{H}_{i+1}$ denote $|\Pr[\mathcal{H}_i = 1] - \Pr[\mathcal{H}_{i+1} = 1]| \leq \mathsf{negl}(n)$.

**Hybrid** $\mathcal{H}_0$: This hybrid corresponds to the original full unforgeability experiment.

**Hybrid** $\mathcal{H}_1$: This hybrid we want to show that the adversary did not create a forgery by adapting with a witness they did not chose. This hybrid is identical to $\mathcal{H}_0$ with the exception that, during the presign

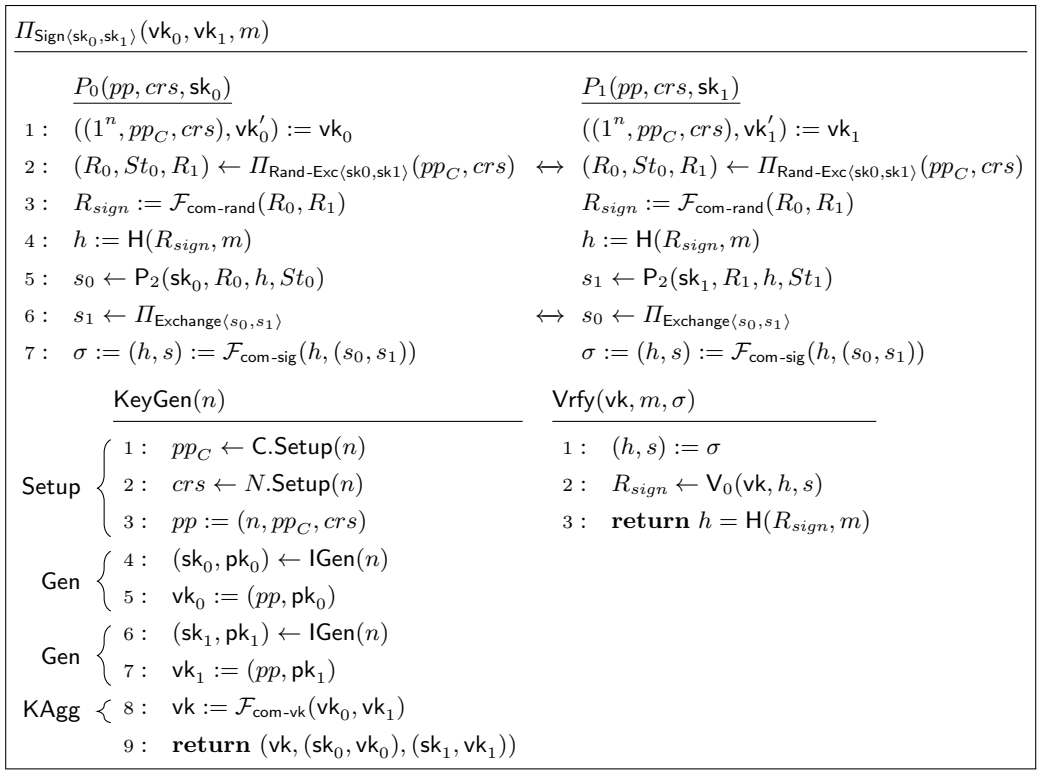



Fig. 35: Two-party signature scheme with aggregatable keys. Algorithm KeyGen in [16] is defined as a protocol between parties $P_0$ and $P_1$ running the algorithms Setup to generate public parameters, Gen to generate the key shares for each party and KAgg to aggregate the shares of the verification keys. For the sake of simplicity as pointed out in Section 4, we include all these operations inside KeyGen. For reference, in this figure, we mark which parts of KeyGen correspond to Setup, Gen and KAgg. In the subprotocols $\Pi_{\mathsf{Rand\text{-}Exc}}$ and $\Pi_{\mathsf{Exchange}}$ inside of the signing protocol are described in Figure 34.

protocol, for each joint randomness $R_{sign}$ we store the transcript of the protocol in a table $\mathcal{T}$. In particular we store the randomness $R_E$ of the adversary, the secret randomness $St_E$ of $\mathcal{H}_1$, the zero-knowledge proof that $\mathcal{A}$ provides, the statement, and the partial presignature of $\mathcal{H}_1$ $\mathcal{T}[R_{sign}] = (R_E, St_A, \pi_E, x, \tilde{s_A})$. At the end of the game, when $\mathcal{A}$ outputs a forgery $\sigma^* = (h^*, s^*)$, we obtain its randomness $R^* := \mathsf{V}_0(\mathsf{vk}, h^*, s^*)$. From this randomness, we check whether there is an entry in $\mathcal{T}$. If there is, it means that the adversary ran the presign protocol on the forgery message $m^*$. Thus we then obtain the transcript $(R_E, St_A, \pi_E, x, \tilde{s_A}) := \mathcal{T}[R^*]$. Now we want to see if the adversary obtained the forgery by adapting the presignature generated in that invocation of the presign protocol. To check this, $\mathcal{H}_1$ first runs the extract algorithm of the NIZK to extract $(\mathsf{sk}_E, St_E) \leftarrow \mathsf{Ext}^{\mathcal{A}}(crs, x, \pi)$. With this, $\mathcal{H}_1$ has all the ingredients to reconstruct the presignature by first running $\tilde{s_E} \leftarrow \mathsf{P}_2(\mathsf{sk}_E, R_E, h^*, St_E)$ and finally computing $\tilde{s} := \mathcal{F}_{\mathsf{com\text{-}sig}}(\tilde{s_E}, \tilde{s_A})$. With this presignature $\hat{\sigma}^* := (h^*, \tilde{s})$ and the forgery signature $\sigma^*$, we try to extract $w' := \mathsf{Extract}(\mathsf{vk}, \hat{\sigma}^*, \sigma^*, x)$. If it holds that $(x, w') \in \mathsf{R}$, then $\mathcal{H}_1$ triggers $\mathsf{Bad}_1$ and aborts.

*Claim.* If the NIZK is simulation-extractable and if the relation R satisfies one-more-hardness, then $\mathcal{H}_0 \approx \mathcal{H}_1$.

*Proof.* First, we remark that the proof $\pi_E$ that $\mathcal{A}$ sends during the presign algorithm must satisfy $\mathsf{Verify}(crs, R_E, \pi_E) = 1$, as otherwise the protocol aborts on the assertion of line 4 in $\mathsf{PreSign}_4$. If the NIZK is simulation-extractable, then the extractor algorithm Ext exists, it is efficient and since the proof $\pi_E$ verifies, then $(\mathsf{sk}_E, St_E) \leftarrow \mathsf{Ext}^{\mathcal{A}}(crs, x, \pi)$ extracts the correct $(\mathsf{sk}_E, St_E)$ corresponding to $R_E$ with non-negligible probability.

With this, we will do a reduction to the one-more of the hard relation. Assume that there exists a PPT adversary $\mathcal{A}$ that is able to trigger $\mathsf{Bad}_1$ with non-negligible probability. Then, we can construct a PPT adversary $\mathcal{B}$ that is able to break the one-more-hardness of the relation R. The challenger of the one-more-hardness begins by creating $q$ statements $x_i$ (for $i \in [q]$) and sends all of them to $\mathcal{B}$. On their side, $\mathcal{B}$ plays the $\mathcal{H}_1$ game with $\mathcal{A}$ honestly. Assume that $q$ is the number of queries to ONewStatement that $\mathcal{A}$ is allowed to make. Whenever $\mathcal{A}$ makes the $i$'th query to the oracle ONewStatement, $\mathcal{B}$ simply replies with the $i$'th statement $x_i$. Whenever $\mathcal{A}$ makes a query $(\hat{\sigma}, x, m, sid)$ to OAdapt, then $\mathcal{B}$ queries $(x)$ to the oracle OWit from the one-more-hardness challenger. Notice that, by the assertion on line 1 of OAdapt, $\mathcal{A}$ can only make queries to this oracle on statements generated by ONewStatement, and therefore these statements are valid queries to OWit. The oracle OWit responds to $\mathcal{B}$ with the witness $w$ that corresponds to $x$. With this, $\mathcal{B}$ computes the rest of the oracle OAdapt normally.

At the end of the game, $\mathcal{A}$ presents a forgery $(m^*, \sigma^*)$, from which $\mathcal{H}_1$ extract a witness $w'$. If $\mathcal{A}$ is able to trigger $\mathsf{Bad}_1$ with non-negligible probability, it means that: (i) by condition $c_1$, $\mathcal{A}$ never queried OAdapt on $m^*$ (line 3 of OAdapt), (ii) by $c_1$, more subtlety, $\mathcal{A}$ never queried for a presignature on $m^*$ and on a statement $x$ that either was not generated by ONewStatement or it was leaked by OAdapt. These two facts imply that $w'$ was never a response from OWit. If $\mathcal{A}$ triggers $\mathsf{Bad}_1$ with non-negligible probability, then $\mathcal{B}$ queries OWit to obtain the witnesses other than $w'$ and returns all these witnesses, breaking the one-more-hardness of the relation R.

**Hybrid** $\mathcal{H}_2$: In this hybrid, we replace the honest $crs$ generation with a simulated one $(crs, \tau) \leftarrow \mathsf{SetupSim}(n)$. Then we modify the sign and presign protocols by replacing the zero-knowledge proof $\pi_A \leftarrow \mathsf{Prove}(crs, R_A, \mathsf{sk}_A)$ with a simulated proof $\pi_A \leftarrow \mathsf{ProveSim}(crs, \tau, R_A)$.

*Claim.* If the NIZK is zero-knowledge, then $\mathcal{H}_1 \approx \mathcal{H}_2$.

*Proof.* Assume that there exists a PPT adversary $\mathcal{A}$that distinguishes between $\mathcal{H}_1$ and $\mathcal{H}_2$. Such adversary satisfies $|\Pr[\mathcal{H}_1^{\mathcal{A}}(n) = 1] - \Pr[\mathcal{H}_2^{\mathcal{A}}(n) = 1]| > \frac{1}{2} + \mathsf{negl}(n)$. Then we can create a PPT adversary $\mathcal{B}$ that breaks zero-knowledge of the NIZK. The challenger of the zero-knowledge has $\mathcal{B}$ as the adversary and, internally, $\mathcal{B}$ plays $\mathcal{H}_1$ with $\mathcal{A}$. First, the zero-knowledge challenger flips a bit $b \leftarrow_{\$} \{0, 1\}$. If $b = 0$, then the zero-knowledge challenger runs $crs \leftarrow \mathsf{Setup}(n)$. Otherwise, the zero-knowledge challenger runs $(crs, \tau) \leftarrow \mathsf{SetupSim}(n)$. In both cases, the challenger sends $crs$ to $\mathcal{B}$. Then, $\mathcal{B}$ plays $\mathcal{H}_1$ honestly with $\mathcal{A}$ with one exception: Whenever $\mathcal{A}$ queries for a signature or a presignature, instead of $\mathcal{B}$ computing the zero-knowledge proof by themself, $\mathcal{B}$ queries $(x := R_A; w := (\mathsf{sk}_A, St_A))$ to the oracle of the zero-knowledge challenger for the proof. If $b = 0$ the oracle of the zero-knowledge challenger would compute $\pi \leftarrow \mathsf{Prove}(crs, w, x)$. Otherwise, the oracle would compute $\pi \leftarrow \mathsf{ProveSim}(crs, \tau, x)$. In both cases, the oracle sends $\pi$ to $\mathcal{B}$. For the rest of $\mathcal{H}_1$ $\mathcal{B}$ plays honestly with $\mathcal{A}$.

Notice that if $b = 0$, then $\mathcal{B}$ would be playing exactly $\mathcal{H}_1$ with $\mathcal{A}$. Otherwise, if $b = 1$, then $\mathcal{B}$ would be playing exactly $\mathcal{H}_2$ with $\mathcal{A}$.

By the end of the game, $\mathcal{A}$ outputs a forgery. By assumption we know that $|\Pr[\mathcal{H}_1^{\mathcal{A}}(n) = 1] - \Pr[\mathcal{H}_2^{\mathcal{A}}(n) = 1]| > \frac{1}{2} + \mathsf{negl}(n)$. Assume without loss of generality that $\Pr[\mathcal{H}_2^{\mathcal{A}}(n) = 1] > \Pr[\mathcal{H}_1^{\mathcal{A}}(n) = 1] + \mathsf{negl}(n)$. If $\mathcal{A}$ outputs a winning forgery, then $\mathcal{B}$ outputs $\hat{b} = 1$, otherwise $\mathcal{B}$ outputs $\hat{b} = 0$. The probability that $\mathcal{B}$ wins the zero-knowledge game is equal to the probability that $\mathcal{A}$ distinguishes $\mathcal{H}_1$ and $\mathcal{H}_2$. In other words, the probability that $\mathcal{B}$ wins is $|\Pr[\mathcal{H}_1^{\mathcal{A}}(n) = 1] - \Pr[\mathcal{H}_2^{\mathcal{A}}(n) = 1]| > \frac{1}{2} + \mathsf{negl}(n)$.

**Hybrid** $\mathcal{H}_3$: In this hybrid, we change the signing and presigning oracles OSign, OPreSignInit and OPreSignInteraction. Upon queries to these oracles, $\mathcal{H}_3$ starts interacting honestly in the respective protocol. Since $\mathcal{A}$ plays the role of the *Extractor*, $\mathcal{A}$ provides a commitment and a zero-knowledge proof $(com, \pi_E)$ as the first message of both protocols. After receiving the first message from $\mathcal{A}$, $\mathcal{H}_3$ uses the trapdoor of the extractable commitment to extract the randomness that $\mathcal{A}$ committed to $R'_E \leftarrow \mathsf{Extract}(pp_C, tr, com)$. Then $\mathcal{H}_3$ interacts honestly with $\mathcal{A}$ during the respective protocol until $\mathcal{A}$ sends the opening to the commitment $open$. At than point, $\mathcal{H}_3$ opens the commitment $R_E \leftarrow \mathsf{Open}(pp_C, com, open)$. Then $\mathcal{H}_3$ triggers $\mathsf{Bad}_3$ and aborts if $R'_E \neq R_E$.

*Claim.* If the commitment scheme is extractable, then $\mathcal{H}_2 \approx \mathcal{H}_3$.

*Proof.* Note that $\mathcal{H}_3$ aborts only if the extracted value $R'_E$ from commitment *com* is not equal to the actual committed value $R_E$ in *com*. In other words, $\mathcal{H}_3$ aborts if $\mathsf{Extract}(p, td, com) \neq \mathsf{Open}(p, com, open)$. By the extractability property of the commitment scheme, this happens only with negligible probability, therefore $\Pr[\mathsf{Bad}_3] \leq \mathsf{negl}(n)$ and thus $\mathcal{H}_2 \approx \mathcal{H}_3$.

**Hybrid** $\mathcal{H}_4$: In this hybrid, we further modify the signing oracle $\mathsf{OSign}$. Upon queries to $\mathsf{OSign}$ oracles, $\mathcal{H}_4$ starts interacting honestly in the respective protocol. After $\mathcal{A}$ sends the first message $(com, \pi_E)$, $\mathcal{H}_4$ extracts the randomness committed to by $\mathcal{A}$ by running $R'_E \leftarrow \mathsf{Extract}(pp_C, tr, com)$. Then, $\mathcal{H}_4$ computes a single-party signature of the *Adaptor* by running $(h^*, s^*) \leftarrow \mathsf{Sign}(\mathsf{sk}_A, m)$. This signature is not a signature w.r.t. the joint randomness, but w.r.t. some individual randomness $R^*$ sampled during the single-party signing algorithm. Then $\mathcal{H}_4$ will use $R^*$ as their own randomness in the signing protocol $R_A := R^*$. Notice that the proof is simulated, therefore $\mathcal{H}_4$ does not need to know the witness for $R^*$ in the zero-knowledge proof. At this point in the sub-protocol $\Pi\mathsf{Rand\text{-}Exc}$, $\mathcal{H}_4$ has both the randomness of the adversary $R_E$ and their own $R_A := R^*$, but crucially, the adversary does not have $R_E$ yet. Then, $\mathcal{H}_4$ computes the joint randomness $R_{sign} := \mathcal{F}_{\mathsf{com\text{-}rand}}(R_A, R_E)$ and then checks whether $(R_{sign}, m)$ has been queried to the random oracle before. $\mathcal{H}_4$ programs the random oracle by setting $\mathsf{H}[R_{sign}, m] := h^*$. $\mathcal{H}_4$ also programs the random oracle by setting $\mathsf{H}[R^*, m] := r$ where $r$ is a fresh uniformly random number. If $(R_{sign}, m)$ or $(R^*, m)$ have been queried to the random oracle before, $\mathcal{H}_4$ triggers the event $\mathsf{Bad}_4$ and aborts. After this, $\mathcal{H}_4$ continues honestly interacting with *adv* in the signing protocol. This programming of the random oracle makes the signature $(h^*, s^*)$ obtained at the beginning be a valid partial signature. At the end of the signing protocol, $\mathcal{H}_4$ provides $(h^*, s^*)$ as their own partial signature $s_A$.

*Claim.* $\mathcal{H}_3 \approx \mathcal{H}_4$.

*Proof.* Assume that there exists a $\mathsf{PPT}$ adversary $\mathcal{A}$ that triggers $\mathsf{Bad}_4$. First note that $\mathsf{Bad}_4$ triggers only if $\mathcal{A}$ queried $(R_{sign}, m)$ or $(R^*, m)$ to the random oracle $\mathsf{H}$. The output of $\mathsf{P}_1$, $R_{pre}$, is uniformly random from a super-polynomial set of size $q$ in the security parameter. From this, it follows that $R_{sign}$ and $R^*$ are distributed uniformly at random in the same set. Since $\mathcal{A}$ is $\mathsf{PPT}$, $\mathcal{A}$ can only make polynomially many queries to $\mathsf{H}$ and the oracles $\mathsf{OPreSignInit}$ and $\mathsf{OPreSignInteraction}$. Let $l$ be the total number of queries to $\mathsf{H}$, $\mathsf{OPreSignInit}$ and $\mathsf{OPreSignInteraction}$. We have that $\Pr[\mathsf{Bad}_4] = \Pr[\mathsf{H}[R_{sign}, m] \neq \perp] + \Pr[\mathsf{H}[R^*, m] \neq \perp] \leq \frac{2l}{q} \leq \mathsf{negl}(n)$. This follows from the fact that $l$ is polynomial in the security parameter.

**Hybrid** $\mathcal{H}_5$: In this hybrid, we do a change similar to what we did in $\mathcal{H}_4$ but this time for the presign protocol. Whenever the adversary queries $\mathsf{OPreSignInit}$, $\mathcal{H}_5$ first gets the commitment *com* from the protocol message $pm_1$ that is part of the query. Then $\mathcal{H}_5$ extracts from the commitment the randomness of $\mathcal{A}$ by running $R'_E \leftarrow \mathsf{Extract}(pp_C, tr, com)$. Next, $\mathcal{H}_5$ computes a single-party signature of the *Adaptor* by running $(h^*, s^*) \leftarrow \mathsf{Sign}(\mathsf{sk}_A, m)$. This signature is not a signature w.r.t. the joint randomness, but w.r.t. some individual randomness $R^*$ sampled during the single-party signing algorithm. After this, $\mathcal{H}_5$ learns the randomness used to compute $(h^*, s^*)$ by running $R^* := \mathsf{V}_0(\mathsf{vk}_E, h^*, s^*)$. Then $\mathcal{H}_5$ will use $R^*$ as their own randomness in the presigning protocol $R_E := R^*$. Notice that the proof is simulated, therefore $\mathcal{H}_5$ does not need to know the witness for $R^*$ in the zero-knowledge proof. At this point, $\mathcal{H}_5$ has both the randomness of the adversary $R_E$ and their own $R_A := R^*$, but crucially, the adversary does not have $R_A$ yet. Then $\mathcal{H}_5$ computes the joint randomness $R_{sign} := \mathcal{F}_{\mathsf{shift}}(\mathcal{F}_{\mathsf{com\text{-}rand}}(R_A, R_E), x)$ and then checks whether $(R_{sign}, m)$ has been queried to the random oracle before. If $(R_{sign}, m)$ has been queried to the random oracle before, $\mathcal{H}_5$ triggers the event $\mathsf{Bad}_5$ and aborts. Otherwise, $\mathcal{H}_5$ programs the random oracle by setting $\mathsf{H}[R_{sign}, m] := h^*$. $\mathcal{H}_5$ also programs the random oracle by setting $\mathsf{H}[R^*, m] := r$ where $r$ is a fresh uniformly random number. After this, $\mathcal{H}_5$ continues honestly interacting with *adv* in the signing protocol. This programming of the random oracle makes the signature $(h^*, s^*)$ obtained at the beginning be a valid partial presignature. At the end of the presigning protocol, $\mathcal{H}_5$ provides $(h^*, s^*)$ as their own partial presignature $\tilde{s_A}$.

*Claim.* $\mathcal{H}_4 \approx \mathcal{H}_5$

*Proof.* The proof is identical to that of $\mathcal{H}_4$

**Reduction to SUF-CMA of Figure 33**

*Claim.* If the single-party digital signatures from Figure 33 is SUF-CMA, and if equation Equation (4) holds, then for any PPT adversary $\mathcal{A}$, it holds that $\Pr[\mathcal{H}_5(\mathcal{A}) = 1] \leq \mathsf{negl}(n)$.

*Proof.* Assume there exists a PPT adversary $\mathcal{A}$ such that $\Pr[\mathcal{H}_5(\mathcal{A}) = 1] > \mathsf{negl}(n)$, then we can create a PPT adversary $\mathcal{B}$ that breaks SUF-CMA of the signature scheme from Figure 33. The challenger will play the SUF-CMA game with $\mathcal{B}$ as the adversary, while $\mathcal{B}$ will play $\mathcal{H}_5$ with $\mathcal{A}$ as the adversary. In this reduction, the SUF-CMA challenger begins by creating a pair of keys $(\mathsf{vk}_A, \mathsf{sk}_A) \leftarrow \mathsf{KeyGen}(n)$ and sending $\mathsf{vk}_A$ to $\mathcal{B}$. On their side, $\mathcal{B}$ runs the KeyGen algorithm from Figure 35, except that $\mathcal{B}$ does not generate the pair of keys in line 4 and instead uses the $\mathsf{vk}_A$ provided by the SUF-CMA challenger. Then $\mathcal{B}$ sends the keys $(\mathsf{vk}, \mathsf{vk}_A, \mathsf{vk}_E, \mathsf{sk}_E)$ to $\mathcal{A}$ and $\mathcal{A}$ can start making queries. Queries to oracle ONewStatement and OAdapt are responded honestly by $\mathcal{B}$. On queries to oracle OSign, $\mathcal{B}$ first needs to generate a signature. Instead of $\mathcal{B}$ computing this signature themself (which they cannot since $\mathcal{B}$ lacks $\mathsf{sk}_A$), $\mathcal{B}$ queries the signing oracle from the SUF-CMA challenger. After generating the single-party signature of the *Extractor*, $\mathcal{B}$ runs the rest of the oracle as in $\mathcal{H}_4$. On queries to oracle OPreSignInit, $\mathcal{B}$ first needs to generate a signature. Again, $\mathcal{B}$ queries the signing oracle from the SUF-CMA challenger to generate this signature. Then $\mathcal{B}$ runs the rest of the oracles OPreSignInit and OPreSignInteraction as in $\mathcal{H}_5$. By the end of $\mathcal{H}_5$, $\mathcal{A}$ returns with a forgery message and signature $(m^*, (h^*, s^*))$. $\mathcal{B}$ then computes $s^*_A := \mathcal{F}_{\mathsf{dec\text{-}sig}}(\mathsf{sk}_E, \mathsf{vk}_E, (h^*, s^*))$ and outputs $(m^*, \sigma^* := (h^*, s^*_A))$ to the SUF-CMA challenger as their forgery.

If $\mathcal{A}$ wins $\mathcal{H}_5$ with non-negligible probability, we have that $\mathsf{Vrfy}(\mathsf{vk}, m^*, (h^*, s^*)) = 1$. By the equation Equation (4) we thus have that $\mathsf{Vrfy}(\mathsf{vk}_A, m^*, \sigma^*) = 1$. If $\mathcal{A}$ wins $\mathcal{H}_5$, then $m^* \notin \mathcal{Q}$, which means that $\mathcal{A}$ did not query the oracle OSign with the forgery message, therefore $\mathcal{B}$ also did not query the SUF-CMA signing oracle with the forgery message. The only possibility of the forgery not being valid is that during a query to OPreSignInit, when $\mathcal{H}_5$ makes another internal query to the SUF-CMA signing oracle, this oracle responds with $\sigma^* := (h^*, s^*)$. This would cause $(m^*, \sigma^*)$ to be added to the set of pairs signature and message revealed by the signing oracle. However, due to the programming on the random oracle in $\mathcal{H}_5$, from the point of view of $\mathcal{A}$ $(h^*, s^*)$ is a valid presignature and not a valid signature. Therefore $\mathsf{Vrfy}(\mathsf{vk}, m^*, (h^*, s^*)) = 1$ would not hold in $\mathcal{H}_5$ and therefore $\mathcal{A}$ could not win by presenting that forgery.

**Lemma 29 (Figure 36 satisfies full extractability).** *If the* NIZK *is zero-knowledge, the commitment scheme hiding and the signature scheme from Figure 33 is SUF-CMA, then the construction from Figure 36 satisfies full extractability.*

*Proof.* The proof follows in a series of hybrids, which we describe below. Let $\mathcal{H}_i \approx \mathcal{H}_{i+1}$ denote $|\Pr[\mathcal{H}_i = 1] - \Pr[\mathcal{H}_{i+1} = 1]| \leq \mathsf{negl}(n)$.

**Hybrid** $\mathcal{H}_0$: This hybrid corresponds to the original full extractability experiment.

**Hybrid** $\mathcal{H}_1$: In this hybrid, we replace the honest $crs$ generation with a simulated one $(crs, \tau) \leftarrow \mathsf{SetupSim}(n)$. Then we modify the sign and presign protocols by replacing the zero-knowledge proof $\pi_E \leftarrow \mathsf{Prove}(crs, R_E, \mathsf{sk}_E)$ with a simulated proof $\pi_E \leftarrow \mathsf{ProveSim}(crs, \tau, R_E)$.

*Claim.* If the NIZK is zero-knowledge, then $\mathcal{H}_0 \approx \mathcal{H}_1$.

*Proof.* Assume that there exists a PPT adversary $\mathcal{A}$ such that $|\Pr[\mathcal{H}_0(\mathcal{A}) = 1] - \Pr[\mathcal{H}_1(\mathcal{A}) = 1]| > \frac{1}{2} + \mathsf{negl}(n)$. If such $\mathcal{A}$ exists, then we can find a PPT adversary $\mathcal{B}$ that breaks zero-knowledge of the NIZK. The challenger of the zero-knowledge first makes a coin flip an gets a bit $b$. If $b = 0$, the challenger computes $crs \leftarrow \mathsf{Setup}(n)$, otherwise, the challenger computes $(crs, \tau) \leftarrow \mathsf{SetupSim}(n)$. In both cases, the challenger gives $crs$ to $\mathcal{B}$. On their side, $\mathcal{B}$ acts as the challenger of full extractability and interacts with $\mathcal{A}$. First $\mathcal{B}$ runs the KeyGen algorithm from Figure 35 except that instead of generating the $crs$ on line 2, $\mathcal{B}$ uses the $crs$ generated by the challenger. When $\mathcal{A}$ queries OSign or OPreSignInit, $\mathcal{B}$ runs the respective protocols honestly up to the point where $\mathcal{B}$ has to compute the zero-knowledge proof. Instead of generating the proof themself, they will query the challenger of the zero-knowledge to generate the proof on the pair of statement and witness $(R_E, \mathsf{sk}_E)$.

If $b = 0$, the zero-knowledge challenger computes the honest proof, otherwise, the challenger computes a simulated proof. By the end of the full extractability game, if $\mathcal{A}$ wins, $\mathcal{B}$ outputs $\tilde{b} = 1$.

Notice that, besides the step of computing the zero-knowledge proof, $\mathcal{H}_1$ is a faithful computation of $\mathcal{H}_0$. Thus, $\mathcal{B}$ breaks the zero-knowledge with the same probability that $\mathcal{A}$ serves as a distinguisher between $\mathcal{H}_0$ and $\mathcal{H}_1$. If $|\Pr[\mathcal{H}_0(\mathcal{A}) = 1] - \Pr[\mathcal{H}_1(\mathcal{A}) = 1]| > \frac{1}{2} + \mathsf{negl}(n)$, then the probability of $\mathcal{B}$ distinguishing in the zero-knowledge experiment is also bigger than $\frac{1}{2} + \mathsf{negl}(n)$.

**Hybrid** $\mathcal{H}_2$: In this hybrid, whenever the adversary queries the signing oracle $\mathsf{OSign}$, $\mathcal{H}_2$ first create a fresh single-party signature of the *Extractor* by running $(h^*, s^*) := \mathsf{Sign}(\mathsf{sk}_E, m)$. After this, $\mathcal{H}_2$ learns the randomness used to compute $(h^*, s^*)$ by running $R^* := \mathsf{V}_0(\mathsf{vk}_E, h^*, s^*)$. This signature is not a signature w.r.t. the joint randomness, but w.r.t. some individual randomness $R^*$ sampled during the single-party signing algorithm. Then $\mathcal{H}_2$ will use $R^*$ as their own randomness in the signing protocol $R_E := R^*$. Notice that the proof is simulated, therefore $\mathcal{H}_2$ does not need to know the witness for $R^*$ in the zero-knowledge proof. Then, $\mathcal{H}_2$ interacts with $\mathcal{A}$ in the two-party signing oracle up to the point where $\mathcal{A}$ sends their randomness $R_A$. At this point in the sub-protocol $\Pi\mathsf{Rand\text{-}Exc}$, $\mathcal{H}_2$ has both the randomness of the adversary $R_A$ and their own $R_E := R^*$, but crucially, the adversary does not have $R_E$ yet, only a commitment to it. Then, $\mathcal{H}_2$ computes the joint randomness $R_{sign} := \mathcal{F}_{\mathsf{com\text{-}rand}}(R_A, R_E)$ and then checks whether $(R_{sign}, m)$ has been queried to the random oracle before. If $(R_{sign}, m)$ has been queried to the random oracle before, $\mathcal{H}_2$ triggers the event $\mathsf{Bad}_2$ and aborts. Otherwise, $\mathcal{H}_2$ programs the random oracle by setting $\mathsf{H}[R_{sign}, m] := h^*$. $\mathcal{H}_2$ also programs the random oracle by setting $\mathsf{H}[R^*, m] := r$ where $r$ is a fresh uniformly random number. After this, $\mathcal{H}_2$ continues honestly interacting with *adv* in the signing protocol. This programming of the random oracle makes the signature $(h^*, s^*)$ obtained at the beginning be a valid partial signature. At the end of the signing protocol, $\mathcal{A}$ must provide their own partial signature $s_A$. Then $\mathcal{H}_2$ uses $s_E := (h^*, s^*)$ as their partial signature and computes the joint signature as $\sigma := \mathcal{F}_{\mathsf{com\text{-}sig}}(h^*, (s_A, s_E))$.

*Claim.* If the commitment scheme is hiding, then $\mathcal{H}_1 \approx \mathcal{H}_2$.

*Proof.* we need to prove that the probability of $\mathsf{Bad}_2$ triggering is negligible. In order to do so, we need to consider what is the probability of $\mathcal{A}$ querying $(R_{sign}, m)$ to the random oracle, where $R_{sign} := \mathcal{F}_{\mathsf{com\text{-}rand}}(R_A, R_E)$. By this point, our challenger has not revealed $R_E$ to the adversary yet. Therefore, the probability of $\mathcal{A}$ guessing $R_{sign}$ by this point is bounded by the probability of randomly guessing $R_E$ plus the probability of using the commitment *com* to extract $R_E$.

The probability of randomly guessing $R_E$ is clearly negligible. However assume that the probability of a $\mathsf{PPT}$ adversary $\mathcal{A}$ triggering $\mathsf{Bad}_2$ was non-negligible. If $\mathcal{A}$ was able to trigger $\mathsf{Bad}_2$ with non-negligible probability, then we could create a $\mathsf{PPT}$ adversaty $\mathcal{B}$ that breaks the hiding property of the commitment scheme. In this reduction, $\mathcal{B}$ plays $\mathcal{H}_2$ honestly with $\mathcal{A}$. Whenever $\mathcal{B}$ needs to commit to $R_E$ during the signing protocol, $\mathcal{B}$ queries the challenger of the hiding of the commitment scheme for a commitment on $R_E$. If the bit $b$ of the challenger is 1, the reply of this oracle will be a commitment to $R_E$, otherwise it will be a commitment to 0. After this, $\mathcal{B}$ continues in $\mathcal{H}_2$ honestly. If $\mathcal{A}$ is able to trigger $\mathsf{Bad}_2$, then $\mathcal{B}$ outputs $\tilde{b} := 1$. The probability that $\mathcal{B}$ wins the hiding game of the commitment scheme is the same that $\mathcal{A}$ triggers this $\mathsf{Bad}_2$, which is non-negligible. Thus, $\mathcal{B}$ would break the hiding of the commitment scheme.

**Hybrid** $\mathcal{H}_3$: In this hybrid, we do a change similar to what we did in $\mathcal{H}_2$ but this time for the presign protocol. Whenever the adversary queries $\mathsf{OPreSignInit}$, $\mathcal{H}_3$ first creates a fresh single-party signature of the *Extractor* by running $(h^*, s^*) := \mathsf{Sign}(\mathsf{sk}_E, m)$. This signature is not a signature w.r.t. the joint randomness, but w.r.t. some individual randomness $R^*$ sampled during the single-party signing algorithm. After this, $\mathcal{H}_3$ learns the randomness used to compute $(h^*, s^*)$ by running $R^* := \mathsf{V}_0(\mathsf{vk}_E, h^*, s^*)$. Then $\mathcal{H}_3$ will use $R^*$ as their own randomness in the presigning protocol $R_E := R^*$. Notice that the proof is simulated, therefore $\mathcal{H}_3$ does not need to know the witness for $R^*$ in the zero-knowledge proof. Then, $\mathcal{H}_3$ interacts with $\mathcal{A}$ in the two-party presign protocol up to the point where $\mathcal{A}$ sends their randomness $R_A$ (that is, during $\mathsf{PreSign}_3$). At this point, $\mathcal{H}_3$ has both the randomness of the adversary $R_A$ and their own $R_E := R^*$, but crucially, the adversary does not have $R_E$ yet, only a commitment to it. Then $\mathcal{H}_3$ computes the joint randomness $R_{sign} := \mathcal{F}_{\mathsf{shift}}(\mathcal{F}_{\mathsf{com\text{-}rand}}(R_A, R_E), x)$ and then checks whether $(R_{sign}, m)$ has been queried to the

random oracle before. If $(R_{sign}, m)$ has been queried to the random oracle before, $\mathcal{H}_3$ triggers the event $\mathsf{Bad}_3$ and aborts. Otherwise, $\mathcal{H}_3$ programs the random oracle by setting $\mathsf{H}[R_{sign}, m] := h^*$. $\mathcal{H}_3$ also programs the random oracle by setting $\mathsf{H}[R^*, m] := r$ where $r$ is a fresh uniformly random number. After this, $\mathcal{H}_3$ continues honestly interacting with $adv$ in the signing protocol. This programming of the random oracle makes the signature $(h^*, s^*)$ obtained at the beginning be a valid partial presignature. At the end of the presigning protocol, $\mathcal{A}$ must provide their own partial presignature $\tilde{s_A}$. Then $\mathcal{H}_3$ uses $\tilde{s_E} := (h^*, s^*)$ as their partial signature and computes the joint presignature as $\hat{\sigma} := \mathcal{F}_{\mathsf{com\text{-}sig}}(h^*, (\tilde{s_A}, \tilde{s_E}))$.

*Claim.* If the commitment scheme is hiding, then $\mathcal{H}_2 \approx \mathcal{H}_3$.

*Proof.* The proof is the same as the proof for $\mathcal{H}_2$ only that the replacement happens during the presign protocol.

**Reduction to SUF-CMA of Figure 33**:

*Claim.* If the single-party digital signatures from Figure 33 is SUF-CMA, then for any PPT adversary $\mathcal{A}$, it holds that $\Pr[\mathcal{H}_3(\mathcal{A}) = 1] \leq \mathsf{negl}(n)$.

*Proof.* Assume there exists a PPT adversary $\mathcal{A}$ such that $\Pr[\mathcal{H}_3(\mathcal{A}) = 1] > \mathsf{negl}(n)$, then we can create a PPT adversary $\mathcal{B}$ that breaks SUF-CMA of the signature scheme from Figure 33. The challenger will play the SUF-CMA game with $\mathcal{B}$ as the adversary, while $\mathcal{B}$ will play $\mathcal{H}_3$ with $\mathcal{A}$ as the adversary. In this reduction, the SUF-CMA challenger begins by creating a pair of keys $(\mathsf{vk}_E, \mathsf{sk}_E) \leftarrow \mathsf{KeyGen}(n)$ and sending $\mathsf{vk}_E$ to $\mathcal{B}$. On their side, $\mathcal{B}$ runs the KeyGen algorithm from Figure 35, except that $\mathcal{B}$ does not generate the pair of keys in line 4 and instead uses the $\mathsf{vk}_E$ provided by the SUF-CMA challenger. Then $\mathcal{B}$ sends the keys $(\mathsf{vk}, \mathsf{vk}_E, \mathsf{vk}_A, \mathsf{sk}_A)$ to $\mathcal{A}$ and $\mathcal{A}$ can start making queries. Queries to oracle ONewStatement are responded honestly by $\mathcal{B}$. On queries to oracle OSign, $\mathcal{B}$ first needs to generate a signature. Instead of $\mathcal{B}$ computing this signature themself, $\mathcal{B}$ queries the signing oracle from the SUF-CMA challenger. After generating such signature, $\mathcal{B}$ runs the rest of the oracle as in $\mathcal{H}_2$. On queries to oracle OPreSignInit, $\mathcal{B}$ first needs to generate a signature. Again, $\mathcal{B}$ queries the signing oracle from the SUF-CMA challenger to generate this signature. Then $\mathcal{B}$ runs the rest of the oracles OPreSignInit and OPreSignInteraction as in $\mathcal{H}_3$. By the end of $\mathcal{H}_3$, $\mathcal{A}$ returns with a forgery message and signature $(m^*, (h^*, s^*))$. $\mathcal{B}$ then computes $s^*_E := \mathcal{F}_{\mathsf{dec\text{-}sig}}(\mathsf{sk}_A, \mathsf{vk}_A, (h^*, s^*))$ and outputs $(m^*, \sigma^* := (h^*, s^*_E))$ to the SUF-CMA challenger as their forgery.

If $\mathcal{A}$ wins $\mathcal{H}_3$ with non-negligible probability, we have that $\mathsf{Vrfy}(\mathsf{vk}, m^*, (h^*, s^*)) = 1$. By the equation Equation (4) we thus have that $\mathsf{Vrfy}(\mathsf{vk}_E, m^*, \sigma^*) = 1$. If $\mathcal{A}$ wins $\mathcal{H}_3$, then $m^* \notin \mathcal{Q}$, which means that $\mathcal{A}$ did not query the oracle OSign with the forgery message, therefore $\mathcal{B}$ also did not query the SUF-CMA signing oracle with the forgery message. The only possibility of the forgery not being valid is that during a query to OPreSignInit, when $\mathcal{H}_3$ makes another internal query to the SUF-CMA signing oracle, this oracle responds with $\sigma^* := (h^*, s^*)$. This would cause $(m^*, \sigma^*)$ to be added to the set of pairs signature and message revealed by the signing oracle. However, due to the programming on the random oracle in $\mathcal{H}_3$, from the point of view of $\mathcal{A}$ $(h^*, s^*)$ is a valid presignature and not a valid signature. Therefore $\mathsf{Vrfy}(\mathsf{vk}, m^*, (h^*, s^*)) = 1$ would not hold in $\mathcal{H}_3$ and therefore $\mathcal{A}$ could not win by presenting that forgery.

**Lemma 30 (Figure 36 satisfies unlinkability).** *If for any signing key* sk *these two distributions are computationally indistinguishable,* $\{R | (R, St) \leftarrow \mathsf{P}_1(\mathsf{sk})\} \approx \{\mathcal{F}_{\mathsf{shift}}(R, x) | R \leftarrow D_R, (x, w) \leftarrow \mathsf{GenR}(n)\}$ *and if equation 2 holds, then the construction from Figure 36 satisfies unlinkability.*

*Proof.* The proof follows in a series of hybrids, which we describe below. Let $\mathcal{H}_i \approx \mathcal{H}_{i+1}$ denote $|\Pr[\mathcal{H}_i = 1] - \Pr[\mathcal{H}_{i+1} = 1]| \leq \mathsf{negl}(n)$.

**Hybrid** $\mathcal{H}_0$: This hybrid corresponds to the original unlinkability experiment.

**Hybrid** $\mathcal{H}_1$: In this hybrid, when the adversary queries the oracle OChall, we modify the way that a signature is computed on line 4. During the sign algorithm, right after the challenger computes $R_{sign} := \mathcal{F}_{\mathsf{com\text{-}rand}}(R_0, R_1)$, we sample a new instance of the hard relation $(x, w) \leftarrow \mathsf{GenR}(n)$ and compute $R'_{sign} := \mathcal{F}_{\mathsf{shift}}(R_{sign}, x)$. For the rest of the protocol, we replace every instance of $R_{sign}$ with $R'_{sign}$.

*Claim.* If for any signing key $\mathsf{sk}$ these two distributions are computationally indistinguishable, $\{R|(R, St) \leftarrow \mathsf{P}_1(\mathsf{sk})\} \approx \{\mathcal{F}_{\mathsf{shift}}(R, x)|R \leftarrow D_R, (x, w) \leftarrow \mathsf{GenR}(n)\}$, then $\mathcal{H}_0 \approx \mathcal{H}_1$

*Proof.* The identification scheme announcement $R_{sign}$ becomes $R'_{sign} = \mathcal{F}_{\mathsf{shift}}(R_{sign}, x)$. From our hypotheses we have that $R_{sign}$ and $R'_{sign}$ are computationally indistinguishable. Then the challenge $h$ becomes $h' = \mathsf{H}(R'_{sign}, m)$. Since $\mathsf{H}$ is a random oracle, $h$ and $h'$ are computationally indistinguishable. Finally, each response $s_0$ and $s_1$ becomes $s'_0 = \mathsf{P}_2(\mathsf{sk}_0, R'_{sign}, h', St)$ and $s'_1 = \mathsf{P}_2(\mathsf{sk}_1, R'_{sign}, h', St)$ respectively. Given that $\mathsf{P}_2$ is efficient, we have that $s_0$ and $s'_0$ are computationally indistinguishable, as well as $s_1$ and $s'_1$. Therefore, an adversary can only distinguish between $\mathcal{H}_0$ and $\mathcal{H}_1$ with at most negligible probability.

**Hybrid** $\mathcal{H}_2$: In this hybrid, when the adversary queries the oracle $\mathsf{OChall}$, we modify the way that a signature is computed on line 4. During the sign algorithm, instead of computing the responses $s_0$ and $s_1$ as $s_0 \leftarrow \mathsf{P}_2(\mathsf{sk}_0, R'_{sign}, h', St)$ and $s_1 \leftarrow \mathsf{P}_2(\mathsf{sk}_1, R'_{sign}, h', St)$, we compute $s'_0 \leftarrow \mathsf{P}_2(\mathsf{sk}_0, R_{sign}, h', St)$ and $s'_1 \leftarrow \mathsf{P}_2(\mathsf{sk}_1, R_{sign}, h', St)$ respectively, then compute $s_c := \mathcal{F}_{\mathsf{com\text{-}sig}}(s'_0, s'_1)$ and finally compute $s' := \mathcal{F}_{\mathsf{adapt}}(s_c, w)$ and output $(h', s')$ as the response. Notice that in $s'_0$ and $s'_1$ we are now using the original announcement $R_{sign}$ and not the shifter one $R'_{sign}$.

*Claim.* If equation 2 holds, then $\mathcal{H}_1 \approx \mathcal{H}_2$.

*Proof.* By the commitment recoverability of the identification scheme we know that $R_{sign} = \mathsf{V}_0(\mathsf{vk}, h, s_c)$. Using equation 2 we have that $\mathcal{F}_{\mathsf{shift}}(\mathsf{V}_0(\mathsf{vk}, h', s_c), x) = \mathsf{V}_0(\mathsf{vk}, h', \mathcal{F}_{\mathsf{adapt}}(s_c, w))$. In $\mathcal{H}_1$ we defined $R'_{sign} := \mathcal{F}_{\mathsf{shift}}(R_{sign}, x)$ and $h' := \mathsf{H}(R'_{sign}, m)$ therefore $\mathcal{F}_{\mathsf{shift}}(R_{sign}, x) = \mathsf{V}_0(\mathsf{vk}, h', s')$. Therefore $\mathcal{H}_2$ produces the same distribution of signatures as $\mathcal{H}_1$. Therefore, an adversary can only distinguish between $\mathcal{H}_1$ and $\mathcal{H}_2$ with at most negligible probability.

In $\mathcal{H}_2$ we arrive at a hybrid in which both branches $b = 0$ and $b = 1$ in oracle $\mathsf{OChall}$ are identical, therefore an adversary is only able to win with the same probability of distinguishing $\mathcal{H}_2$ and $\mathcal{H}_0$, which is at most negligible probability.

**Lemma 31 (Figure 36 satisfies preverify-soundness).** *The construction from Figure 36 satisfies preverify-soundness.*

*Proof.* The proof follows from the definition of $\mathsf{PreVrfy}$ in Figure 34. On line 1, we assert that $x \in L_{\mathsf{R}}$. If this does not hold, the algorithm rejects the presignature. Therefore for any $x \notin L_{\mathsf{R}}$ we have that $\mathsf{PreVrfy}(\mathsf{vk}, m, x, \hat{\sigma}) = 0$.

# L On One-Way Presignatures

In this section we present a faulty construction that motivates the notion of one-way presignatures and then prove that this notion is implied by *adaptability* and *weak unforgeability*

## L.1 Faulty Construction

In Figure 37 we present a the construction that motivates one-way presignatures. In this construction, the *Adaptor* discloses the witness $w$ as part of the protocol message $pm$ that is output of $\mathsf{PreSign}_A$. This implies that the *Extractor* learns the $w$ before the presignature is generated and before the presignature is adapted into a valid signature. This construction is therefore not suitable for many applications like zero-knowledge contingent payment, where disclosing the witness before adapting the signature implies that the buyer can obtain the product without paying by means of the adapted signature.

This construction should therefore be considered completely insecure, however, it satisfies all security notions except for one-way presignatures, which captures precisely the fact that the presignature should not leak the witness.

**Lemma 32.** *If BLS signatures are unforgeable and unique and if* $\mathsf{R}$ *is q-hard, then the construction in Figure 10 satisfies full extractability, weak unforgeability, designated adaptability, unlinkability, computational preverify soundness* but not *one-way signatures in the Random Oracle Model.*

*Proof.* The proof of full extractability follows the same steps as the proof of Lemma 7. The proof of weak unforgeability follows the same steps as the proof of Lemma 8. The proof of designated adaptability follows the same steps as the proof of Lemma 9. The proof of unlinkability follows the same steps as the proof of Lemma 10. The proof of computational preverify soundness follows the same steps as the proof of Lemma 11.

The construction of Figure 37 does not have one-way presignatures. Therefore there exists a $\mathsf{PPT}$ adversary $\mathcal{A}$ that wins this game with non-negligible probability. This adversary wins by simply querying for a new statement $x \leftarrow \mathsf{ONewStatement}()$, samples a random message $m$ and queries for the protocol message $pm \leftarrow \mathsf{OPreSign}_A(m, x)$. Since the protocol message contains the witness $w^* := pm$, $\mathcal{A}$ outputs $w^*$.

$\mathcal{A}$ wins the one-way presignature game with the same probability that $(x, w^*) \in \mathsf{R}$ holds. By the definitino of Figure 37 this condition holds with probability 1, and therefore $\mathcal{A}$ wins with non-negligible probability.

**Lemma 33 (Adaptability and Weak Unforgeability Imply One-Way Presignatures).** *If a 2P-AISO-AS scheme is correct and satisfies* adaptability *and* weak unforgeability *then it also satisfies* one-way presignatures.

*Proof.* Assume that a 2P-AISO-AS scheme does not satisfy *one-way presignatures*, then there must exist a $\mathsf{PPT}$ adversary $\mathcal{A}$ that wins the one-way presignatures game. If such adversary exists, then we can create another $\mathsf{PPT}$ adversary $\mathcal{B}$ that using $\mathcal{A}$ and *adaptability* breaks *weak unforgeability*.

The weak unforgeability challenger plays against $\mathcal{B}$ and $\mathcal{B}$ internally plays the one-way presignatures game with $\mathcal{A}$. At the beginning of the reduction, $\mathcal{B}$ receives keys $(\mathsf{vk}, \mathsf{vk}_A, \mathsf{vk}_E, \mathsf{sk}_E)$, and $\mathcal{B}$ forwards these keys to $\mathcal{A}$. Whenever $\mathcal{A}$ queries $\mathsf{ONewStatement}$ or $\mathsf{OPreSign}_A$, $\mathcal{B}$ forwards this query to the challenger and uses the response of the oracle of the challenger to respond to the query of $\mathcal{A}$. By the end of the one-way presignatures game, $\mathcal{A}$ sends $w^*$ such that, for some statement $x^*$ responded by $\mathsf{ONewStatement}$, it holds that $(x^*, w^*) \in \mathsf{R}$. With this, $\mathcal{B}$ simply samples a random message $m^*$ sends the query $(m^*, x^*)$ to $\mathsf{OPreSign}_A$. The oracle responds with a protocol message $pm$, on which $\mathcal{B}$ runs $\hat{\sigma} \leftarrow \mathsf{PreSign}_E(\mathsf{sk}_E, m^*, x^*, pm)$. By the correctness of the 2P-AISO-AS scheme, the presignature $\hat{\sigma}$ satisfies $\mathsf{PreVrfy}(\mathsf{vk}, \hat{\sigma}, m^*, x^*) = 1$. Then $\mathcal{B}$ uses $w^*$ to compute $\sigma^* \leftarrow \mathsf{Adapt}(\mathsf{vk}, \hat{\sigma}, w, \bot)$. By adaptability, we know that $\sigma^*$ satisfies $\mathsf{Vrfy}(\mathsf{vk}, m^*, \sigma^*) = 1$ and thus $\mathcal{B}$ sets $(m^*, \sigma^*)$ as the output.

The probability that $\mathcal{B}$ wins the weak unforgeability game is the same that $\mathcal{A}$ wins the one-way presignature game.

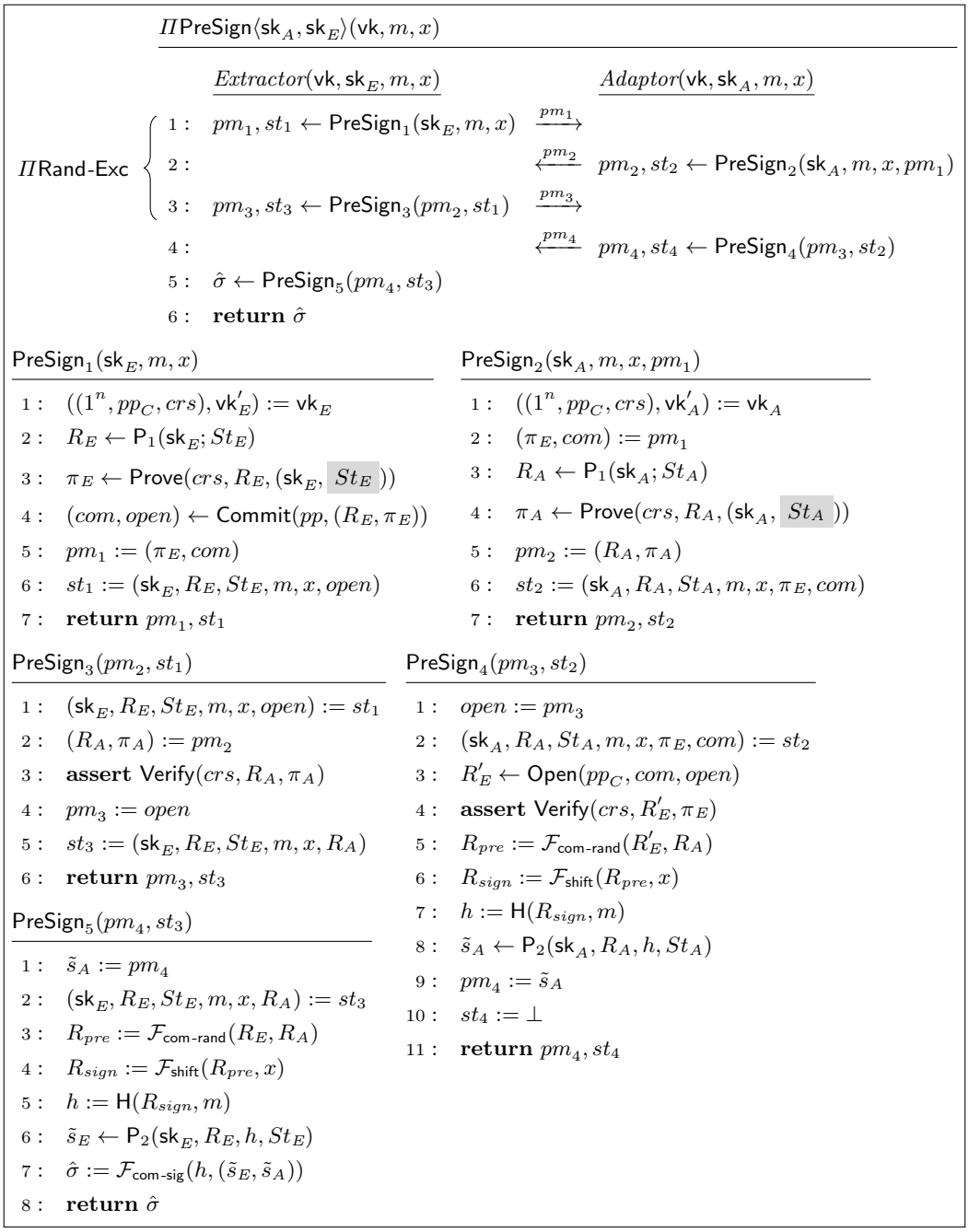



Fig. 36: 2P-AISO-AS construction based on 2P-AS from identification schemes construction of [16].

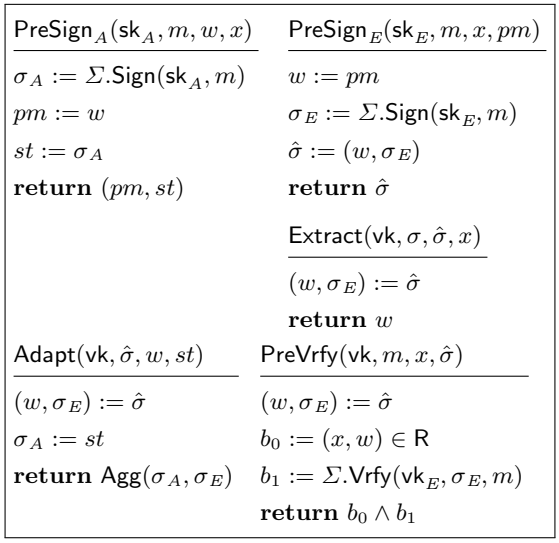



Fig. 37: Faulty construction of 2P-AISO-AS for BLS signatures. We denote by $\Sigma$ the BLS signature scheme. $\mathsf{Agg}$ denotes the BLS aggregation function of two signatures. To ease notation, we assume that after executing $(\mathsf{vk}, (\mathsf{sk}_A, \mathsf{vk}_A), (\mathsf{sk}_E, \mathsf{vk}_E)) \leftarrow \mathsf{KeyGen}(1^n)$, both $\mathsf{vk}_A$ and $\mathsf{vk}_E$ are implicit parameters.