Universally Composable Adaptor Signatures

Paul Gerhart[1], Daniel Rausch[2], and Dominique Schröder[1]

[1] TU Wien, Austria
[2] University of Stuttgart, Germany

Abstract. Adaptor signatures extend the functionality of digital signatures by enabling the computation of *pre-signatures* on messages relative to statements in NP relations. Pre-signatures are publicly verifiable objects that simultaneously hide and commit to a standard signature on the same message. Anyone possessing a valid witness for the statement can adapt the pre-signature into a full signature under the underlying signature scheme. Since adaptor signatures are commonly used as building blocks in larger systems—such as blockchain protocols—it is natural to seek a security definition within the Universal Composability (UC) framework. A recent attempt by Tairi et al. (CCS'23) introduced the first UC functionality for adaptor signatures.

This paper makes both negative and positive contributions. On the negative side, we show that the functionality proposed by Tairi et al. suffers from critical limitations:

- The functionality fails to guarantee *extractability* and *adaptability*—the core security properties of adaptor signatures—to higher-level protocols.
- No adaptor signature scheme can realize the functionality.

On the positive side, we propose a new UC functionality that faithfully captures the latest security guarantees of adaptor signatures as formalized via game-based notions by Gerhart et al. (EUROCRYPT'24).

- Our functionality guarantees extractability, unique extractability, and pre-signature adaptability in a way that is composable and meaningful for higher-level protocols.
- We show that it is realizable by an enhanced Schnorr-based adaptor signature scheme that we construct. Our construction maintains compatibility with existing infrastructure and is efficient enough for practical deployment, particularly in Bitcoin-like environments.

1 Introduction

Smart contracts have become a foundational component of decentralized finance (DeFi), enabling automated and trustless interactions between mutually distrusting parties. By delegating enforcement to the blockchain, they circumvent the classical impossibility of two-party fairness [18]. This functionality has fueled widespread adoption across DeFi protocols, NFT platforms, and token swaps. As of 2025, over $118 billion is locked in smart contracts [1], with more than 77 million contracts deployed on Ethereum alone [2]. Despite this success, smart contracts face well-known limitations [55]: execution is costly–especially on Ethereum, where gas fees for even simple operations can be prohibitive [50]–and their reliance on expressive scripting languages makes them incompatible with blockchains like Bitcoin, which deliberately restrict on-chain programmability [44].

Adaptor Signatures as Middle Ground. To address these challenges, cryptographers introduced *adaptor signatures* [3, 29]—a primitive that enables fair exchange on blockchains without the need for complex on-chain scripting. Adaptor signatures extend standard digital signatures by enabling the computation of *pre-signatures* for NP statements. A pre-signature $\widetilde{\sigma}$ is a publicly verifiable object that commits to a valid signature σ on a message m and a statement Y, while hiding the signature. Anyone possessing the corresponding witness y for Y can transform $\widetilde{\sigma}$ into the valid signature σ. Most importantly, each party possessing both the pre-signature $\widetilde{\sigma}$ and the adapted signature σ can extract a valid witness for Y. They hence provide a fair exchange of a witness for a valid signature. Adaptor signatures enable a constrained class of smart contract functionality—most notably, conditional payments and atomic swaps [3, 24]. These contracts are limited in expressiveness, as they only support fairness conditions tied to signature release. However, even within this constrained model, adaptor signatures have proven to be highly versatile. Adaptor signatures have become foundational in off-chain cryptographic protocols, initially proposed for payment channels [22, 23, 42, 43],

and now underpinning diverse applications such as privacy-preserving coin mixing [30,45], oracle-based payments [41], fair digital signature exchange [31,54], and puzzle-solving incentives [55]. A major driver of their adoption is compatibility with existing signature standards. In particular, Bitcoin's Taproot upgrade [56] enables native deployment of Schnorr-based adaptor signatures, making these schemes highly practical today.

Security of Adaptor Signatures. Since adaptor signatures need to guarantee multiple security properties to protect mutually distrusting parties with seemingly contradicting security goals, formalizing security for adaptor signatures is non-trivial. The first formal model for adaptor signature security was introduced by Aumayr et al. [3]. Subsequent work by Dai et al. [20, 29] identified shortcomings in this model. Gerhart et al. [29] identified more shortcomings and extended the framework to contexts beyond payment channels. In more detail, adaptor signatures involve two parties: the signer, who generates pre-signatures bound to statements from hard relations, and the adaptor, who uses pre-signatures and witnesses for the statements to finalize and publicly release signatures. To ensure fairness between the signer and the adaptor, an adaptor signature scheme needs to ensure two core properties: The signer must learn a valid witness on the statement when the adaptor releases a valid signature (extractability). The adaptor must be able to compute a valid signature when it knows a valid pre-signature and a valid witness for the statement with respect to which the pre-signature pre-verifies (adaptability).

Universally Composable Security. Although game-based definitions accurately describe specific security threats, they only guarantee security in isolation and do not account for interactions with other cryptographic primitives, which is a rare occurrence in practice, especially for adaptor signature schemes, which are mainly considered in blockchain contexts. Consequently, universally composable (UC) security (e.g., [11, 15, 40]) definitions would be highly desirable. Such UC notions provide security within arbitrary contexts, support modular security analyses, and allow for reusing security results without redoing proofs. An immediate practical benefit for protocol designers is that UC security gets rid of labor-intensive and error-prone manual reductions that would otherwise have to be performed for each new protocol based on a cryptographic primitive. UC security can instead be proven once for the primitive in an isolated setting by showing that the primitive realizes a suitable ideal functionality. A general UC composition theorem then immediately implies, without needing any further proofs, that security is retained for any future UC protocol built on top of the primitive. In this sense, investigating UC security can also be seen as answering several major research questions: "Is there a generic reduction to a primitive that can be applied in a blackbox manner for arbitrary higher-level protocols? Are established game-based security notions sufficient for this reduction? Or are there settings and protocols which cannot be reduced such that further properties need to be required?" For these reasons, it is standard practice to translate and study cryptographic primitives - even basic ones with well-established game-based security notions - in the UC framework (e.g., [4, 12, 14, 16, 27, 28, 32, 36, 38, 39]).

UC Secure Adaptor Signatures. Tairi et al. [53] took the first step toward this goal by formalizing an ideal functionality for adaptor signatures, which we refer to as "$\mathcal{F}_{[53]}$." This first functionality is an important advance; however, similar challenges that plagued game-based approaches exist in the UC setting, and modeling a functionality for adaptor signatures turns out to be highly non-trivial. In particular, we demonstrate in Section 3.2 that no adaptor signature scheme can realize $\mathcal{F}_{[53]}$. Based on this result—and on more problems—we show that Schnorr adaptor signatures cannot realize $\mathcal{F}_{[53]}$. Furthermore, we show that the functionality does not provide extractability nor adaptability to higher-level protocols.

This situation is highly unsatisfactory. Either we rely on game-based definitions and hope they offer sufficient robustness for practical composability, or we rely on an ideal functionality that does not formalize the required game-based notions nor can be realized by any adaptor signature scheme. This leads us to the following questions:

- *How can adaptor signatures be properly formalized in the UC setting?*
- *Given the practical importance of Schnorr-based schemes, can we provide UC guarantees for them?*

We answer both questions affirmatively. More specifically:

1.1 Our Contributions

In this paper, we explore how adaptor signature security can be meaningfully captured within the Universal Composability (UC) framework. We present both negative results—revealing structural obstacles in existing formulations—and positive results, culminating in a new, realizable ideal functionality for adaptor signatures.

Negative Results. We demonstrate several fundamental limitations of the functionality $\mathcal{F}_{[53]}$ proposed by Tairi et al. [53]:

- We prove that no adaptor signature scheme can realize $\mathcal{F}_{[53]}$.
- Even if a realizing scheme existed, we show that $\mathcal{F}_{[53]}$ fails to formalize *extractability* and *adaptability*—the defining security property of adaptor signatures—thereby undermining compositional guarantees for protocols built on top.

Positive Results. To overcome these limitations, we develop a new ideal functionality for adaptor signatures that aligns with modern game-based security definitions and is realizable in practice:

- We propose a redesigned ideal functionality that faithfully captures the core guarantees of adaptor signatures while avoiding the pitfalls of prior formulations.
- We prove that, if an adaptor signature scheme achieves all core security guarantees as defined via the game-based notions of [29] and additionally achieves two further properties that we introduce here, then it realizes our functionality.
- We give evidence that both of these additional requirements are inherently needed to define UC secure adaptor signatures, hence identifying a so-far unknown gap between established game-based notions [29] and fully composable security. We also find that the standard implementation of Schnorr-based adaptor signatures, which achieves the notions of [29], does not satisfy both additional requirements.
- We construct an enhanced Schnorr-based adaptor signature scheme that not just achieves all core security guarantees but also the two additional requirements, yielding the first UC-secure adaptor signature scheme. Our enhanced scheme does not introduce additional assumptions and is highly efficient. Crucially, our construction remains compatible with Schnorr signatures, making it suitable for deployment on Bitcoin and similar platforms.

1.2 Related Work

Adaptor signatures were initially introduced in the informal work of Poelstra [44] and later received a formal treatment by Aumayr et al. [3]. Dai et al. [20] identified shortcomings in this formalization and proposed refined game-based security definitions. Gerhart et al. [29] extended this refined framework and provided game-based security definitions for adaptor signatures in a broader context—including coin-mixing services. These further extensions were necessary, since adaptor signatures have been applied in a variety of contexts, private coin mixing [30, 45], oracle-based payment systems [41], and fair signature exchange [31, 54] beyond the initial application for Payment Channel Networks (PCNs) [22, 23, 42, 43].

Other lines of work for adaptor signatures have focused on finding adaptor signatures from a variety of assumptions. Adaptor signatures have been found in the post-quantum setting, including proposals by Esgin et al. [25], Tairi et al. [52], and Renan et al. [47]. A generic construction for adaptor signatures exists from multi-party [17] computation, and a class of dichotomic signatures can be generally transformed into adaptor signatures [24, 29].

To enable applications of adaptor signatures in which multiple signers are required, threshold and two-party adaptor signatures have been introduced [5, 24, 34]. To extend the general extracting functionality, Vanjani et al. [55] introduced functional adaptor signatures, in which the signer extracts a function of the witness rather than the witness itself.

A treatment for adaptor signatures in the Universal Composability (UC) framework was developed by Tairi, Moreno-Sanchez, and Schneidewind [53]. We provide a detailed comparison with our approach in Section 3.2.

In addition to the work of directly related to adaptor signatures, adaptor signatures share properties with verifiably encrypted signatures [8, 48]. However, these schemes differ fundamentally in that they do not support public adaptivity or efficient witness extraction from the signature–ciphertext pair.

2 Preliminaries

By $x \leftarrow\!\$\ X$, we denote the uniform sampling of x from the set X, and λ represents the security parameter. By $y \leftarrow \mathsf{A}(x; r)$ we denote a probabilistic polynomial time (PPT) algorithm A that on input x and randomness r, outputs y. When A is a deterministic polynomial time (DPT) algorithm, we use the notation $x := \mathsf{A}(y)$. When we write $if \exists x$ in an algorithm to check the existence of a value (e.g., in a tuple or relation), then within that if clause we bind the variable x to refer to that value. If there are multiple such values, x refers to the chronologically first one. We call a function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}$ negligible in n if for every $k \in \mathbb{N}$, there exists $\lambda_0 \in \mathbb{N}$ s.t. for every $\lambda \geq \lambda_0$ it holds that $|\mathsf{negl}(\lambda)| \leq \lambda^{-k}$. We use the function **assert** to check if a condition holds. The function call **assert** $\mathsf{condition}$ aborts the game in which it is called, if $\mathsf{condition}$ evaluates to false.

In this section, we focus on adaptor signatures and the UC framework. We recall definitions for non-interactive zero-knowledge proofs, signatures, and hard relations, in Appendix A.

2.1 The UC Framework

Universal Composability (UC) is a simulation-based paradigm for modeling and proving security of protocols (e.g., [11, 15, 40]). In a UC security analysis, one first defines an *ideal functionality* $\mathcal{F}$ that specifies the intended security and functionality of a protocol. The functionality $\mathcal{F}$ is secure by definition, but typically cannot be run in practice. To analyze the security of a concrete protocol $\mathcal{P}$ (called the *real protocol*), one shows that there exists a probabilistic polynomial-time (PPT) *simulator* $\mathcal{S}$ that controls the adversarial interface of $\mathcal{F}$ such that no PPT *environment* $\mathcal{E}$ can distinguish whether it is interacting with the real world (running $\mathcal{P}$) or the ideal world (running $\mathcal{F}$ with $\mathcal{S}$, denoted by $\mathcal{F} \mid \mathcal{S}$). Since $\mathcal{F}$ is secure by definition, this indistinguishability implies that $\mathcal{P}$ preserves the same guarantees. In this case, we say that $\mathcal{P}$ securely implements, or *(UC-)realizes*, $\mathcal{F}$, written as $\mathcal{P} \leq \mathcal{F}$. The UC framework supports a powerful *composition theorem*: if $\mathcal{P} \leq \mathcal{F}$, then any PPT protocol $\mathcal{Q}$ using $\mathcal{P}$ as a subroutine (written $\mathcal{Q} \mid \mathcal{P}$) also realizes the version using $\mathcal{F}$ instead (i.e., $\mathcal{Q} \mid \mathcal{P} \leq \mathcal{Q} \mid \mathcal{F}$). This allows one to first prove that $\mathcal{Q} \mid \mathcal{F}$ realizes another ideal functionality $\mathcal{F}'$, and then replace $\mathcal{F}$ with $\mathcal{P}$ without requiring further proof. Protocols such as $\mathcal{Q} \mid \mathcal{F}$ that use ideal subroutines are called hybrid protocols.

2.2 Adaptor Signatures

Adaptor signatures are primarily used in cryptocurrency protocols, such as atomic swaps [3] and payment channel constructions [24], where signatures must conform to a fixed, predefined format (e.g., Schnorr or ECDSA) to be accepted by the blockchain. This practical requirement means that adaptor signatures must produce valid instances of the underlying scheme. This limits the design space and makes compatibility with existing formats a critical constraint. Therefore, an *adaptor signature scheme* $\mathsf{AS}_{\Sigma,\mathcal{R}}$ extends a signature scheme $\Sigma = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ with four additional algorithms: $\mathsf{AS}_{\Sigma,\mathcal{R}} = (\mathsf{pSign}, \mathsf{pVrfy}, \mathsf{Adapt}, \mathsf{Extract})$. The pre-signing algorithm $\mathsf{pSign}(\mathsf{sk}, x, Y)$ allows a signer with secret key sk to tie a statement $Y \in \mathcal{L}_{\mathcal{R}}$ and a message x, producing a pre-signature $\widetilde{\sigma}$. The pre-signature can be publicly verified using $\mathsf{pVrfy}(\mathsf{pk}, x, Y, \widetilde{\sigma})$ for a public key pk. Given a witness y for the statement Y, the adapting algorithm $\mathsf{Adapt}(\mathsf{pk}, \widetilde{\sigma}, y, Y)$ transforms the pre-signature into a valid full signature σ. Finally, the extracting algorithm $\mathsf{Extract}(\mathsf{pk}, \widetilde{\sigma}, \sigma, Y)$ recovers the witness y from a valid signature–pre-signature pair. We define adaptor signatures following the latest formulation in [29]:

Definition 1 (Adaptor Signature). *An* adaptor signature scheme $\mathsf{AS}_{\Sigma,\mathcal{R}}$ *w.r.t. a signature scheme* $\Sigma = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *and a hard relation* $\mathcal{R}$ *consists of a tuple of algorithms* $\mathsf{AS}_{\Sigma,\mathcal{R}} = (\mathsf{pSign}, \mathsf{Adapt}, \mathsf{pVrfy}, \mathsf{Extract})$ *defined as*

- $\underline{\widetilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, x, Y)}$. *The* pre-signing *algorithm is a probabilistic polynomial-time (PPT) algorithm that, on input a secret key* sk*, a message* $x \in \{0,1\}^{\ell_m}$*, and a statement* $Y \in \mathcal{L}_{\mathcal{R}}$*, outputs a pre-signature* $\widetilde{\sigma}$*.*
- $\underline{b \leftarrow \mathsf{pVrfy}(\mathsf{pk}, x, Y, \widetilde{\sigma})}$. *The* pre-verification *algorithm is a deterministic polynomial-time (DPT) algorithm that, on input a public key* pk*, message* $x \in \{0,1\}^{\ell_m}$*, statement* $Y \in \mathcal{L}_{\mathcal{R}}$*, and pre-signature* $\widetilde{\sigma}$*, outputs a bit* b *indicating acceptance or rejection.*

– $\sigma := \underline{\mathsf{Adapt}(\mathsf{pk}, \widetilde{\sigma}, y, Y)}$. *The* adapting *algorithm is a DPT algorithm that, given a public key* pk*, pre-signature* $\widetilde{\sigma}$*, witness* y*, and a statement* Y*, outputs a full signature* σ*.*
– $y := \underline{\mathsf{Extract}(\mathsf{pk}, \widetilde{\sigma}, \sigma, Y)}$. *The* extracting *algorithm is a DPT algorithm that, on input a public key* pk*, a pre-signature* $\widetilde{\sigma}$*, a full signature* σ*, and a statement* $Y \in \mathcal{L}_{\mathcal{R}}$*, outputs a witness* y *such that* $(Y, y) \in \mathcal{R}$*, or outputs* $\perp$ *if extraction fails.*

To improve readability, we often omit the subscript in $\mathsf{AS}_{\Sigma,\mathcal{R}}$ and simply write AS when the underlying signature scheme Σ and relation $\mathcal{R}$ are clear from context. For technical convenience, we require the Adapt algorithm to never output an error symbol. Additionally, we modify its input to explicitly include the statement Y with respect to which the pre-signature is being adapted. This change does not restrict generality: any language can be redefined so that the witness y includes the statement as part of its structure. However, making the statement explicit simplifies the formalization of adaptor signatures in the UC framework.

Game-Based Security Definitions. Beginning with applications in Payment Channel Networks (PCNs) [22, 23, 42, 43], the security requirements for adaptor signatures have progressively evolved to accommodate more advanced use cases such as private coin mixing [30, 45] and oracle-based payments [41], among others. Based on these applications, adaptor signatures are thus generally expected [29] to provide the following three security properties, where each property is formalized via a separate security game. We provide an intuition here and defer a more formal treatment to Appendix A.6.

Extractability. Extractability generalizes existential unforgeability for signatures to the adaptor signature setting by incorporating witness extraction: The adversary can query sign and pre-sign oracles to generate signatures on messages and pre-signatures on message–statement pairs. To break extractability, an adversary must output a valid signature on a message that was not signed before (unforgeability) and that also cannot be paired with any previously generated pre-signature to extract a valid witness for the corresponding statement (witness extraction). This property protects the *signer*: if a valid signature appears (e.g., on-chain), the signer is assured that it either produced a signature on the message itself or can extract a witness from it using a pre-signature it previously issued.

Pre-Signature Adaptability. Pre-signature adaptability guarantees that any valid pre-signature—when combined with a correct witness—can always be adapted into a valid full signature on the associated message. Thus, the pre-signature acts as a verifiable commitment to a future signature that the adapter can complete once he obtains a witness, which might not yet be the case when the pre-signature is received and verified. This property, therefore, protects the *adapter* by ensuring that he can safely proceed with an exchange whenever pre-signatures are valid, confident that future possession of a witness will eventually yield a valid, publicly verifiable signature.

Unique Extractability. Adaptor signatures have also been employed in three-party settings, particularly in coin-mixing applications [30, 45], where an additional security property—unique extractability—is required to ensure that a single pre-signature corresponds to *exactly* one adapted signature, preventing ambiguity or double-spending within the protocol. That is, no efficient adversary can derive two different valid signatures on the same message that both extract with a single pre-signature on that message(potentially to different witnesses). This property protects the *signer* by ensuring that pre-signatures cannot be equivocally interpreted.

Schnorr Adaptor Signatures. Due to its compatibility with Bitcoin, the Schnorr adaptor signature [3] is the most prominent instance that we briefly recall and defer a formal description to Construction 2. The construction follows the structure of Schnorr signatures, which have a secret key $\mathsf{sk} \in \mathbb{Z}_q$ and a corresponding public key $\mathsf{pk} = g^{\mathsf{sk}}$, and uses the discrete logarithm (DLog) relation over a prime-order group $\mathbb{G}$. A statement $Y \in \mathbb{G}$ is of the form $Y = g^y$ for a witness $y \in \mathbb{Z}_q$. To produce a pre-signature on a message x, the signer samples a random $r \leftarrow\!\!\$\ \mathbb{Z}_p$, computes the nonce $R = g^r$, and derives the challenge $c = \mathsf{H}_{\mathsf{Sign}}(\mathsf{pk}, x, R \cdot Y)$.

The signer then computes $s = r + c \cdot \mathsf{sk}$ and returns the resulting pre-signature $\widetilde{\sigma} = (R \cdot Y, s)$. Given a valid witness y for Y, the adaptor can compute a full signature $\sigma = (R \cdot Y, s + y)$, which is a valid Schnorr signature under the nonce $R \cdot Y = g^{r+y}$. Anyone who receives both the pre-signature $\widetilde{\sigma}$ and the full signature σ can extract the embedded witness as $y = s' - s$ where s' is the second component of σ. A pre-signature $\widetilde{\sigma} = (R', s)$ is considered *valid* for a message x and a statement Y if, given the public key pk, the verifier can compute the challenge $c = \mathsf{H}_{\mathsf{Sign}}(\mathsf{pk}, x, R')$ and check that the following equality holds: $g^s \cdot Y \stackrel{?}{=} R' \cdot \mathsf{pk}^c$. This condition ensures that the pre-signature is structurally consistent with the Schnorr signature equation and will yield a valid signature when adapted with a correct witness.

3 A First UC definition: Technical Gaps in $\mathcal{F}_{[53]}$

As mentioned in the introduction, recently Tairi, Moreno-Sanchez, and Schneidewind proposed the first and so far only UC definition of adaptor signatures [53]; we will refer to their functionality as $\mathcal{F}_{[53]}$. Before we discuss $\mathcal{F}_{[53]}$, we present two technical observations about $\mathcal{F}_{[53]}$ that clarify its scope and place it in its timely cryptographic context.

First, Tairi et al. [53] define adaptor signature security in terms of two earlier game-based notions: unforgeability and witness extractability. These notions have since been subsumed by the more expressive notion of *extractability*, which strictly generalizes both [20]. While this shows that the UC formalization builds on a now-outdated foundation, this does not invalidate our analysis: all gaps we identify with respect to extractability remain meaningful, even when interpreted under the older notion of witness extractability. Moreover, the limitations of those earlier definitions, such as their one-time security scope [29], were neither intended by [53] nor enforced by $\mathcal{F}_{[53]}$. For these reasons, we adopt extractability as the appropriate baseline for our evaluation, but our negative results apply regardless.

Second, [53] models (pre-)signing as a distributed two-party protocol that jointly computes (pre-)signatures, as this was required for their case study. However, it is not just straightforward to interpret $\mathcal{F}_{[53]}$ as a single-signer functionality but all of our results concering $\mathcal{F}_{[53]}$ are actually independent of whether a single- or two-signer setting is considered. For simplicity of presentation and since we aim for the standard case of single-signer UC secure adaptor signatures, we present and discuss $\mathcal{F}_{[53]}$ directly in the single-signer setting.

3.1 Definition of $\mathcal{F}_{[53]}$

The definition by Tairi et al. builds on and extends the UC signature functionality of Kiayias et al. [35], which itself extends Canetti's foundational ideal functionality for digital signatures [14]. Roughly speaking, these prior ideal signature functionalities offer a Sign and a Vrfy oracle that the environment/parties in a higher-level protocol can access.[3]

When a party requests a signature on a message x, the functionality forwards the request to the simulator, which returns a signature σ to be handed back to the caller. The functionality internally maintains a list $\mathcal{Q}$ of all message–signature pairs it has issued in this way. Verification is likewise modeled as an interactive procedure. Before consulting the simulator on what the result of verification should be, the functionality first checks the list $\mathcal{Q}$ to enforce unforgeability and two functional properties:

- **(Existential) Unforgeability:** If the message x was never signed, i.e., no pair (x, σ') does appear in $\mathcal{Q}$, the functionality does not verify the signature.
- **Completeness/Correctness:** If the pair was honestly generated via signing, it verifies.
- **Consistency:** If the pair (x, σ) has been verified before, the functionality returns the same decision.

If none of these cases apply, the functionality defers to the simulators response. The final outcome is then stored in the list $\mathcal{Q}$ to ensure consistent future responses. Observe that these oracle definitions illustrate a

[3] We say "oracle" when we want to refer to the interface provided by a UC protocol. We say "algorithm" to refer to the actual algorithm being used as part of a signature scheme. In the realization of an ideal functionality, an oracle internally just runs the corresponding algorithm and returns the result.

fundamental design decision that Tairi et al. also largely follow: Rather than locally executing the corresponding algorithms to determine oracle outputs, they are instead obtained via an interactive process from the simulator.

Augmenting the Signature Functionality for the Adaptor Setting. To obtain the ideal adaptor signature functionality $\mathcal{F}_{[53]}$, Tairi et al. augment the UC signature functionality by introducing further oracles for pre-signing, pre-verification, adaptation, and extraction. To formalize additional security properties, they further introduce a second list $\mathcal{P}$ that records and keeps track of pre-signatures generated through honest pre-signing. We provide a high-level summary of $\mathcal{F}_{[53]}$ in Figure 1 and briefly describe the main concepts next, which is sufficient to follow our results. We refer interested readers to Figure 9 in [53] for the full formal definition.

When the pSign oracle is invoked on a message x and statement Y, $\mathcal{F}_{[53]}$ queries the simulator to obtain a pre signature $\widetilde{\sigma}$ – subject to the restriction that pVrfy has not previously returned false for x and $\widetilde{\sigma}$ – and stores a new entry in $\mathcal{P}$ of the form $e := (x, Y, \widetilde{\sigma}, 1)$, where the bit 1 marks the entry as valid.

The pVrfy oracle is defined similarly to the Vrfy oracle: Upon being called with a message x, a pre-signature $\widetilde{\sigma}$, and a statement Y, the oracle queries the simulator to determine the verification result, subject to the following restrictions which are computed based on the list $\mathcal{P}$ list: If $\widetilde{\sigma}$ was previously returned by the pSign oracle for x and Y, then it must verify (completeness/correctness). If the pSign oracle was never called on the message x and Y, then the verification must fail (existential unforgeability of pre-signatures). If this combination of inputs has been queried before to pVrfy, then the output must be the same (consistency). Finally, if the verification result has not yet been stored for a pre-signature–statement pair, then a new entry is added to the list $\mathcal{P}$, analogous to the pSign oracle.

The Adapt oracle can be called by providing a witness y and a pre-signature $\widetilde{\sigma}$. The oracle aborts if there is no unadapted entry $(x, Y, \widetilde{\sigma}, 1) \in \mathcal{P}$ containg $\widetilde{\sigma}$. It then checks that y is a witness for Y. If so, it locally runs the deterministic Adapt algorithm of the adaptor signature scheme on $\widetilde{\sigma}$ and y to compute a signature σ. It finally adds the pair (x, σ) to $\mathcal{Q}$ such that they will be verified by the Vrfy oracle and updates the entry in $\mathcal{P}$ to contain y and σ, effectively marking the pre-signature as adapted

The Extract oracle is then defined as the inverse of the Adapt oracle. That is, when called on a pre-signature $\widetilde{\sigma}$ and a signature σ, the Extract oracle checks whether there is a matching adapted pre-signature, such that for the pair $(\widetilde{\sigma}, \sigma)$, a witness y is stored. If so, the oracle returns y. In all other cases, the extraction oracle fails by returning $\perp$. Observe that the set $\mathcal{P}$ plays a central role in formalizing two of the major security properties from Section 2.2:

- **Pre-Signature Adaptability:** Any pre-signature either generated by pSign (which is guaranteed to verify) or verified by pVrfy can be completed into a valid signature using a valid witness via a call to the Adapt oracle. Since the resulting signature is added to $\mathcal{Q}$, it will verify in the Vrfy oracle.
- **Extractability:** If a signature verifies under Vrfy, then it was added either by the Sign or the Adapt oracle to the set $\mathcal{Q}$. In the latter case, the Adapt oracle guarantees that this is only possible for a valid witness and a pre-signature on a message that was previously pre-signed. The definition of the Extract guarantees that the witness used for the call to Adapt will be returned when queried with that pre-signature.

The ideal functionality by Tairi et al. is not designed to formalize the third property from Section 2.2, i.e., *unique extractability*, as this property was not needed for their specific case study. Tairi et al. [53] state that their ideal functionality can be realized by *any* adaptor signatures that satisfy standard game-based security:

Theorem 1 (Informal, Theorem 2 of [53]). *Let $\mathsf{AS}_{\Sigma,\mathcal{R}}$ be an adaptor signature scheme for a hard relation $\mathcal{R}$ and secure signature scheme Σ. If $\mathsf{AS}_{\Sigma,\mathcal{R}}$ is extractable and pre-signature adaptable, then $\mathsf{AS}_{\Sigma,\mathcal{R}}$ UC-realizes the ideal functionality.*

Schnorr adaptor signatures —the most widely used adaptor signatures in practice— are known to achieve these game-based security notions [24, 29]. If the above theorem were correct, it would imply that Schnorr adaptor signatures realize the $\mathcal{F}_{[53]}$ functionality. However, we show that the theorem is incorrect, and this implication does not hold.

The functionality maintains two lists: $\mathcal{Q}$ for signatures and $\mathcal{P}$ for (adapted) pre-signatures.

(Pre-)Signature Generation: Upon receiving a signing request for a message x, or a pre-signing request for a message–statement pair (x, Y), forward the request to Sim. Upon receiving a signature σ, abort if (x, σ) or (x, Y, σ) is marked invalid in $\mathcal{Q}$ or $\mathcal{P}$. Otherwise, store the result as valid and output σ.

- If this is a signature, record $(x, \sigma, 1)$ in $\mathcal{Q}$.
- If this is a pre-signature, record $(x, Y, \sigma, 1)$ in $\mathcal{P}$.

(Pre-)Signature Verification: Upon receiving a verification request for (x, Y, σ)—where $Y = \perp$ indicates that σ is a signature rather than a pre-signature—forward the request to Sim.

- If $(x, Y, \sigma, 1)$ is recorded in $\mathcal{Q}$ or $\mathcal{P}$, return 1 (completeness).
- If (x, Y, σ, f) exists, return the stored bit f (consistency).
- If no entry for $(x, Y, \cdot)$ exists, return 0 and record $(x, Y, \sigma, 0)$ in $\mathcal{Q}$ or $\mathcal{P}$ (unforgeability).
- Otherwise, return the simulator's decision and record (x, Y, σ, f) in $\mathcal{Q}$ or $\mathcal{P}$.

Adaptation: Upon receiving an adaptation request $(\mathsf{pk}, \widetilde{\sigma}, y)$, check whether $\widetilde{\sigma}$ is marked as valid and unadapted in $\mathcal{P}$, and whether pk is a valid public key. If not, abort. Otherwise, compute the adapted signature locally, update the entry in $\mathcal{P}$ with the adapted signature and witness.

Extraction: Upon receiving an extraction request $(\mathsf{pk}, \widetilde{\sigma}, \sigma)$, return the stored witness if $(\widetilde{\sigma}, \sigma)$ appears as an adapted pair in $\mathcal{P}$; otherwise, return $\perp$.

Fig. 1. Simplified functionality of Tairi et al. [53]. We omit key generation, session identifiers, and internal bookkeeping details, which are not relevant for this overview. For the full specification, see [53].

3.2 Issues in the existing UC Definition

As our first main contribution, we identify a series of severe conceptual and technical issues in $\mathcal{F}_{[53]}$. They can be categorized into three types depending on their implications:

- We identify *additional requirements of $\mathcal{F}_{[53]}$ on realizations that are not captured by Theorem 1*, therefore falsifying the theorem. Notably, we can show that Schnorr adaptor signatures *do not* realize $\mathcal{F}_{[53]}$. This shows that stronger preconditions on realizations are needed.
- We show that $\mathcal{F}_{[53]}$ *formalizes neither pre-signature adaptability nor extractability* towards higher-level protocols. A hybrid protocol relying on $\mathcal{F}_{[53]}$ can hence not use these properties in its security analysis, defeating the point of a modular UC analysis.
- We show that $\mathcal{F}_{[53]}$ *fundamentally cannot be realized by any adaptor signature scheme*, even under additional stronger assumptions. This is due to certain underlying design decisions that we discuss in the following.

In what follows, we explain each issue.

Requirement: Existential Pre-signature Unforgeability. First, we observe that in $\mathcal{F}_{[53]}$ a pre-signature may successfully pre-verify on a message-statment pair (x, Y) only if (x, Y) was previously pre-signed. We show two ways in which Schnorr adaptor signatures [3] do not comply with this notion.

The first way, we call "reverse adapting". Given a statement–witness pair (Y, y) and a valid Schnorr signature (σ_1, σ_2) on a message x, one can compute a valid pre-signature by "reversing" the adaptation: simply set $\widetilde{\sigma} = (\sigma_1, \sigma_2 - y)$. This pre-signature will satisfy the pre-verification check for (x, Y), even though it was never produced via an honest pre-signing invocation. In other words, an adversary with access to a valid signature and a statement-witness pair of its choice can forge a valid pre-signature on this statement.

The second way, we call "statement shifting". That is, given a valid pre-signature $\widetilde{\sigma} = (R \cdot Y, s)$ on some statement $Y = g^y$, it is possible to derive a valid pre-signature on a related statement without querying the pre-signing oracle. From a valid pre-signature $\widetilde{\sigma} = (R \cdot Y, s)$, one can compute a new valid pre-signature $\widetilde{\sigma}' = (R \cdot Y,\ s - 1)$ which is valid w.r.t. the shifted statement $Y' = Y \cdot g = g^{y+1}$. This transformation preserves the combined commitment $R \cdot Y$, which appears in the challenge computation. This pre-signature is valid

under the same message as $\widetilde{\sigma}$ but under the shifted statement Y' and breaks pre-signature unforgeability. Interestingly, this property of Schnorr does not contradict extractability: For both $\widetilde{\sigma}$ and $\widetilde{\sigma}'$, adapting with the corresponding witness will yield the same overall signature σ since the addition of the witness removes the difference introduced by statement shifting. Hence, when the signer obtains σ and runs extract on the original $\widetilde{\sigma}$ that he generated, he will obtain the witness y on his statement Y (rather than the $y+1$ on Y').

Requirement: Extractability limited by number of Pre-Signatures. Observe that the Extract oracle in $\mathcal{F}_{[53]}$ returns a witness only if there exists an entry $e := (x, Y, \widetilde{\sigma}, 1, y, \sigma)$ that was previously added by the Adapt oracle by updating an entry of the form $e' := (x, Y, \widetilde{\sigma}, 1)$. Such an entry e' can only be added to $\mathcal{P}$ by the pSign and pVrfy oracles – at most one per call to each oracle. Hence $\mathcal{F}_{[53]}$ requires something similar to a *one-to-one relationship* between the number of pre-signatures seen so far and witnesses that can be extracted. This is not implied by extractability and pre-signature adaptability and hence falsifies Theorem 1.

As a concrete counterexample, consider the adaptor signature scheme AS constructed in Figure 5 of [29]. This scheme is extractable and pre-signature adaptable. But there exists a ppt attacker A that, given a single valid pre signature $\widetilde{\sigma}$, can compute two distinct full signatures $\sigma \neq \sigma'$ such that both signatures are valid and both of them extract w.r.t. $\widetilde{\sigma}$. This scheme therefore does not realize $\mathcal{F}_{[53]}$: An environment can distinguish both cases by running A, who queries only the pSign oracle and only once, and then checking whether the Extract oracle returns a witness for both σ and σ'.

One way to prevent this distinguishing attack would be to additionally require that the adaptor signature scheme AS provides *unique extractability*. In fact, it seems possible – albeit out of the scope of this work – to formally show that if there is a negligible likelihood for pre-signatures generated by the pSign algorithm to collide, then such an adaptor signature scheme AS can realize $\mathcal{F}_{[53]}$ only if it achieves unique extractability. This is despite $\mathcal{F}_{[53]}$ not being designed to and indeed not formalizing this security property towards higher-level protocols.

Requirement: Adaptability/Extraction only for Verified Pre-Signatures. The Adapt and Extract oracles in $\mathcal{F}_{[53]}$ will return an output only for pre-signatures that are marked as verified in $\mathcal{P}$, either because they were previously returned by the pSign oracle or successfully verified by the pVrfy oracle. No existing adaptor signature construction provides these properties, e.g., compare the Adapt (Extract) algorithm used for Schnorr adaptor signatures, which will also return an output for invalid pre-signatures. Since Adapt does not take the message x as input, existing algorithms also cannot simply be changed to first verify the pre-signature by running the pVrfy algorithm. While it might be possible to construct adaptor signatures with these properties, this really seems to be an unintended artifact of $\mathcal{F}_{[53]}$.

Property not formalized: Pre-Signature Adaptability. The Adapt oracle in $\mathcal{F}_{[53]}$ uses the pre-signature input $\widetilde{\sigma}$ to locate *an* entry in the list that matches $\widetilde{\sigma}$, i.e, has the form $e := (x, Y, \widetilde{\sigma}, 1)$, and then updates that entry with an adapted signature σ. However, there might be a second entry $e' := (x', Y, \widetilde{\sigma}, 1)$, indicating that the same pre-signature is also associated with a different message x'. This can happen if the pSign oracle is called twice, once on x and once on x', and the attacker chooses to provide the same pre-signature $\widetilde{\sigma}$ for both queries.

As a consequence, the resulting adapted signature σ is guaranteed to be valid with respect to x but will be invalid for x', thereby *contradicting pre-signature adaptability*. Notably, this has the effect that a higher-level/hybrid protocol building on $\mathcal{F}_{[53]}$ cannot actually use/assume that any pre-signature which verifies for some message x' will be adapted to a valid signature for the same message.

Property not formalized: Extractability. Higher-level/hybrid protocols using $\mathcal{F}_{[53]}$ further cannot make use of the extractability property since this is also not actually captured. Specifically, consider the following sequence: An honest signer P_{hon} in a higher-level protocol uses the pSign oracle on some message x to obtain a pre-signature $\widetilde{\sigma}$. A malicious party P_{mal} then uses the pVrfy oracle to verify a different pre-signature $\widetilde{\sigma}'$ w.r.t. the same message x. Since that message has already been pre-signed, $\mathcal{F}_{[53]}$ allows the adversary, which is controlled by the environment in hybrid settings, to return successful verification for $\widetilde{\sigma}'$ (existential unforgeability for pre-signatures). The malicious party P_{mal} can then use the Adapt oracle on $\widetilde{\sigma}'$ to obtain a

valid signature σ which $\mathcal{F}_{[53]}$ then guarantees to extract (only) w.r.t. to $\widetilde{\sigma}'$. However, *σ will not extract w.r.t. to any pre-signature that P_{hon} knows*, i.e., malicious parties can obtain a valid signature without having to reveal a witness to the honest signer.

Jumping slightly ahead, it appears that the underlying issue lies in existential unforgeability actually being too weak to define extractability in a UC setting. We address this point by instead requiring a form of strong unforgeability. We will further discuss in Section 5.1 why this appears to be an inherent requirement due to the difference between non-composable game-based security and composable UC security.

Unrealizability: One-Time Adaptability. A pre-signature $\widetilde{\sigma}$ in $\mathcal{F}_{[53]}$ can only be adapted if an entry of the form $e := (x, Y, \widetilde{\sigma}, 1)$ is present in the list $\mathcal{P}$. Once the pre-signature is successfully adapted, the functionality updates the corresponding entry to $e := (x, Y, \widetilde{\sigma}, 1, y, \sigma)$. As a result, further invocations of the adaptation oracle using the same pre-signature are no longer possible, since no entry of the required form $(x, Y, \widetilde{\sigma}, 1)$ is present in $\mathcal{P}$.

This is impossible to achieve with any (non-interactive) adaptor signature scheme: the Adapt algorithm would have to fail for some party P_1 depending on whether a different party P_2 has already run the Adapt algorithm on the same input.

Unrealizability Due to Local Algorithms. Fundamentally, the entire design of $\mathcal{F}_{[53]}$ is based on the implicit assumption that all algorithms can be executed only by calling the corresponding oracle of $\mathcal{F}_{[53]}$ such that the functionality always has full information about all inputs and actions. Such a design is possible for algorithms relying on secret inputs, such as the pSign algorithm, which requires a secret signing key that is abstracted by $\mathcal{F}_{[53]}$ and hence can only be computed by using the corresponding pSign oracle of $\mathcal{F}_{[53]}$. However, several algorithms, notably Adapt and Extract, take only public inputs and can hence be locally computed by an environment without using the oracles provided by $\mathcal{F}_{[53]}$. This allows for several trivial distinguishing attacks on *all* adaptor signature schemes.

As just one concrete example, consider the following scenario: The environment uses the pSign oracle to obtain a (valid) pre-signature $\widetilde{\sigma}$ on a message x and a statement Y for which it knows a corresponding witness y. The environment then internally computes the Adapt *algorithm* on $(\widetilde{\sigma}, y)$ to obtain an adapted signature σ – without querying the Adapt *oracle* of the functionality/real protocol. Finally, it calls the Extract oracle on $\widetilde{\sigma}$ and σ. By definition of $\mathcal{F}_{[53]}$, since the Adapt oracle was never used, there is no record in $\mathcal{P}$ containing σ or even just the witness – the Extract oracle will hence return $\perp$ in the ideal world. However, in the real world, the Extract oracle will run the Extract algorithm, which returns y as the signature was correctly adapted (formally follows from extractability, which is required to be able to realize $\mathcal{F}_{[53]}$).

4 Technical Outline: Formalizing $\mathcal{F}_{\text{asig}}$

While $\mathcal{F}_{[53]}$ is an important step towards universally composable adaptor signatures with strong security guarantees, the variety of technical issues we found in $\mathcal{F}_{[53]}$ shows that defining such an ideal adaptor signature functionality turns out to be non-trivial and much harder than perhaps expected.

We fill this gap in the literature by proposing a new ideal adaptor signature functionality $\mathcal{F}_{\text{asig}}$. Our goals for this functionality are as follows: Besides the usual *correctness* sanity check, it shall formalize the three key security properties of adaptor signatures – *extractability*, *unique extractability*, and *pre-signature adaptability* from Section 2.2 – in such a manner that composed hybrid protocols built on top of $\mathcal{F}_{\text{asig}}$ can rely on these security properties for their security analysis. The functionality shall further be realizable by correct adaptor signature schemes meeting the corresponding game-based notions of *extractability*, *unique extractability*, and *pre-signature adaptability* from Section 2.2, plus possibly some additional requirements needed for defining and obtaining composable security. We will identify these additional requirements and aim to keep them minimal.

This section presents the core technical ideas underlying the design of $\mathcal{F}_{\text{asig}}$. We begin by identifying the inherent modeling challenges that arise when attempting to capture the behavior of adaptor signature schemes in the UC framework. We then detail the design of $\mathcal{F}_{\text{asig}}$ – thus obtaining the *first ideal adaptor*

signature functionality that is not just realizable but also offers composable security guarantees that can be used by higher-level protocols.

4.1 Inherent Conflicts in Modeling Adaptor Signature Security

A key challenge in defining *any* ideal adaptor signature functionality lies in resolving the fundamental conflict between "positive" security properties such as correctness and pre-signature adaptability, which imply that signatures must verify under certain conditions, and "negative" properties such as extractability and unique extractability, which imply that signatures must not verify under certain conditions. This conflict is often relatively easy to resolve in game-based notions, which consider only one property at a time and can argue over the whole run that has already concluded while having access to all information of that run. In contrast, a UC ideal functionality runs into the issue that it has to essentially "commit" on one or the other decision, e.g., whether a signature verifies, while the run is still ongoing and while having only limited information. Once committed, the functionality must be able to "patch" all future actions in such a way that they remain consistent with the required implications.

For example, consider pre-signature adaptability and extractability: intuitively, pre-signature adaptability is the implication "if a pre-signature verifies under pVrfy, then a signature returned by Adapt given a valid witness is guaranteed to verify under Vrfy". Extractability intuitively is the implication "if a signature was not returned by Sign and does not extract under Extract for any pre-signature (returned by $\widetilde{\sigma}$), then it must not verify". The ideal functionality now has to resolve the following situation: If it returns true for a pre-signature $\widetilde{\sigma}$ entered into the pVrfy oracle, then the functionality is committed in the sense that all signatures σ returned by the Adapt oracle must verify using the Vrfy oracle (pre-signature adaptability). At the same time, the ideal functionality should be defined to reject such signatures if they are not extractable. Furthermore, if an adapted σ is accepted by Vrfy due to pre-signature adaptability, then the ideal functionality is again committed in the sense that, for extractability, the Extract oracle has to return a witness for at least one pre-signature $\widetilde{\sigma}'$ previously returned by pSign that might potentially not be the same as $\widetilde{\sigma}$, say, because pre-signatures are malleable.

Jumping slightly ahead, to resolve this clash, we identify conditions that $\mathcal{F}_{\text{asig}}$ can impose on pVrfy such that $\mathcal{F}_{\text{asig}}$ remains realizable, yet these conditions are sufficient to imply that any resulting adapted signatures are extractable. Since pVrfy does not yet have access to a witness, this also involves patching the output of the Extract oracle in certain cases, which bears some similarity to the functionality of Tairi et al. [53]. Both of these changes allow $\mathcal{F}_{\text{asig}}$ to always accept such adapted signatures in Vrfy. Supporting unique extractability in $\mathcal{F}_{\text{asig}}$ brings similar complications and requires imposing further conditions on pVrfy as well as patching of the Adapt output to avoid clashes with pre-signature adaptability in Vrfy. A major challenge here is that patching of Adapt depends on the unique extractability property, which itself is defined over the signed message x, yet the Adapt oracle does not have access to x. We address this point by additionally requiring that valid pre-signatures are bound to a unique message x such that Adapt can retrieve that message based on the pre-signature.

4.2 A New Functionality

The issues that we identified for $\mathcal{F}_{[53]}$ are partly caused by an underlying fundamental design decision, namely, the implicit assumption that all algorithms can only and indeed are always executed via oracle calls. As explained above, this is not the case for locally executable algorithms relying only on publicly available information such as Adapt, Extract, pVrfy, Vrfy. For $\mathcal{F}_{\text{asig}}$ we therefore start from scratch following a different design philosophy: Similar to other existing ideal signature functionalities such as [37,39], all oracles of $\mathcal{F}_{\text{asig}}$ will internally run the real algorithms and then only perform certain additional checks on top, possibly altering the output in the process, to enforce security properties. While many aspects and certain ideas of the functionality of Tairi et al. [53] will also be represented in some form in $\mathcal{F}_{\text{asig}}$, by following this different definitional style, we can indeed show that $\mathcal{F}_{\text{asig}}$ is realizable while providing expected security guarantees to higher-level protocols.

The functionality is parameterized by a polytime decidable relation $\mathcal{R}$, ppt algorithms $\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{pSign}$ and dpt algorithms $\mathsf{Vrfy}, \mathsf{pVrfy}, \mathsf{Adapt}, \mathsf{Extract}$. Upon first activation, initialize this instance by running and storing $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.

On $\mathtt{PubKey}$? from anyone, return $(\mathtt{PubKey}, \mathsf{pk})$.

On $(\mathtt{Sign}, x)$ for party pid in session $sid = (pid, sid')$, do:
Return $(\mathtt{Signature}, \mathsf{Sign}(\mathsf{sk}, x))$

On $(\mathtt{pSign}, x, Y)$ for party pid in session $sid = (pid, sid')$, do:
Return $(\mathtt{PreSignature}, \mathsf{pSign}(\mathsf{sk}, x, Y))$

On $(\mathtt{Adapt}, pk, \widetilde{\sigma}, y, Y)$ for party pid, return $(\mathtt{Adapt}, \mathsf{Adapt}(pk, \widetilde{\sigma}, y, Y))$

On $(\mathtt{Extract}, pk, \widetilde{\sigma}, \sigma, Y)$ for party pid, return $(\mathtt{Extract}, \mathsf{Extract}(pk, \widetilde{\sigma}, \sigma, Y))$

On $(\mathtt{pVerify}, pk, x, Y, \widetilde{\sigma})$ for party pid, return $(\mathtt{VerResult}, \mathsf{pVrfy}(pk, x, Y, \widetilde{\sigma}))$

On $(\mathtt{Verify}, pk, x, \sigma)$ for party pid, return $(\mathtt{VerResult}, \mathsf{Vrfy}(pk, x, \sigma))$

As usual, the attacker may corrupt any party pid dynamically and gains full control over that party. In addition, if pid is the owner of the key pair modeled by this instance, i.e., the session ID of this instance is $sid = (pid, sid')$, then the attacker also learns sk; for notational convenience, we set $\mathsf{corrupted} = \mathsf{true}$ iff this is the case.

Fig. 2. Baseline ideal functionality for adaptor signatures/envisioned realization. The functionality is parameterized by the local algorithms invoked in response to oracle queries. At this stage, no security properties are enforced; the functionality simply reflects the behavior of a concrete adaptor signature scheme when used as realization.

It turns out that to define and reconcile the three key security properties of extractability, adaptability, and unique extractability within a single composable ideal functionality $\mathcal{F}_{\text{asig}}$, the functionality additionally has to formalize and require two further properties that, at first glance, might seem unrelated (strongly unforgeable and message-bound pre-signatures). In Section 5.1 we discuss why both of these properties appear to be inherently needed for any definition of an ideal adaptor signature functionality.

In the rest of this technical outline, we give an overview of the construction of $\mathcal{F}_{\text{asig}}$ via a sequence of refinement steps. We start from a baseline version that only runs the corresponding algorithms without any security guarantees, i.e., essentially the envisioned realization (cf. Figure 2). We then iteratively add changes that enforce security properties until we reach the final version (cf. Figure 5). This should not only help follow and verify the design rationale. It also serves as an extended proof sketch, as these steps correspond to game hops in our realization proof.

Correctness. We begin by modifying the Sign and pSign oracles to internally maintain lists $\mathsf{H}_{\mathsf{sign}}$ and $\mathsf{H}_{\mathsf{pSign}}$, which store all message-signature pairs or message-pre-signature-statement triples that have been successfully produced via signing or pre-signing, respectively. Additionally, we introduce the lists $\mathsf{H}_{\mathsf{verify}}$ and $\mathsf{H}_{\mathsf{pVerify}}$, which record the same type of pairs (triples) but for successfully verified calls the Vrfy and pVrfy oracles. Finally, we add the list $\mathsf{H}_{\mathsf{adapt}}$, which stores tuples of the form $(x, \widetilde{\sigma}, Y, y, \sigma)$ whenever the Adapt oracle is used on input pre-signature $\widetilde{\sigma}$ and valid witness y for statement Y such that $(x, \widetilde{\sigma}, Y) \in \mathsf{H}_{\mathsf{pVerify}}$. In other words, $\mathsf{H}_{\mathsf{adapt}}$ keeps track of pre-signatures that were successfully pre-verified and are now adapted correctly for the same statement with a valid witness. We note that together with the next change (message-bound pre-signatures; introduced right after correctness), there will be at most one message x such that $(x, \widetilde{\sigma}, Y) \in \mathsf{H}_{\mathsf{pVerify}}$, i.e., the message x will be uniquely defined.

These lists are used as bookkeeping to enforce security properties in the following steps. We keep separate lists for each operation rather than trying to merge several steps into a single list, such as $\mathcal{P}$ in $\mathcal{F}_{\text{[53]}}$. This makes it easier to formalize fine-grained security properties later on. It also reduces the risk of accidental incompatibilities with realizations due to locally executable algorithms: local execution only affects individual lists for the corresponding algorithms, rather than everything in a single merged list.

Based on these lists, we first enforce correctness. To this end, we augment the Vrfy and pVrfy oracles to consult the $\mathsf{H}_{\mathsf{sign}}$ and $\mathsf{H}_{\mathsf{pSign}}$ lists: When the oracle is called, it checks whether the inputs appear in $\mathsf{H}_{\mathsf{sign}}$ or $\mathsf{H}_{\mathsf{pSign}}$ respectively, i.e., were previously generated by a successful (pre-)sign operation. If so, the oracle overwrites a potentially negative verification result of the (pre-)verification algorithm with true, ensuring that all honestly generated (pre-)signatures always verify successfully. Additionally, we enhance the Extract oracle to preserve correctness in witness extraction for signatures computed by the Adapt oracle. More precisely, when extracting from a pre-signature–signature–statement triple $(\widetilde{\sigma}, \sigma, Y)$, the functionality checks whether this triple is recorded in $\mathsf{H}_{\mathsf{adapt}}$. If so, but the extract algorithm fails to return a valid witness for Y, the functionality overrides the output with the correct witness y previously stored in $\mathsf{H}_{\mathsf{adapt}}$ (choosing the first such witness if there are multiple stored in $\mathsf{H}_{\mathsf{adapt}}$). Observe that, by definition of $\mathsf{H}_{\mathsf{adapt}}$, this guarantees correct witness extraction for signatures returned by the Adapt oracle using a verified pre-signature with a valid witness. It does not, however, provide any guarantees for adapted signatures using a pre-signature that was not pre-verified via the pVrfy oracle, using an invalid witness or different statement, or for signatures computed by running the Adapt algorithm locally without querying the oracle. This is therefore not yet a very useful property on its own but will rather be needed to reconcile extractability with pre-signature adaptability in a later step (see paragraph *Pre-Signature Unforgeability*).

We provide the corresponding version of our functionality in Figure 13, and formally prove in Claim 1 that every adaptor signature scheme AS satisfying both message-boundedness and correctness securely realizes this version of the functionality.

Message-bound pre-signatures. Once established correctness, we enforce that each pre-signature is valid for exactly one message. As mentioned in Section 4.1 and further discussed later in Section 5.1, while this is not implied by the key security properties that we want $\mathcal{F}_{\text{asig}}$ to formalize, it appears to be a necessary additional requirement to reconcile pre-signature adaptability and unique extractability in the definition of a composable ideal functionality. We refer to this requirement as having *message-bound* pre-signatures, following prior terminology [9, 19, 33, 51].

To enforce this property, the functionality checks during pre-signing and pre-verification whether a candidate pre-signature $\widetilde{\sigma}$ has already been accepted as valid for a different message during pre-signing or pre-verification, i.e., already is in the sets $\mathsf{H}_{\mathsf{pSign}}$ or $\mathsf{H}_{\mathsf{pVerify}}$. If so, the pre-signing oracle instead returns an error $\perp$, and the pre-verification oracle returns false. Observe that the negative *message-bound pre-signature* property does not clash with the positive *correctness* property due to how we change the pSign oracle: pre-signature creation is successful and hence will add the message and pre-signature to $\mathsf{H}_{\mathsf{pSign}}$ only if the pre-signature does not violate message-boundedness. But from then on, the pre-signature is bound to that message, i.e., will also not violate message-boundedness during verification w.r.t. the original message. One can thus safely define verification to always succeed in this case.

We depict this enhanced version of the functionality in Figure 14 with changes marked in red, and prove in Claim 2 that any adaptor signature scheme AS with message-bound pre-signatures (cf. Definition 3) realizes this version.

Pre-Signature Adaptability. Due to careful preparation in the previous steps, we can now add *pre-signature adaptability* via a very simple change: The Vrfy oracle is changed to check whether the signature σ and message x occur in the list $\mathsf{H}_{\mathsf{adapt}}$ and, if so, always returns true. Due to the definition of $\mathsf{H}_{\mathsf{adapt}}$, the functionality now indeed guarantees that any pre-signature first successfully verified by the pVrfy oracle and then adapted correctly via the Adapt oracle will result in a signature that verifies in Vrfy. While there are no guarantees in cases where signatures were adapted by running the Adapt algorithm locally rather than using the Adapt oracle, this guarantee is sufficient since honest parties in higher-level protocols will always use the oracle. We present the modifications for pre-signature adaptability in Figure 15 and show that it can be realized by adaptor signatures that additionally provide pre-signature adaptability in Claim 3.

Unique Extractability. Next, we extend the functionality to guarantee *unique extractability*. Recall that this property requires that for any given pre-signature $\widetilde{\sigma}$ which is valid for a message x and statement Y, there exists at most one valid full signature σ on the same message x such that the pair $(\widetilde{\sigma}, \sigma)$ enables extraction

of a valid witness for statement Y. Our functionality must enforce this condition at all points where new valid (pre-)signatures are introduced, since those may later be used in verification or extraction. Concretely, this situation arises in three cases:

- **During signing and pre-signing:** (Pre-)signatures generated by oracles are automatically considered valid due to correctness by the previous step. Thus, (pre-)signature generation must not output (pre-)signatures that would violate unique extractability.
- **During (pre-)verification:** any new (pre-)signature violating unique extractability must not (pre-)verify.
- **During honest adaptation:** Due to pre-signature adaptability, we define all full signatures contained in $\mathsf{H}_{\mathsf{adapt}}$ to always be considered valid. Thus, Adapt will then be able to introduce new valid signatures, and we have to change the oracle now to ensure those signatures added to $\mathsf{H}_{\mathsf{adapt}}$ do not violate unique extractability.

We now explain how we enforce unique extractability. When a new pre-signature $\widetilde{\sigma}$ on some message x and statement Y appears - either because the pSign oracle internally computed $\widetilde{\sigma}$ or because pVrfy was called with $\widetilde{\sigma}$ given as input - the functionality checks whether there already exist *two distinct valid signatures* $\sigma_1 \neq \sigma_2$ for x such that both $(\widetilde{\sigma}, \sigma_1)$ and $(\widetilde{\sigma}, \sigma_2)$ successfully extract (possibly different) witnesses for Y. Checking this involves considering the lists $\mathsf{H}_{\mathsf{sign}}$ and $\mathsf{H}_{\mathsf{verify}}$, which contain all valid signatures seen so far, both due to the correctness of signing and also due to past calls to the Vrfy oracle. If such a pair of distinct signatures is found, the pre-signature $\widetilde{\sigma}$ needs to be rejected by the functionality: the pSign oracle is changed to instead return an error, and the pVrfy oracle instead returns false. Note that this change does not clash with the correctness property of pVrfy because we have modified pSign as well: a pre-signature is added to $\mathsf{H}_{\mathsf{pSign}}$ only if it passes the unique extractability check during pre-signing, so it can still safely be accepted by pVrfy. Since this consistency check must be performed both during pre-signing and pre-verification, we encapsulate it in the helper method BreakpSigUnExt, depicted in Figure 6.

A similar situation arises when a new signature σ on some message x appears - during a call to the Sign, Vrfy, or Adapt oracles. In this case, the functionality must ensure that unique extractability is preserved with respect to existing valid pre-signatures for the same message x. Specifically, it checks whether there exists a valid pre-signature $\widetilde{\sigma}$ on x and some statement Y (stored in $\mathsf{H}_{\mathsf{pSign}}$ or $\mathsf{H}_{\mathsf{pVerify}}$) and a valid signature σ' on x (stored in $\mathsf{H}_{\mathsf{sign}}$, $\mathsf{H}_{\mathsf{verify}}$, or $\mathsf{H}_{\mathsf{adapt}}$) such that the Extract oracle would return witnesses for Y from both $(\widetilde{\sigma}, \sigma)$ and $(\widetilde{\sigma}, \sigma')$. If such a conflicting signature σ' exists, the functionality takes two actions: the Sign oracle returns an error, consistently with the pSign oracle. The Adapt is modified to instead return the already known extracting signature σ'. Both actions effectively prevent the introduction of non-uniquely extracting signatures. At the same time, the Vrfy oracle is modified to reject σ, ensuring that at most one valid extracting signature is accepted per pre-signature. We capture this behavior in the helper function BreakSigUnExt, shown in Figure 6.

Let us point out several important details: First, signature correctness is again compatible with the changes to Vrfy by the same reasoning as for pre-signature correctness. Second, while for pSign and Sign we can simply return errors in the case where a conflicting (pre-)signature would be introduced, the Adapt oracle has to follow the reprogramming approach. This is the case since, for pre-signature adaptability, we have to ensure that Adapt, when given correct inputs, will return a signature. Third, to run the BreakSigUnExt check in Adapt, Adapt needs access to the underlying message x, which we can obtain from the input pre-signature thanks to message-boundedness introduced in a previous step. Fourth, since we have changed the Extract oracle to sometimes return witnesses when the underlying Extract algorithm might not, namely whenever pre-signature and signature are in the set $\mathsf{H}_{\mathsf{adapt}}$, the helper function BreakSigUnExt needs to be a bit more involved and take this modification into account. In contrast, BreakpSigUnExt can be simpler since the Extract oracle differs from the Extract algorithm only if the pre-signature $\widetilde{\sigma}$ being checked by BreakpSigUnExt is in $\mathsf{H}_{\mathsf{adapt}}$, in which case $\widetilde{\sigma}$ is not new but has already previously been verified to not violate unique extractability.

We show in Claim 4 that an adaptor signature scheme that is uniquely extractable, in addition to the already required properties, satisfies the enhanced intermediate functionality (Figure 16).

Pre-Signature Unforgeability. The next property we enforce is *pre-signature unforgeability*. As already mentioned in Sections 3.2 and 4.1 and further discussed in Section 5.1, just as for message-boundedness this property is not implied by the three main properties we want to model but appears to be an essential additional requirement needed to resolve the tension between extractability and pre-signature adaptability in a composable ideal functionality. Concretely, we require that if a message x has not been signed, then it must be infeasible to produce a new valid pre-signature $\widetilde{\sigma}$ on x for some statement Y such that $\widetilde{\sigma}$ was not returned by running pSign on x and Y. On a technical level, this property is enforced by adding a check to the pVrfy oracle that overrides the output to false whenever a pre-signature would violate this unforgeability notion.

This condition resolves the aforementioned conflict: If x has already been signed, then extractability is trivially fulfilled, even if no witnesses can be computed by $\mathsf{Extract}$, such that the functionality can safely allow pre-signature adaptability on any pre-signature for that message. Conversely, if x is unsigned, then any valid pre-signature that might get adapted must have originated from the pSign oracle, ensuring that the witness entered in the Adapt oracle and hence returned by the $\mathsf{Extract}$ oracle will be extractable w.r.t. a pre-signature that the signer knows and for a statement Y that the signer chose. We defer a detailed discussion of this pre-signature unforgeability notion to Section 5.1 and provide the corresponding version of the functionality in Figure 17. We prove in Claim 5 that any adaptor signature scheme additionally satisfying our definition of pre-signature unforgeability (cf. Definition 2) realizes this functionality.

Extractability. The last guarantee we need to add is *extractability*. Again, due to careful preparations in the previous steps leading to disjoint cases among the negative extractability and the positive pre-signature adaptability property, we can now formalize extractability via another simple change as follows. The Vrfy oracle is extended to additionally check the extractability property, but only in those cases where the signature is not already guaranteed to verify by correctness or by pre-signature adaptability. If the check determines that extractability would be violated, then the output of Vrfy is changed to be false.

It is obvious that this change does not break any of the previous properties while guaranteeing extractability for any signature that does not fall under the correctness or pre-signature adaptability check. For signatures that fall under the correctness check, we have that they were previously returned by the Sign oracle such that extractability always holds trivially. For the case of signatures falling under pre-signature adaptability, extractability also always holds by what we discussed in the *pre-signature unforgeability* paragraph. Hence, the ideal functionality formalizes extractability for all possible cases.

We present the modifications for extractability in Figure 18. This finally yields our complete functionality $\mathcal{F}_{\mathrm{asig}}$ as given in Figure 5. We show that this last modification is realizable by adaptor signature schemes that additionally provide extractability in Claim 6.

4.3 Realizing Our Functionality

As a general result, we show in Theorem 2 that all game-based secure adaptor signature schemes that additionally satisfy our new notions of *message-boundedness* and *pre-signature unforgeability* UC-realize $\mathcal{F}_{\mathrm{asig}}$. The proof is structured as a sequence of games that follow along the modifications presented in the previous section.

To obtain a concrete realization, we first show that standard Schnorr adaptor signatures satisfy message-boundedness but fail to meet the requirement of unforgeable pre-signatures. We hence propose an enhanced version of the Schnorr adaptor signature scheme that we show to be secure and satisfy both additional properties. By Theorem 2 it immediately follows that our enhanced scheme realizes $\mathcal{F}_{\mathrm{asig}}$, thereby establishing the *first UC-secure adaptor signature scheme*. Notably, our scheme still produces and hence remains compatible with Schnorr signatures, thus serving as a drop-in replacement. In what follows, we give a more detailed overview of our results for Schnorr adaptor signatures.

Schnorr Adaptor Signatures Satisfy Message-Boundedness. Message-boundedness requires that a pre-signature is valid for at most one message; that is, no pre-signature $\widetilde{\sigma} = (R', s)$ should verify successfully on two distinct messages $x \neq x'$. Schnorr adaptor signatures achieve message-boundness because a Schnorr pre-signature (R', s) is valid for exactly one challenge $c := \mathsf{H}_{\mathsf{Sign}}(\mathsf{pk}, x, R')$. Under the random oracle model, $\mathsf{H}_{\mathsf{Sign}}$ is

collision-resistant and injective over distinct inputs (with overwhelming probability). So, unless $x = x'$, such a collision cannot occur. Therefore, a pre-signature cannot verify two different messages, and Schnorr adaptor signatures satisfy message-boundedness. We formalize this argument in Section 6.1.

Pre-Signature Unforgeable Schnorr Adaptor Signatures. The "statement shifting" property of standard Schnorr adaptor signatures that we identify in Section 3.2 directly falsifies pre-signature unforgeability: given a valid pre-signature, one can compute a valid pre-signature for a different statement that was never pre-signed. To create an adaptor signature scheme with unforgeable pre-signatures, we propose a modification to Schnorr adaptor signatures in which each pre-signature $\widetilde{\sigma} = (R, \widetilde{s})$ on a message x is accompanied by a Schnorr proof-of-knowledge of the underlying randomness r used in the nonce $R = g^r$. A statement for such a proof is the tuple $(x, R, \widetilde{s}, Y)$ and the corresponding witness is r, such that $g^r = R$. By the simulation-sound extractability of the proof [7], one cannot modify the pre-signature to be valid w.r.t. a different Y without knowing the witness r. By the structure of Schnorr (pre-)signatures, requiring to prove knowledge of r is equivalent to requiring the knowledge of the signing key sk, since $\widetilde{s} = \mathsf{sk} \cdot h + r$ for a publicly known h. Thus, valid pre-signatures cannot be obtained by malling valid signatures or other valid pre-signatures without knowing the signing key sk.

This effectively enforces that only the legitimate signer can produce valid pre-signatures, rendering the scheme pre-signature unforgeable. Most notably, we implement this proof of knowledge using a standard Schnorr-style proof-of-knowledge, requiring no additional setup beyond the existing random oracle, and imposing the same computational overhead as signing itself. This makes our enhancement both secure and efficient, while retaining compatibility with the Schnorr-signature verification algorithm.

In more detail, a pre-signature in our modified scheme consists of two components: the standard Schnorr adaptor pre-signature $\widetilde{\sigma}$, and a proof π proving knowledge of the randomness r used for computing $\widetilde{\sigma}$. This proof takes the form of a Schnorr-style proof of knowledge: the signer samples a fresh scalar $r' \leftarrow\!\!\$\ \mathbb{Z}_p$ and computes $R' = g^{r'}$ along with the response $s' = r \cdot \mathsf{H}_{\mathsf{Prove}}(R', \widetilde{\sigma}, Y, x) + r'$, where $\mathsf{H}_{\mathsf{Prove}}$ is a hash function modeled as a random oracle. The full enhanced pre-signature is then

$$\widetilde{\sigma}' = (\widetilde{\sigma}, \pi) = \big(\widetilde{\sigma},\ (R', s')\big),$$

and it is considered valid if the pre-signature $\widetilde{\sigma}$ pre-verifies, *and* the proof-of-knowledge is valid. Checking the validity of the proof-of-knowledge equals is similar to verifying Schnorr signatures. I.e., it requires verifying the equation $R^{\mathsf{H}_{\mathsf{Prove}}(R', \widetilde{\sigma}, Y, x)} \cdot R' \stackrel{?}{=} g^{s'}$.

For the adapt and extract algorithms, our enhanced scheme ignores the additional proof and just runs the original algorithms on the pre-signature part. We provide a detailed description and proof that this scheme satisfies all required security properties in Section 6.1. Most importantly, our enhanced Schnorr adaptor signature scheme realizes our functionality $\mathcal{F}_{\text{asig}}$.

5 An Ideal Adaptor Signature Functionality

In this section, we provide technical details for the overview given in Section 4.2. We start by formally introducing the two new properties that an adaptor signature scheme must satisfy – in addition to correctness and standard security (extractability, unique extractability, pre-sig adaptability) – to realize our functionality. As we detail below, these additional requirements appear to be *inherently required for any UC formalization of ideal adaptor signatures*, thus exposing a previously unknown gap between standard (non-composable and isolated) game-based security notions and composable unified UC security. Later in Section 6.1, we show that our enhanced Schnorr adaptor signature scheme satisfies all required properties. After discussing these properties, we present the formal definition of our ideal adaptor signature functionality in Figure 5. We finally discuss some further properties of adaptor signatures that have been considered in other works, but that are beyond the scope of our functionality.

5.1 Additional Properties for Realizing Our Functionality

The two additionally required properties are *message-bound pre-signatures* and *unforgeable pre-signatures*. We formally define and discuss both properties in what follows.

Unforgeable Pre-Signatures. An adaptor signature scheme satisfies *unforgeable pre-signatures* if it is infeasible for an adversary to compute a fresh valid pre-signature for a message-statement pair where the message was never signed by the signer. This property is crucial to simultaneously enforce the notions of extractability and pre-signature adaptability in an ideal functionality for adaptor signatures such as $\mathcal{F}_{\text{asig}}$. In particular, extractability quantifies over all *honestly generated* pre-signatures, while adaptability considers all *pre-verified* pre-signatures, regardless of their origin. Without unforgeable pre-signatures, the two notions would conflict: for example, if an adversary generates a valid but fresh pre-signature $\widetilde{\sigma}^*$, pre-signature adaptability would require the resulting adapted signature to verify. However, extractability would require that the adapted signature is valid only if it also extracts with a pre-signature $\widetilde{\sigma}$ output by the pSign oracle, which cannot be enforced by an ideal functionality in general. Crucially, while an ideal functionality would have obtained a witness for such a fresh pre-signature $\widetilde{\sigma}^*$ during adaptation, that witness might not match any statement that was previously pre-signed via the pSign oracle. This witness, hence, cannot be used to enforce extractability by programming the $\mathsf{Extract}$ oracle. The statement shifting property of Schnorr adaptor signatures that we identify in Section 3.2 actually gives a concrete example of this issue, where the witness y' used during adaptation does not match any previously pre-signed statement Y and hence cannot be used by an ideal functionality to enforce extractability, yet this issue does not break game-based extractability.

The pre-signature unforgeability requirement prevents this inconsistency by disallowing such adversarially generated pre-signatures altogether. As we show in Section 6.1, unforgeable pre-signatures are not implied by existing security guarantees since Schnorr adaptor signatures satisfy these notions but fail for unforgeable pre-signatures.

Definition 2 (Strong Existentially Unforgeable Pre-Signatures). *An adaptor signature scheme* AS *satisfies* strong existentially unforgeable pre-signatures *if for every* PPT *adversary* $\mathcal{A}$ *there exists a negligible function* negl *such that for every* $\lambda \in \mathbb{N}$*,*

$$\Pr\left[\mathsf{Un\text{-}Forge\text{-}pSig}_{\mathcal{A},\mathsf{AS}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda),$$

where the experiment $\mathsf{Un\text{-}Forge\text{-}pSig}_{\mathcal{A},\mathsf{AS}}$ *is given in Figure 3, and the probability is over all randomness.*

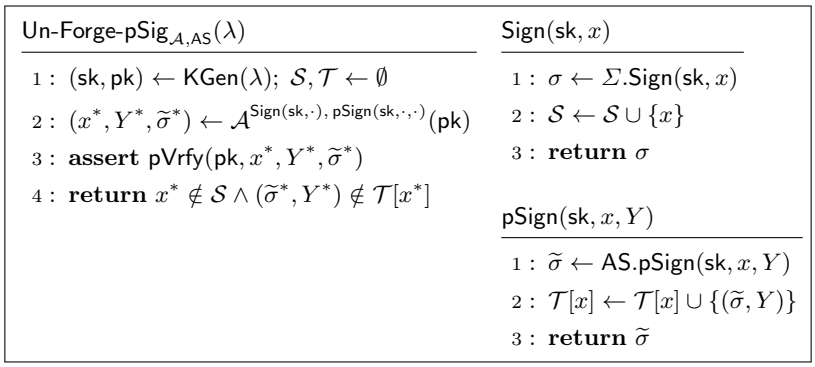

Fig. 3. The unforgeable pre-signature experiment $\mathsf{Un\text{-}Forge\text{-}pSig}_{\mathcal{A},\mathsf{AS}}(\lambda)$.

We note that the previous functionality $\mathcal{F}_{[53]}$ also formalizes a notion of existentially unforgeable pre-signatures. However, this notion is not strong enough to resolve the tension between adaptability and extractability since it allows for malleable pre-signatures (c.f. Section 3.2). In contrast, our notion enforces

strongly existentially unforgeable pre-signatures, and we can show in Theorem 2 that this notion suffices to simultaneously enforce pre-signature adaptability and extractability.

Message-Bound Pre-Signatures. Another property we require is message-bound pre-signatures. An adaptor signature scheme has *message-bound pre-signatures* if any given pre-signature is valid for at most one message, except with negligible probability. The concept of message-bound signatures is well-established for ordinary signatures [9, 19, 33, 51]; here, we adapt and refine this idea in the context of pre-signatures.

The underlying issue that message-bound pre-signatures resolve is that the Adapt and Extract algorithms, and hence the corresponding oracles in an ideal functionality, do not take the original message that was pre-signed as input. Indeed, at the time of adaptation, the adapting party might be entirely unaware of the original message. However, information about the underlying message is needed in the ideal functionality to enforce the unique extractability property during adaptation, since this property depends on the (pre-) signed message. By requiring message-bound pre-signatures – i.e., each pre-signature verifying for at most one unique message – the ideal functionality is able to look up the corresponding message in its verification history (c.f. Section 4.1).

Definition 3 (Message-Bound Pre-Signatures). *An adaptor signature scheme* AS *has* message-bound pre-signatures *if for every* PPT *adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that for every* $\lambda \in \mathbb{N}$*,*

$$\Pr\left[\mathsf{Un\text{-}Msg\text{-}pSig}_{\mathcal{A},\mathsf{AS}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda) \ ,$$

where the experiment $\mathsf{Un\text{-}Msg\text{-}pSig}_{\mathcal{A},\mathsf{AS}}$ *is defined in Figure 4, and the probability is taken over the randomness of all probabilistic algorithms.*

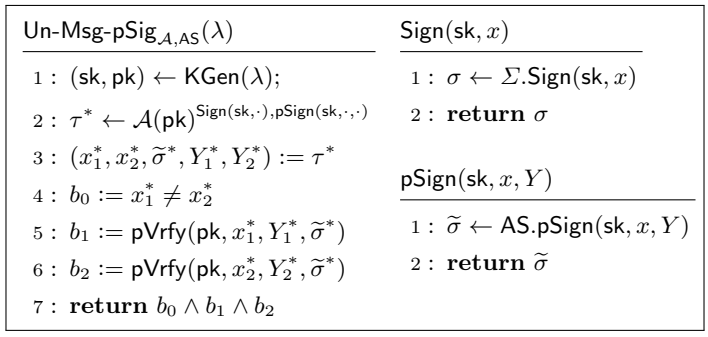

Fig. 4. The message-bound pre-signature experiment $\mathsf{Un\text{-}Msg\text{-}pSig}_{\mathcal{A},\mathsf{AS}}(\lambda)$.

5.2 The Functionality

In our technical outline, we have already discussed the core design principles and intermediate stages that led to our final ideal functionality. In this section, we present the complete version of $\mathcal{F}_{\text{asig}}$, shown in Figure 5, which guarantees *extractability*, *unique extractability*, and *pre-signature adaptability* to higher-level protocols, while imposing as few additional constraints on adaptor signature schemes as possible and being as simple and modular as possible.

In terms of party and session modeling, $\mathcal{F}_{\text{asig}}$ follows established conventions from previous ideal signature functionalities: Each instance of $\mathcal{F}_{\text{asig}}$ represents one signature key pair that can be used by an unbounded number of parties, each of them identified by a party ID pid_i. To distinguish different key pairs, instances are identified via a session ID of the form $sid = (pid, sid')$ where pid is the ID of the party owning the key pair. The sub-identifier sid' allows for each party owning multiple key pairs, if needed. All parties pid_i can call all

The functionality is parameterized by a polytime decidable relation $\mathcal{R}$, ppt algorithms $\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{pSign}$ and dpt algorithms $\mathsf{Vrfy}, \mathsf{pVrfy}, \mathsf{Adapt}, \mathsf{Extract}$. Upon first activation, initialize this instance by running and storing $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.

On $\texttt{PubKey}$? from anyone, return $(\texttt{PubKey}, \mathsf{pk})$.

On $(\texttt{Sign}, x)$ for party pid in session $sid = (pid, sid')$, do:
- $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, x)$
- **if** $\mathsf{BreakSigUnExt}(x, \sigma) \neq \bot$**:** $\sigma := \bot$ {*Unique Extractability*
- **if** $\sigma \neq \bot$**: add** σ **to** $\mathsf{H}_{\mathsf{sign}}[x]$
- Return $(\texttt{Signature}, \sigma)$.

On $(\texttt{pSign}, x, Y)$ for party pid in session $sid = (pid, sid')$, do:
- $\widetilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, x, Y)$
- **if** $\neg\mathsf{corrupted} \wedge \exists x' \neq x : \widetilde{\sigma} \in \mathsf{H}_{\mathsf{pSign}}[x'] \cup \mathsf{H}_{\mathsf{pVerify}}[x']$**:** $\widetilde{\sigma} := \bot$ {*Message Bounding*
- **if** $\mathsf{BreakpSigUnExt}(x, \widetilde{\sigma}, Y) \neq \bot$**:** $\widetilde{\sigma} := \bot$ {*Unique Extractability*
- **if** $\widetilde{\sigma} \neq \bot$**: add** $(\widetilde{\sigma}, Y)$ **to** $\mathsf{H}_{\mathsf{pSign}}[x]$
- Return $(\texttt{PreSignature}, \widetilde{\sigma})$

On $(\texttt{Adapt}, pk, \widetilde{\sigma}, y, Y)$ for party pid, do:
- $\sigma \leftarrow \mathsf{Adapt}(pk, \widetilde{\sigma}, y, Y)$
- **if** $pk = \mathsf{pk} \wedge (Y, y) \in \mathcal{R} \wedge \exists x : (\widetilde{\sigma}, Y) \in \mathsf{H}_{\mathsf{pVerify}}[x]$**:**
  - **if** $\sigma' := \mathsf{BreakSigUnExt}(x, \sigma) \wedge \sigma' \neq \bot$**:** $\sigma \leftarrow \sigma'$ {*Unique Extractabilty*
  - **add** $(\widetilde{\sigma}, Y, y, \sigma)$ **to** $\mathsf{H}_{\mathsf{adapt}}[x]$
- Return $(\texttt{Adapt}, \sigma)$

On $(\texttt{Extract}, pk, \widetilde{\sigma}, \sigma, Y)$ for party pid, do:
- $y := \mathsf{Extract}(pk, \widetilde{\sigma}, \sigma, Y)$
- **if** $(Y, y) \notin \mathcal{R} \wedge \exists (x, y') : (\widetilde{\sigma}, Y, y', \sigma) \in \mathsf{H}_{\mathsf{adapt}}[x]$**:** $y := y'$ {*Extract Correctness*
- Return $(\texttt{Extract}, y)$

On $(\texttt{pVerify}, pk, x, Y, \widetilde{\sigma})$ for party pid, do:
- $b_{\mathsf{pVerify}} := \mathsf{pVrfy}(pk, x, Y, \widetilde{\sigma})$
- **if** $pk = \mathsf{pk} \wedge (\widetilde{\sigma}, Y) \in \mathsf{H}_{\mathsf{pSign}}[x]$**:** $b_{\mathsf{pVerify}} := 1$ {*Pre-Sign Correctness*
- **else if** $pk = \mathsf{pk} \wedge \neg\mathsf{corrupted}$**:**
  - **if** $\exists x' \neq x : \widetilde{\sigma} \in \mathsf{H}_{\mathsf{pSign}}[x'] \cup \mathsf{H}_{\mathsf{pVerify}}[x']$**:** $b_{\mathsf{pVerify}} := 0$ {*Message Bounding*
    - **if** $\mathsf{BreakpSigUnExt}(x, \widetilde{\sigma}) \neq \bot$**:** $b_{\mathsf{verify}} := 0$ {*Unique Extractability*
    - **if** $\mathsf{H}_{\mathsf{sign}}[x] = \emptyset$**:** $b_{\mathsf{pVerify}} := 0$ {*Pre-Sig Unforgeability*
- **if** $pk = \mathsf{pk} \wedge b_{\mathsf{pVerify}}$**: add** $(\widetilde{\sigma}, Y)$ **to** $\mathsf{H}_{\mathsf{pVerify}}[x]$
- Return $(\texttt{VerResult}, b_{\mathsf{pVerify}})$

On $(\texttt{Verify}, pk, x, \sigma)$ for party pid, do:
- $b_{\mathsf{verify}} := \mathsf{Vrfy}(pk, x, \sigma)$
- $b_{\mathsf{extracts}} := \exists (\widetilde{\sigma}, Y) \in \mathsf{H}_{\mathsf{pSign}}[x] : (Y, \mathsf{Extract}(pk, \widetilde{\sigma}, \sigma, Y)) \in \mathcal{R}$
- **if** $pk = \mathsf{pk} \wedge (x, \sigma) \in \mathsf{H}_{\mathsf{sign}}$**:** $b_{\mathsf{verify}} := 1$ {*Correctness*
- **else if** $pk = \mathsf{pk} \wedge \exists (\widetilde{\sigma}, Y, y, \sigma) \in \mathsf{H}_{\mathsf{adapt}}[x]$**:** $b_{\mathsf{verify}} := 1$ {*Pre-Sig Adaptability*
- **else if** $pk = \mathsf{pk} \wedge \neg\mathsf{corrupted}$**:**
  - **if** $\mathsf{BreakSigUnExt}(x, \sigma) \neq \bot$**:** $b_{\mathsf{verify}} := 0$ {*Unique Extractabilty*
  - **if** $\mathsf{H}_{\mathsf{sign}}[x] = \emptyset \wedge \neg b_{\mathsf{extracts}}$**:** $b_{\mathsf{verify}} := 0$ {*Extractabilty*
- **if** $pk = \mathsf{pk} \wedge b_{\mathsf{verify}}$**: add** σ **to** $\mathsf{H}_{\mathsf{verify}}[x]$
- Return $(\texttt{VerResult}, b_{\mathsf{verify}})$

As usual, the attacker may corrupt any party pid dynamically and gains full control over that party. In addition, if pid is the owner of the key pair modeled by this instance, i.e., the session ID of this instance is $sid = (pid, sid')$, then the attacker also learns sk; for notational convenience, we set $\mathsf{corrupted} = \mathsf{true}$ iff this is the case.

Fig. 5. Ideal adoptor signatures functionality $\mathcal{F}_{\text{asig}}$.

$\mathsf{BreakSigUnExt}(x, \sigma)$

1 : **if** corrupted : **return** $\bot$
2 : $\mathcal{Q} \leftarrow \{(\widetilde{\sigma}, Y) | (\widetilde{\sigma}, Y) \in \mathsf{H}_{\mathsf{pVerify}}[x] \cup \mathsf{H}_{\mathsf{pSign}}[x] : (Y, \mathsf{Extract}(\mathsf{pk}, \widetilde{\sigma}, \sigma, Y)) \in \mathcal{R}\}$
3 : **if** $\exists (\widetilde{\sigma}, Y, \cdot, \sigma') \in \mathsf{H}_{\mathsf{adapt}}[x] : (\sigma' \neq \sigma \wedge (\widetilde{\sigma}, Y) \in \mathcal{Q})$
4 : **return** σ'
5 : **if** $\exists \sigma' \neq \sigma \in \mathsf{H}_{\mathsf{sign}}[x] \cup \mathsf{H}_{\mathsf{verify}}[x], (\widetilde{\sigma}, Y) \in \mathcal{Q} : (Y, \mathsf{Extract}(\mathsf{pk}, \widetilde{\sigma}, \sigma', Y)) \in \mathcal{R}$
6 : **return** σ'
7 : **return** $\bot$

$\mathsf{BreakpSigUnExt}(x, \widetilde{\sigma}, Y)$

1 : **if** corrupted : **return** $\bot$
2 : $\mathcal{Q} \leftarrow \{\sigma | \sigma \in \mathsf{H}_{\mathsf{verify}}[x] \cup \mathsf{H}_{\mathsf{sign}}[x] : (Y, \mathsf{Extract}(\mathsf{pk}, \widetilde{\sigma}, \sigma, Y)) \in \mathcal{R}\}$
3 : **if** $|\mathcal{Q}| > 1$: **return** $\top$
4 : **return** $\bot$

Fig. 6. Helper Functions for Unique Extractability.

oracles of the functionality $\mathcal{F}_{\text{asig}}$ except for Sign and pSign, which are restricted to the party *pid* contained in the session ID $sid = (pid, sid')$. This captures that only the owner knows the secret key and hence is the only one able to (pre-)sign.

Let us briefly discuss one design decision: We chose to parameterize $\mathcal{F}_{\text{asig}}$ with a set of algorithms rather than, as is often done in such functionalities, asking the adversary to provide these algorithms at the start of the run. Note that this does not weaken the security statement or expressiveness. One can prove a higher-level protocol based on $\mathcal{F}_{\text{asig}}$ to be secure for any arbitrary but fixed set of parameters of $\mathcal{F}_{\text{asig}}$ – which is precisely what one would have to show if $\mathcal{F}_{\text{asig}}$ were to instead ask the adversary to provide algorithms. However, fixing algorithms as parameters actually increases impressiveness: One can alternatively also show security of a higher-level protocol based on $\mathcal{F}_{\text{asig}}$ only for a specific set of algorithms, such as those compatible with an output format used in the higher-level protocol, rather than being forced to consider arbitrary ones. Fixing algorithms as parameters also completely avoids a class of artificial distinguishing attacks that often arise when an attacker chooses not to respond to a request for algorithms (see [10] for an in-depth discussion).

5.3 Further Properties of Adaptor Signatures

Beyond the core security guarantees formalized in our functionality, adaptor signatures are sometimes associated with additional privacy properties such as *witness privacy* [20] and the stronger notion of *unlinkability* [29]. These properties aim to ensure that, in the absence of the original pre-signature, an adversary cannot distinguish an adapted pre-signature from a standard signature. Another relevant property is *pre-verify soundness* [29], which strengthens adaptability by requiring that pre-signatures do not verify for invalid statements. We deliberately do not model these properties in our functionality. Our objective is to capture the essential *security* guarantees that are both necessary and sufficient for securely composing adaptor signatures in higher-level protocols. Including unlinkability or pre-verify soundness would significantly increase the complexity of the functionality without offering fundamental benefits in the composable setting.

In practice, we advocate delegating the enforcement of *pre-verify soundness* to the surrounding higher-level protocol when required: if the language of statements is efficiently decidable, then this is just a trivial additional check that can be performed in parallel to pre-verification – a formalization via an ideal functionality would thus provide no benefit. If the language cannot be decided in polynomial time without access to a witness, then *pre-verify soundness* cannot even be defined in an ideal functionality, which is required to be PPT, so any potentially resulting issues would have to be resolved by the higher-level protocol anyway.

As for *unlinkability*, which constitutes a privacy rather than a security guarantee, we consider it outside the scope of our functionality. If desired, one option to make use of *unlinkability* in combination with $\mathcal{F}_{\text{asig}}$ is by requiring that the parameters, i.e., algorithms, of $\mathcal{F}_{\text{asig}}$ are such that the combination of pSign and Adapt and the Sign algorithms satisfy the game-based unlinkability notion. A security analysis of a higher-level protocol based on $\mathcal{F}_{\text{asig}}$ can then perform a reduction to the game-based unlinkeability notion.

6 Realizing the Ideal Functionality

We show that any adaptor signature scheme satisfying the game-based security properties defined in [29], along with our additional requirement of unforgeable, message-bound pre-signatures, securely realizes the ideal functionality $\mathcal{F}_{\text{asig}}$. To establish this, we construct a simulator that ensures indistinguishability between the real and ideal worlds: no environment can distinguish whether it is interacting with the real-world protocol $\mathsf{AS}_{\Sigma,\mathcal{R}}$ or with the ideal functionality $\mathcal{F}_{\text{asig}}$ in interaction with the simulator. This demonstrates that the real protocol leaks no more information to an adversary than the ideal functionality, thereby achieving simulation-based security in the UC framework.

Theorem 2. *Let* $\mathsf{AS}_{\Sigma,\mathcal{R}} = (\mathsf{pSign}, \mathsf{pVrfy}, \mathsf{Adapt}, \mathsf{Extract})$ *be an adaptor signature scheme for a correct signature scheme* $\Sigma = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ *and a hard relation* $\mathcal{R}$*. Let* $\mathcal{F}_{asig}$ *be parameterized with the same algorithms. If* $\mathsf{AS}_{\Sigma,\mathcal{R}}$ *is extractable, pre-signature adaptable, uniquely extractable, correct, and has unforgeable and message-bound pre-signatures, then*

$$\mathsf{AS}_{\Sigma,\mathcal{R}} \leq \mathcal{F}_{asig}.$$

We defer the formal proof of Theorem 2 to Appendix B. Conceptually, the proof proceeds via a sequence of game hops: we begin with a simple base functionality (Figure 2) and gradually extend it to capture all security properties formalized by our target functionality $\mathcal{F}_{\text{asig}}$. We omit a detailed overview here, as the corresponding steps have already been discussed in our technical outline in Section 4.2.

6.1 Schnorr Realization of $\mathcal{F}_{\text{asig}}$

We have already discussed the mechanics of Schnorr adaptor signatures in Section 2.2, and a formal construction is provided in Appendix A. In this section, we analyze whether Schnorr adaptor signatures can realize our ideal functionality $\mathcal{F}_{\text{asig}}$. We begin by showing that the standard construction fails to satisfy *pre-signature unforgeability*, and thus cannot realize $\mathcal{F}_{\text{asig}}$. To address this limitation, we introduce an enhanced construction that preserves the correctness and security guarantees of the original scheme, while additionally ensuring unforgeable pre-signatures. To the best of our knowledge, this yields the first adaptor signature scheme that UC-realizes an ideal functionality for adaptor signatures.

Lemma 1. *Schnorr adaptor signatures (Construction 2) do not satisfy pre-signature unforgeability (Definition 2).*

Proof. We show a concrete attack demonstrating that Schnorr adaptor signatures violate pre-signature unforgeability. The attack leverages a technique we call *statement shifting*. Let $(\mathsf{pk}, \mathsf{sk})$ be a Schnorr key pair, and let x be a message. Let $Y = g^y$ be a statement from the DLog hard relation, and let $\widetilde{\sigma} = (R \cdot Y, s)$ be a valid pre-signature on x for the statement Y, obtained via the pre-signing oracle. Here, R is the nonce commitment, and s is the masked response value. The Schnorr adaptor signature scheme computes the challenge as $c = H(\mathsf{pk}, R \cdot Y, x)$ and verifies a pre-signature (R', s) by checking that

$$g^s \cdot Y = R' \cdot \mathsf{pk}^c.$$

Now consider a new statement $Y' = Y \cdot g = g^{y+1}$ and define a new pre-signature

$$\widetilde{\sigma}' = (R',\ s - 1).$$

Note that $\widetilde{\sigma}'$ reuses the same commitment $R' = R \cdot Y = R \cdot Y' \cdot g^{-1}$ and the same message x, and thus the challenge remains unchanged, i.e., $c' = H(\mathsf{pk}, R', x) = c$. We now verify that $\widetilde{\sigma}'$ is a valid pre-signature on x with respect to Y'. The verification equation becomes:

$$g^{s-1} \cdot Y' = g^{s-1+y+1} = g^s \cdot Y = \mathsf{pk}^c \quad \Leftrightarrow \quad g^s = R' \cdot \mathsf{pk}^c,$$

which holds by assumption since $\widetilde{\sigma}$ was valid. Hence, $\widetilde{\sigma}'$ is a valid pre-signature on the same message x but with respect to the shifted statement Y'. Since $\widetilde{\sigma}'$ was computed without querying the pre-signing oracle on x and Y', this constitutes a successful forgery in the sense of pre-signature unforgeability (Definition 2). Therefore, Schnorr adaptor signatures do not satisfy pre-signature unforgeability.

To overcome this limitation, we propose an enhanced Schnorr adaptor signature scheme that achieves pre-signature unforgeability by augmenting each pre-signature with a simulation-sound extractable, Schnorr-style zero-knowledge proof of knowledge of the signing randomness.

Construction 1 (Enhanced Schnorr Adaptor Signature Scheme) *Let* $\mathsf{AS} = (\mathsf{pSign}, \mathsf{Adapt}, \mathsf{pVrfy}, \mathsf{Extract})$ *denote the standard Schnorr adaptor signature scheme as defined in Figure 9. We define the enhanced scheme* $\mathsf{AS}' = (\mathsf{pSign}', \mathsf{Adapt}', \mathsf{pVrfy}', \mathsf{Extract}')$ *in Figure 7.*

$\mathsf{pSign}'(\mathsf{sk}, x, Y)$

1 : $r \leftarrow\!\$\ \mathbb{Z}_p$;
2 : $\widetilde{\sigma} := \mathsf{pSign}(\mathsf{sk}, x, Y; r)$
3 : $r' \leftarrow\!\$\ \mathbb{Z}_p$; $R' := g^{r'}$
4 : $c' := \mathsf{H}_{\mathsf{Prove}}(\mathsf{pk}, x, \widetilde{\sigma}, Y, R')$
5 : $s' := r' + c' \cdot r$
6 : **return** $(\widetilde{\sigma}, (R', s'))$

$\mathsf{pVrfy}'(\mathsf{pk}, x, Y, \widetilde{\sigma})$

1 : $((\widetilde{R}, \widetilde{s}), (R', s')) := \widetilde{\sigma}$
2 : $c' := \mathsf{H}_{\mathsf{Prove}}(\mathsf{pk}, x, \widetilde{\sigma}_1, Y, R')$
3 : $b_1 := \mathsf{pVrfy}(\mathsf{pk}, x, Y, \widetilde{\sigma}_1)$
4 : $b_2 := [g^{s'} = (\widetilde{R}/Y)^{c'} \cdot R']$
5 : **return** $b_1 \wedge b_2$

$\mathsf{Adapt}'(\mathsf{pk}, \widetilde{\sigma}, y, Y)$

1 : $((R \cdot Y, s), (\cdot, \cdot)) := \widetilde{\sigma}$
2 : **return** $(R \cdot Y, s + y)$

$\mathsf{Extract}'(\mathsf{pk}, \widetilde{\sigma}, \sigma, Y)$

1 : $(\cdot, \widetilde{s}), (\cdot, \cdot)) := \widetilde{\sigma}$
2 : $(\cdot, s) := \sigma$
3 : **return** $s - \widetilde{s}$

Fig. 7. Enhanced Schnorr Adaptor Signature Construction for the DLog relation.

Theorem 3. *The enhanced Schnorr adaptor signature construction (Construction 1) is correct and achieves extractability, unique extractability, pre-signature adaptability, and message-bound, unforgeable pre-signatures in the random oracle model.*

We defer the proof of Theorem 3 to Appendix B, and provide intuition for why the theorem holds. For all desired properties—except for pre-signature unforgeability—we rely on the fact that the added proof of knowledge is zero-knowledge. This allows us to reduce the security of the enhanced scheme to that of the underlying Schnorr adaptor signature scheme. In particular, we can use the (pre-)signing oracles from the security experiments of the standard scheme to obtain ordinary pre-signatures, and then employ the zero-knowledge simulator of the proof system to augment them into valid pre-signatures of the enhanced scheme—*without* knowing the underlying randomness. Using this approach, we show that all security properties of the standard Schnorr adaptor signature scheme carry over to the enhanced construction.

To show message-bound pre-signatures, we utilize the collision resistance of the random oracle model and show that it is infeasible for a pre-signature to be valid under two distinct messages. For pre-signature unforgeability, we proceed similarly, but now rely on the *extractability* of the Schnorr adaptor signature scheme and the simulation-sound extractability of the Schnorr proof-of-knowledge [7]. We simulate the pre-signing and signing oracles using the oracles provided by the reduction against extractability and the simulator of the PoK. When the adversary outputs a pre-signature forgery, we use the extractor of the proof to extract the randomness used for such a pre-signature. Given this randomness, the reduction can compute the signing key and trivially break extractability, thereby contradicting the underlying scheme's security.

References

1. Defillama: Total value locked (tvl). https://defillama.com, accessed: 2025-05-13
2. Ethereum smart contracts stats – etherscan. https://etherscan.io/contractsVerified, accessed: 2025-05-13
3. Aumayr, L., Ersoy, O., Erwig, A., Faust, S., Hostáková, K., Maffei, M., Moreno-Sanchez, P., Riahi, S.: Generalized channels from limited blockchain scripts and adaptor signatures. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part II. LNCS, vol. 13091, pp. 635–664. Springer, Cham (Dec 2021). https://doi.org/10.1007/978-3-030-92075-3_22
4. Backes, M., Hofheinz, D.: How to break and repair a universally composable signature functionality. In: Zhang, K., Zheng, Y. (eds.) ISC 2004. LNCS, vol. 3225, pp. 61–72. Springer, Berlin, Heidelberg (Sep 2004). https://doi.org/10.1007/978-3-540-30144-8_6
5. Baecker, R., Gerhart, P., Katz, J., Schröder, D.: Fair exchange for decentralized autonomous organizations via threshold adaptor signatures. Cryptology ePrint Archive, Report 2025/388 (2025), https://eprint.iacr.org/2025/388
6. Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: Juels, A., Wright, R.N., De Capitani di Vimercati, S. (eds.) ACM CCS 2006. pp. 390–399. ACM Press (Oct / Nov 2006). https://doi.org/10.1145/1180405.1180453
7. Bernhard, D., Pereira, O., Warinschi, B.: How not to prove yourself: Pitfalls of the Fiat-Shamir heuristic and applications to Helios. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 626–643. Springer, Berlin, Heidelberg (Dec 2012). https://doi.org/10.1007/978-3-642-34961-4_38
8. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer, Berlin, Heidelberg (May 2003). https://doi.org/10.1007/3-540-39200-9_26
9. Brendel, J., Cremers, C., Jackson, D., Zhao, M.: The provable security of Ed25519: Theory and practice. In: 2021 IEEE Symposium on Security and Privacy. pp. 1659–1676. IEEE Computer Society Press (May 2021). https://doi.org/10.1109/SP40001.2021.00042
10. Camenisch, J., Enderlein, R.R., Krenn, S., Küsters, R., Rausch, D.: Universal composition with responsive environments. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part II. LNCS, vol. 10032, pp. 807–840. Springer, Berlin, Heidelberg (Dec 2016). https://doi.org/10.1007/978-3-662-53890-6_27
11. Camenisch, J., Krenn, S., Küsters, R., Rausch, D.: iUC: Flexible universal composability made simple. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part III. LNCS, vol. 11923, pp. 191–221. Springer, Cham (Dec 2019). https://doi.org/10.1007/978-3-030-34618-8_7
12. Camenisch, J., Krenn, S., Shoup, V.: A framework for practical universally composable zero-knowledge protocols. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 449–467. Springer, Berlin, Heidelberg (Dec 2011). https://doi.org/10.1007/978-3-642-25385-0_24
13. Camenisch, J., Stadler, M.: Proof systems for general statements about discrete logarithms. Technical Report 260, Department of Computer Science, ETH Zurich (March 1997), available at ftp://ftp.inf.ethz.ch/pub/publications/tech-reports/
14. Canetti, R.: Universally composable signature, certification, and authentication. In: Proceedings. 17th IEEE Computer Security Foundations Workshop, 2004. pp. 219–233 (2004). https://doi.org/10.1109/CSFW.2004.1310743
15. Canetti, R.: Universally composable security. J. ACM **67**(5) (Sep 2020). https://doi.org/10.1145/3402457, https://doi.org/10.1145/3402457
16. Canetti, R., Fischlin, M.: Universally composable commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Berlin, Heidelberg (Aug 2001). https://doi.org/10.1007/3-540-44647-8_2

17. Ciampi, M., Liu, X., Tzannetos, I., Zikas, V.: Universal adaptor signatures from blackbox multi-party computation. Cryptology ePrint Archive, Paper 2024/1773 (2024), https://eprint.iacr.org/2024/1773
18. Cleve, R.: Limits on the security of coin flips when half the processors are faulty (extended abstract). In: 18th ACM STOC. pp. 364–369. ACM Press (May 1986). https://doi.org/10.1145/12130.12168
19. Cremers, C., Düzlü, S., Fiedler, R., Fischlin, M., Janson, C.: BUFFing signature schemes beyond unforgeability and the case of post-quantum signatures. In: 2021 IEEE Symposium on Security and Privacy. pp. 1696–1714. IEEE Computer Society Press (May 2021). https://doi.org/10.1109/SP40001.2021.00093
20. Dai, W., Okamoto, T., Yamamoto, G.: Stronger security and generic constructions for adaptor signatures. In: Isobe, T., Sarkar, S. (eds.) INDOCRYPT 2022. LNCS, vol. 13774, pp. 52–77. Springer, Cham (Dec 2022). https://doi.org/10.1007/978-3-031-22912-1_3
21. De Santis, A., Micali, S., Persiano, G.: Non-interactive zero-knowledge proof systems. In: Pomerance, C. (ed.) CRYPTO'87. LNCS, vol. 293, pp. 52–72. Springer, Berlin, Heidelberg (Aug 1988). https://doi.org/10.1007/3-540-48184-2_5
22. Decker, C., Wattenhofer, R.: A fast and scalable payment network with bitcoin duplex micropayment channels. In: Pelc, A., Schwarzmann, A.A. (eds.) Stabilization, Safety, and Security of Distributed Systems. pp. 3–18. Springer International Publishing, Cham (2015)
23. Eckey, L., Faust, S., Hostáková, K., Roos, S.: Splitting payments locally while routing interdimensionally. Cryptology ePrint Archive, Report 2020/555 (2020), https://eprint.iacr.org/2020/555
24. Erwig, A., Faust, S., Hostáková, K., Maitra, M., Riahi, S.: Two-party adaptor signatures from identification schemes. In: Garay, J. (ed.) PKC 2021, Part I. LNCS, vol. 12710, pp. 451–480. Springer, Cham (May 2021). https://doi.org/10.1007/978-3-030-75245-3_17
25. Esgin, M.F., Ersoy, O., Erkin, Z.: Post-quantum adaptor signatures and payment channel networks. In: Chen, L., Li, N., Liang, K., Schneider, S.A. (eds.) ESORICS 2020, Part II. LNCS, vol. 12309, pp. 378–397. Springer, Cham (Sep 2020). https://doi.org/10.1007/978-3-030-59013-0_19
26. Fischlin, M.: Communication-efficient non-interactive proofs of knowledge with online extractors. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 152–168. Springer, Berlin, Heidelberg (Aug 2005). https://doi.org/10.1007/11535218_10
27. Freire, E.S.V., Hesse, J., Hofheinz, D.: Universally composable non-interactive key exchange. In: Abdalla, M., De Prisco, R. (eds.) SCN 14. LNCS, vol. 8642, pp. 1–20. Springer, Cham (Sep 2014). https://doi.org/10.1007/978-3-319-10879-7_1
28. Garay, J.A., MacKenzie, P.D., Yang, K.: Strengthening zero-knowledge protocols using signatures. Journal of Cryptology **19**(2), 169–209 (Apr 2006). https://doi.org/10.1007/s00145-005-0307-3
29. Gerhart, P., Schröder, D., Soni, P., Thyagarajan, S.A.K.: Foundations of adaptor signatures. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part II. LNCS, vol. 14652, pp. 161–189. Springer, Cham (May 2024). https://doi.org/10.1007/978-3-031-58723-8_6
30. Glaeser, N., Maffei, M., Malavolta, G., Moreno-Sanchez, P., Tairi, E., Thyagarajan, S.A.K.: Foundations of coin mixing services. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022. pp. 1259–1273. ACM Press (Nov 2022). https://doi.org/10.1145/3548606.3560637
31. Hafezi, H., Partap, A., Das, S., Bonneau, J.: Fair signature exchange. Cryptology ePrint Archive, Report 2025/059 (2025), https://eprint.iacr.org/2025/059
32. Hofheinz, D., Müller-Quade, J.: Universally composable commitments using random oracles. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 58–76. Springer, Berlin, Heidelberg (Feb 2004). https://doi.org/10.1007/978-3-540-24638-1_4
33. Jackson, D., Cremers, C., Cohn-Gordon, K., Sasse, R.: Seems legit: Automated analysis of subtle attacks on protocols that use signatures. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019. pp. 2165–2180. ACM Press (Nov 2019). https://doi.org/10.1145/3319535.3339813
34. Kajita, K., Ohtake, G., Takagi, T.: Consecutive adaptor signature scheme: From two-party to n-party settings. Cryptology ePrint Archive, Paper 2024/241 (2024), https://eprint.iacr.org/2024/241, https://eprint.iacr.org/2024/241
35. Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: A provably secure proof-of-stake blockchain protocol. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 357–388. Springer, Cham (Aug 2017). https://doi.org/10.1007/978-3-319-63688-7_12
36. Küsters, R., Rausch, D.: A framework for universally composable diffie-hellman key exchange. In: 2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017. pp. 881–900. IEEE Computer Society (2017). https://doi.org/10.1109/SP.2017.63, https://doi.org/10.1109/SP.2017.63
37. Küsters, R., Tuengerthal, M.: Joint state theorems for public-key encryption and digital signature functionalities with local computation. In: Sabelfeld, A. (ed.) CSF 2008 Computer Security Foundations Symposium. pp. 270–284. IEEE Computer Society Press (2008). https://doi.org/10.1109/CSF.2008.18

38. Küsters, R., Tuengerthal, M.: Universally composable symmetric encryption. In: Mitchell, J.C. (ed.) CSF 2009 Computer Security Foundations Symposium. pp. 293–307. IEEE Computer Society Press (2009). https://doi.org/10.1109/CSF.2009.18
39. Küsters, R., Tuengerthal, M., Rausch, D.: Joint state composition theorems for public-key encryption and digital signature functionalities with local computation. Journal of Cryptology **33**(4), 1585–1658 (Oct 2020). https://doi.org/10.1007/s00145-020-09353-0
40. Küsters, R., Tuengerthal, M., Rausch, D.: The IITM Model: a Simple and Expressive Model for Universal Composability. Journal of Cryptology **33**(4), 1461–1584 (2020). https://doi.org/10.1007/s00145-020-09352-1
41. Madathil, V., Thyagarajan, S.A.K., Vasilopoulos, D., Fournier, L., Malavolta, G., Moreno-Sanchez, P.: Cryptographic oracle-based conditional payments. In: NDSS 2023. The Internet Society (Feb 2023)
42. Malavolta, G., Moreno-Sanchez, P., Schneidewind, C., Kate, A., Maffei, M.: Anonymous multi-hop locks for blockchain scalability and interoperability. In: NDSS 2019. The Internet Society (Feb 2019). https://doi.org/10.14722/ndss.2019.23330
43. Miller, A., Bentov, I., Bakshi, S., Kumaresan, R., McCorry, P.: Sprites and state channels: Payment networks that go faster than lightning. In: Goldberg, I., Moore, T. (eds.) FC 2019. LNCS, vol. 11598, pp. 508–526. Springer, Cham (Feb 2019). https://doi.org/10.1007/978-3-030-32101-7_30
44. Poelstra, A.: Scriptless scripts. Presentation Slides (2017)
45. Qin, X., Pan, S., Mirzaei, A., Sui, Z., Ersoy, O., Sakzad, A., Esgin, M.F., Liu, J.K., Yu, J., Yuen, T.H.: BlindHub: Bitcoin-compatible privacy-preserving payment channel hubs supporting variable amounts. In: 2023 IEEE Symposium on Security and Privacy. pp. 2462–2480. IEEE Computer Society Press (May 2023). https://doi.org/10.1109/SP46215.2023.10179427
46. Rausch, D., Huber, N., Kuesters, R.: Verifiable e-voting with a trustless bulletin board. Cryptology ePrint Archive, Paper 2025/841 (2025), https://eprint.iacr.org/2025/841
47. Renan, F., Kutas, P.: SQIAsignHD: SQIsignHD adaptor signature. Cryptology ePrint Archive, Report 2024/561 (2024), https://eprint.iacr.org/2024/561
48. Rückert, M., Schröder, D.: Security of verifiably encrypted signatures and a construction without random oracles. In: Shacham, H., Waters, B. (eds.) PAIRING 2009. LNCS, vol. 5671, pp. 17–34. Springer, Berlin, Heidelberg (Aug 2009). https://doi.org/10.1007/978-3-642-03298-1_2
49. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: 40th FOCS. pp. 543–553. IEEE Computer Society Press (Oct 1999). https://doi.org/10.1109/SFFCS.1999.814628
50. Severin, B., Hesenius, M., Blum, F., Hettmer, M., Gruhn, V.: Smart money wasting: Analyzing gas cost drivers of ethereum smart contracts. In: 2022 IEEE International Conference on Software Maintenance and Evolution (ICSME). pp. 293–304 (2022). https://doi.org/10.1109/ICSME55016.2022.00034
51. Stern, J., Pointcheval, D., Malone-Lee, J., Smart, N.P.: Flaws in applying proof methodologies to signature schemes. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 93–110. Springer, Berlin, Heidelberg (Aug 2002). https://doi.org/10.1007/3-540-45708-9_7
52. Tairi, E., Moreno-Sanchez, P., Maffei, M.: Post-quantum adaptor signature for privacy-preserving off-chain payments. Cryptology ePrint Archive, Report 2020/1345 (2020), https://eprint.iacr.org/2020/1345
53. Tairi, E., Moreno-Sanchez, P., Schneidewind, C.: LedgerLocks: A security framework for blockchain protocols based on adaptor signatures. In: Meng, W., Jensen, C.D., Cremers, C., Kirda, E. (eds.) ACM CCS 2023. pp. 859–873. ACM Press (Nov 2023). https://doi.org/10.1145/3576915.3623149
54. Thyagarajan, S.A.K., Malavolta, G., Moreno-Sanchez, P.: Universal atomic swaps: Secure exchange of coins across all blockchains. In: 2022 IEEE Symposium on Security and Privacy. pp. 1299–1316. IEEE Computer Society Press (May 2022). https://doi.org/10.1109/SP46214.2022.9833731
55. Vanjani, N., Soni, P., Thyagarajan, S.A.K.: Functional adaptor signatures: Beyond all-or-nothing blockchain-based payments. In: Luo, B., Liao, X., Xu, J., Kirda, E., Lie, D. (eds.) ACM CCS 2024. pp. 1493–1507. ACM Press (Oct 2024). https://doi.org/10.1145/3658644.3690240
56. Wuille, P., Nick, J., Ruffing, T.: Schnorr signatures for secp256k1. Bitcoin Improvement Proposal 340 (2020), https://github.com/bitcoin/bips/blob/master/bip-0340.mediawiki

A Preliminaries

A.1 Hard Relations

A relation $\mathcal{R}$ is a mapping defined as $\mathcal{R} : \mathcal{D}_{\mathsf{S}} \times \mathcal{D}_{\mathsf{W}} \to \{0,1\}$ where $\mathcal{D}_{\mathsf{S}}$ is the space of statements and $\mathcal{D}_{\mathsf{W}}$ is the space of witnesses. Let $Y \in \mathcal{D}_{\mathsf{S}}$ be a statement and $y \in \mathcal{D}_{\mathsf{W}}$ be a witness. $\mathcal{R}$ maps (S, w) to 1 if and

only if w is a witness for the statement S. The relation is hard if, with only the statement S given, it is computationally infeasible to compute a witness w such that the relation is satisfied. For practical reasons, verifying the validity of a witness/statement pair and sampling instances (w, s) of the relation should be computationally easy.

A.2 Digital Signatures

A digital signature scheme $\Sigma = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ consists of three efficient algorithms. The key generation algorithm $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^\lambda)$ outputs a secret/public key pair. The signing algorithm $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, x)$ takes a secret key and a message and produces a signature. The verification algorithm $b \leftarrow \mathsf{Vrfy}(\mathsf{pk}, x, \sigma)$ checks whether σ is a valid signature on x under the public key pk.

Definition 4 (EUF-CMA Security). *A signature scheme Σ is* existentially unforgeable under chosen message attacks (EUF-CMA) *if for every* PPT *adversary $\mathcal{A}$, there exists a negligible function* negl *such that for each $\lambda \in \mathbb{N}$*

$$\Pr\left[\mathsf{SigForge}_{\mathcal{A},\Sigma}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda),$$

where the experiment is defined in Figure 8 and the randomness is taken over the random choices of all probabilistic algorithms.

$\mathsf{SigForge}_{\mathcal{A},\Sigma}(\lambda)$	$\mathcal{O}_{\mathsf{Sign}}(x)$
$\mathcal{Q} := \emptyset$	$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, x)$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^\lambda)$	$\mathcal{Q} = \mathcal{Q} \cup x$
$(x^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}}(\mathsf{pk})$	**return** σ
return $(x^* \notin \mathcal{Q} \wedge \mathsf{Vrfy}(\mathsf{pk}, x^*, \sigma^*))$	

Fig. 8. Existential unforgeability experiment for digital signatures.

Definition 5 (Correctness). *A signature scheme Σ satisfies* correctness *if for all $\lambda \in \mathbb{N}$ and all messages $x \in \{0,1\}^*$,*

$$\Pr\left[\mathsf{Vrfy}(\mathsf{pk}, x, \sigma) = 1 \;\middle|\; \begin{array}{l}(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^\lambda),\\ \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, x) \wedge \\ \sigma \neq \bot\end{array}\right] = 1.$$

A.3 Non-Interactive Zero-Knowledge Proofs

A non-interactive zero-knowledge (NIZK) proof system [21, 26] allows a prover to convince a verifier of the truth of a statement without revealing any additional information and without requiring interaction. In this work, we adopt a definition in the *random oracle model* with a *transparent setup*, following [7, 46].

Definition 6 (NIZK Proof System in the Random Oracle Model). *Let $\mathcal{R} \subseteq \{0,1\}^* \times \{0,1\}^*$ be an NP relation with associated language $\mathcal{L}_\mathcal{R} := \{x \mid \exists y \text{ such that } \mathcal{R}(x, y) = 1\}$. A* non-interactive zero-knowledge proof system $\mathsf{NIZK} = (\mathsf{P}, \mathsf{V})$ *for $\mathcal{R}$ consists of two probabilistic polynomial-time algorithms with access to a random oracle $\mathcal{RO}$:*
$\pi \leftarrow \mathsf{P}(x, y)$: A prover that, given a statement $x \in \mathcal{L}_\mathcal{R}$ and a corresponding witness y, outputs a proof $\pi \in \{0,1\}^*$. The prover may query the random oracle $\mathcal{RO}$.
$b \leftarrow \mathsf{V}(x, \pi)$: A verifier that, given a statement x from an efficiently decidable superset $\Lambda \supseteq \mathcal{L}_\mathcal{R}$ and a candidate proof π, returns a bit $b \in \{0,1\}$. The verifier may also query the random oracle $\mathcal{RO}$ and accepts only if it is convinced that $x \in \mathcal{L}_\mathcal{R}$.

Definition 7 (Zero-Knowledge Simulator [49]). *A simulator* Sim *for a* NIZK*proof system is a PPT algorithm that responds to queries to a random oracle and produces simulated proofs for statements* $x \in \Lambda$*, where* $\Lambda \supseteq \mathcal{L}_{\mathcal{R}}$ *is an efficiently decidable superset of valid statements. The simulator may generate accepting proofs even for statements outside the language, i.e.,* $x \notin \mathcal{L}_{\mathcal{R}}$*. It supports the following two types of queries:*

- $\mathcal{RO}(y)$*: The simulator maintains a list of oracle query/response pairs* (y, r)*. On input* y*, it returns the same* r *as before if the query has already been made. Otherwise, it samples a fresh random* r *from the oracle range, records* (y, r)*, and returns* r*.*
- $\mathsf{Simulate}(x)$*: For any* $x \in \Lambda$*, the simulator returns a proof* π *such that* $\mathsf{V}^{\mathsf{Sim}}(x, \pi) = 1$*, where the verifier uses the simulator's programmed oracle. To ensure validity, the simulator may program the oracle by adding appropriate entries to its internal query/response list.*

We require the following security properties from NIZK proof systems:

- **(Perfect) Completeness:** For all $x \in \mathcal{L}_{\mathcal{R}}$ and corresponding witnesses y, a valid proof is accepted with probability 1:
$$\Pr[\mathsf{V}(x, \pi) = 1 \mid \pi \leftarrow \mathsf{P}(x, y)] = 1.$$
- **Soundness:** For any $x \notin \mathcal{L}_{\mathcal{R}}$, it is computationally infeasible to produce an accepting proof:
$$\Pr[\mathsf{V}(wx, \pi) = 1 \mid \pi \leftarrow \mathcal{A}(x)] \leq \mathsf{negl}(\lambda).$$
- **Zero-Knowledge:** There exists a simulator Sim (with programming access to $\mathcal{RO}$) such that simulated proofs are computationally indistinguishable from real ones:
$$\left|\Pr\left[\mathcal{A}^{\mathsf{Sim}(x)} = 1\right] - \Pr\left[\mathcal{A}^{\mathsf{V}(x,\pi)} = 1\right]\right| \leq \mathsf{negl}(\lambda),$$
where $\pi \leftarrow \mathsf{P}(x, y)$ and $\mathsf{Sim}(x)$ denotes a simulated proof generated without y.
- **Simulation-Sound Extractability [7]:** If NIZK is zero-knowledge with respect to a simulator Sim, and there exists a PPT extractor Ext such that for any efficient adversary $\mathcal{A}$, the following game is won by Ext with overwhelming probability, then we say that NIZK satisfies *simulation-sound extractability.*
  1. *Initial run:* The game samples a random tape ω and runs $\mathcal{A}(\omega)$ against the simulator Sim, which answers random oracle queries and simulation requests. The adversary outputs a list of statement/proof pairs $\{(x_i, \pi_i)\}_i$. If any (x_i, π_i) fails verification or π_i was generated by a previous $\mathsf{Simulate}(x_i)$ query, the extractor wins trivially.
  2. *Extraction phase:* The extractor Ext receives the transcript of the initial run and the adversary's output $\{(x_i, \pi_i)\}_i$. It may issue Invoke queries, which re-run $\mathcal{A}$ with the same randomness ω. All queries made during these invocations are forwarded to Ext.
  3. *Winning:* The extractor succeeds if it outputs witnesses $\{w_i\}_i$ such that for all i, $(x_i, w_i) \in \mathcal{R}$.

A.4 Schnorr-Style Proofs of Knowledge

We instantiate our proofs using sigma protocols for the discrete logarithm (DLog) relation, following Camenisch and Stadler [13], who proved completeness, honest-verifier zero-knowledge, and special soundness for this class. When combined with the Fiat–Shamir transform, these interactive protocols become non-interactive, retain their zero-knowledge property [13], and are extractable using the forking lemma [6,7]. For straight-line extractability, the Fischlin transform [26] can be applied instead of the Fiat-Shamir transform.

A.5 Schnorr Adaptor Signature Construction

Construction 2 (Schnorr Adaptor Signatures) *Let* $\Sigma = (\mathsf{Sign}, \mathsf{Vrfy})$ *be the Schnorr signature scheme defined over a group* $\mathbb{G}$ *of prime order* p *with generator* g*. Let* $\mathcal{R} = \{(Y, y) \mid Y = g^y\}$ *be the discrete logarithm relation. We define the adaptor signature scheme* $\mathsf{AS} = (\mathsf{pSign}, \mathsf{Adapt}, \mathsf{pVrfy}, \mathsf{Extract})$ *in Figure 9, which extends Schnorr signatures to embed a witness* y *for a statement* $Y \in \mathbb{G}$*.*

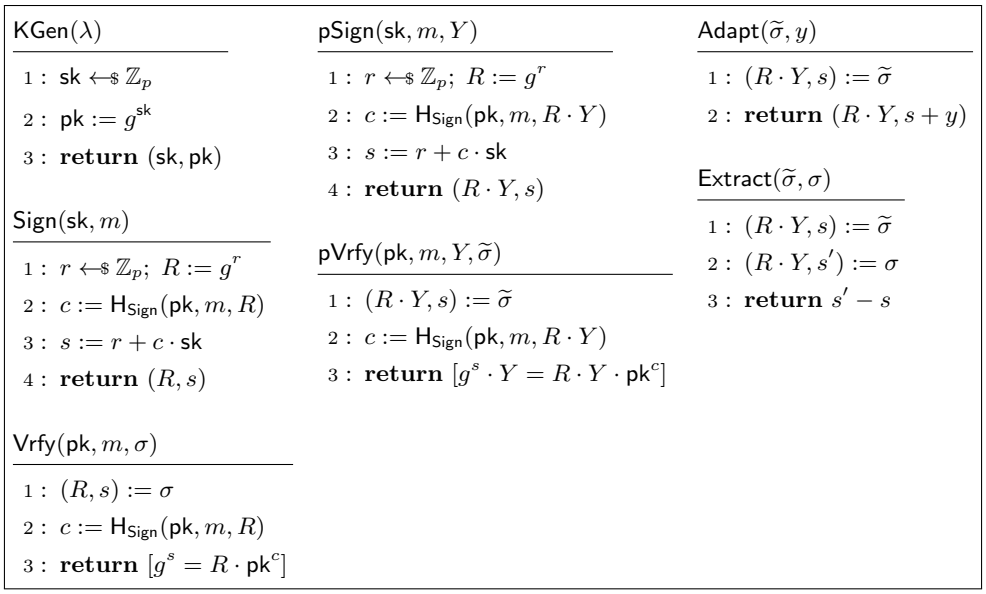

Fig. 9. Schnorr Adaptor Signature Construction for the DLog relation.

A.6 Security of Adaptor Signatures

Correctness. The correctness of adaptor signatures is captured by two properties: *pre-signature correctness* and *pre-signature adaptability*. Previous work defined pre-signature correctness as a combined property encompassing three guarantees. First, classical correctness: any honestly generated pre-signature should successfully pass pre-verification. Second, completeness of adaptation: such a pre-signature, when combined with a valid witness, should adapt to a valid full signature. Third, completeness of extraction: given a valid pre-signature and a corresponding adapted signature, the extract algorithm should return a valid witness for the associated statement. In our treatment, we explicitly split correctness into two distinct properties.

As first property, we assume pre-sign correctness. An adaptor signature is correct if, for an honestly generated keypair, pre-signing any message–statement pair results in a valid pre-signature w.r.t. this message–statement pair. This should hold for all bistrings of statements, as long as such a statement is a valid input for the pre-signing algorithm.

Definition 8 (Pre-Sign Correctness). *An adaptor signature scheme* AS *satisfies* pre-sign correctness *if for all* $\lambda \in \mathbb{N}$ *and all messages* $x \in \{0,1\}^*$ *and all* $Y \in \{0,1\}^*$,

$$\Pr\left[\mathsf{pVrfy}(\mathsf{pk}, x, Y, \widetilde{\sigma}) = 1 \;\middle|\; \begin{array}{l}(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{KGen}(1^\lambda),\\ \widetilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, x, Y) \wedge\\ \widetilde{\sigma} \neq \bot\end{array}\right] = 1.$$

As a second property, we assume extract correctness. Extract correctness enforces that when a pre-signature w.r.t. some statement Y is adapted using a valid witness, the pre-signature–adapted-pre-signature pair allows extracting a valid witness for Y.

Definition 9 (Extract Correctness). *An adaptor signature scheme* AS *satisfies* extract correctness *if for all messages* $x \in \{0,1\}^*$*, all* $(Y, y) \in \mathcal{R}$*, and all public keys* pk *and pre-signatures* $\widetilde{\sigma} \in \{0,1\}^*$,

$$\Pr\left[(Y, y') \in \mathcal{R} \;\middle|\; \begin{array}{l}\mathsf{pVrfy}(\mathsf{pk}, x, Y, \widetilde{\sigma}) = 1 \wedge\\ \sigma := \mathsf{Adapt}(\mathsf{pk}, \widetilde{\sigma}, y, Y) \wedge\\ y' := \mathsf{Extract}(\mathsf{pk}, \widetilde{\sigma}, \sigma, Y)\end{array}\right] = 1.$$

While our two-phase correctness notion is technically stronger than the established one as extract correctness is required for all signatures that verify rather than only those returned by the signing algorithm, it better reflects the intuitive design goal of a correct adaptor signature scheme and is captured by most adaptor signature schemes where the extract algorithm is essentially the inverse function of the adapt algorithm. Compared to the correctness definition of [29], we additionally relax the original restriction to fixed-length messages and now require this property for arbitrary messages (in both our correctness definitions). This change does not actually impose stronger requirements but is rather a generalization: any scheme that is correct for messages of a specific fixed length can be interpreted as a scheme that is correct on $\{0,1\}^*$ by having (pre-)sign and (pre-)verify algorithms reject inputs when messages do not adhere to the original fixed length. The main reason for this modification is that it aligns more closely with UC security, which requires properties to hold true for all possible inputs that the environment might provide. We show in Section 6.1 that our Schnorr adaptor signature scheme satisfies this two-part definition.

Pre-Signature Adaptability. Pre-signature adaptability guarantees that if a pre-signature is valid w.r.t. an honestly generated statement, then adapting such a pre-signature with a valid witness for the statement yields a valid signature. We also relax the restriction for fixed-length messages, due to the same reasons as for correctness.

Definition 10 (Pre-signature adaptability). *An adaptor signature scheme* AS *satisfies* pre-signature adaptability *if for all* $\lambda \in \mathbb{N}$*, all messages* $x \in \{0,1\}^*$*, all* $(Y, y) \in \mathcal{R}$*, and all public keys* pk *and pre-signatures* $\widetilde{\sigma} \in \{0,1\}^*$*, it holds that if* $\mathsf{pVrfy}(\mathsf{pk}, x, Y, \widetilde{\sigma}) = 1$*, then* $\mathsf{Vrfy}(\mathsf{pk}, x, \mathsf{Adapt}(\mathsf{pk}, \widetilde{\sigma}, y, Y)) = 1$.

Extractability. Extractability extends the notion of existential unforgeability from standard signature schemes to the adaptor signature setting. As in unforgeability, the adversary wins if it can produce a valid signature on a message that was not previously signed by the signing oracle. However, in the adaptor setting, the adversary is also granted access to a pre-signing oracle for any message–statement pair, as well as an oracle that samples fresh statements from the hard relation. The winning condition is strengthened: the adversary must produce a signature on a fresh message that not only verifies, but also cannot be used—together with any pre-signature obtained from the pre-signing oracle—to extract a valid witness for the associated statement.

Definition 11 (Extractability). *An adaptor signature scheme* AS *satisfies* extractability *if for every* PPT *adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that for every* $\lambda \in \mathbb{N}$*,*

$$\Pr\left[\mathsf{Ext}_{\mathcal{A},\mathsf{AS}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda) \ ,$$

where the experiment $\mathsf{Ext}_{\mathcal{A},\mathsf{AS}}$ *is defined in Figure 10, and the probability is taken over the randomness of all probabilistic algorithms.*

Unique Extractability. Unique extractability guarantees that any valid pre-signature acts as a commitment to both a *single* valid signature and a *single* witness. That is, no efficient adversary can produce a pre-signature $\widetilde{\sigma}$ for a message x and statement Y such that there exist two distinct signatures on x that both allow extraction of valid witnesses (with respect to Y) from the same pre-signature. More formally:

Definition 12 (Unique Extractability). *An adaptor signature scheme* AS *satisfies* unique extractability *if for every* PPT *adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that for every* $\lambda \in \mathbb{N}$*,*

$$\Pr\left[\mathsf{UniqueExtractability}_{\mathcal{A},\mathsf{AS}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda) \ ,$$

where the experiment $\mathsf{UniqueExtractability}_{\mathcal{A},\mathsf{AS}}$ *is defined in Figure 11, and the probability is taken over the randomness of all probabilistic algorithms.*

As discussed in the technical outline, we omit the privacy notion of unlinkability and the notion of pre-verify soundness in this work.

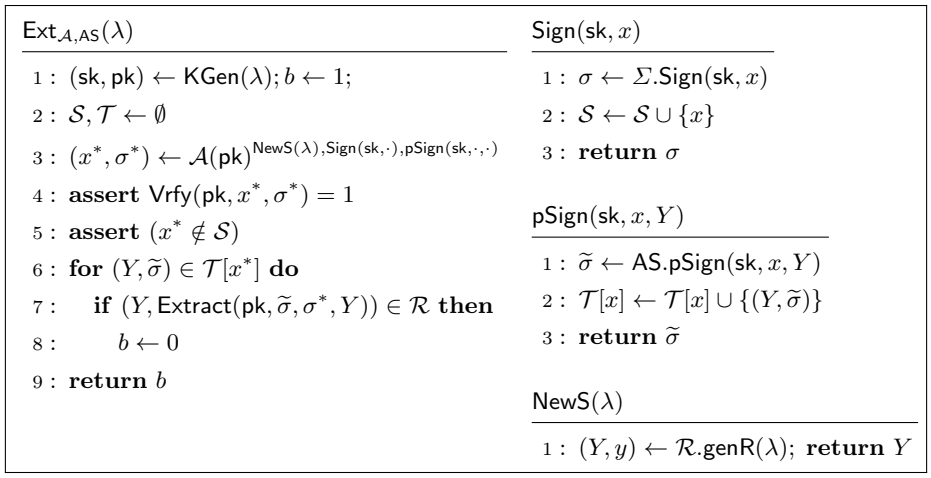

Fig. 10. The extractability experiment $\mathsf{Ext}_{\mathcal{A},\mathsf{AS}}(\lambda)$.

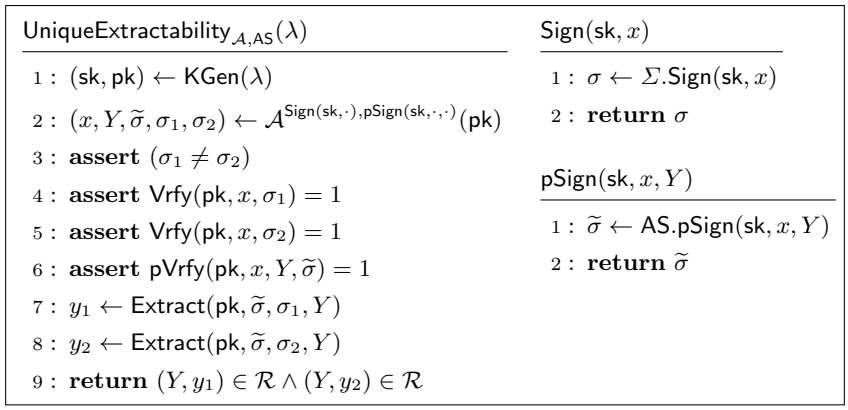

Fig. 11. The unique extractability experiment $\mathsf{UniqueExtractability}_{\mathcal{A},\mathsf{AS}}(\lambda)$.

Former correctness notions.

Definition 13 (Pre-signature correctness). *An adaptor signature scheme* AS *satisfies* pre-signature correctness *if for all* $\lambda \in \mathbb{N}$ *and all messages* $x \in \{0,1\}^{\ell_m}$,

$$\Pr\left[\begin{array}{l|l} \mathsf{pVrfy}(\mathsf{pk}, x, Y, \widetilde{\sigma}) = 1 \wedge & (\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{KGen}(1^\lambda), \\ \mathsf{Vrfy}(\mathsf{pk}, x, \sigma) = 1 \wedge & (Y, y) \leftarrow \mathsf{genR}(1^\lambda), \\ (Y, y') \in \mathcal{R} & \widetilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, x, Y), \\ & \sigma := \mathsf{Adapt}(\mathsf{pk}, \widetilde{\sigma}, y), \\ & y' := \mathsf{Extract}(\mathsf{pk}, \widetilde{\sigma}, \sigma, Y) \end{array}\right] = 1.$$

B Proofs

Proof (Proof of Theorem 3). To prove Theorem 3, we proceed in two parts. First, we show that correctness, extractability, unique extractability, pre-signature adaptability, and message-bound pre-signatures of the enhanced Schnorr adaptor signature scheme reduce to the corresponding security properties of the original Schnorr adaptor signature construction and the completeness of the proof system. Second, we prove that the enhanced scheme additionally satisfies pre-signature unforgeability, a property not achieved by the original

construction. Note that the PoK scheme of Figure 7 is simulation-sound extractable [7] and zero-knowledge in the random oracle model (c.f. Appendix A).

Correctness. For correctness, we show both pre-sign correctness and extract correctness. For pre-sign correctness, we have to show that honest pre-signing on any statement and any message leads to a valid pre-signature, if the pre-signing algorithm does not fail. If the bitstring of the statement is parseable as a group element, and the message is of length ℓ_M, then this follows from the correctness of the standard Schnorr adaptor signature scheme and the (perfect) completeness of the NIZK proof. If not, the underlying Schnorr pre-signing algorithm will fail, and correctness holds trivially.

To establish extract correctness, we must show that for any valid pre-signature (σ_1, σ_2) on a message x with respect to a statement Y, and any valid witness y for Y, the extract algorithm correctly recovers y when given the adapted signature obtained via adapt. In our construction, adapt and extract are inverse operations. Adapting yields $(\sigma_1, \sigma_2 + y)$, which is well-defined since σ_2 and y lie in the same cyclic group. Applying extract then computes $(\sigma_2 + y) - \sigma_2 = y$. Thus, the extracted witness is identical to the one used for adapting and is valid for Y.

Extractability. We show that if extractability is violated in the enhanced Schnorr adaptor signature scheme, then it can also be broken in the underlying Schnorr scheme. To this end, we construct a reduction that uses an adversary $\mathcal{A}$ against extractability in the enhanced scheme to break extractability in the standard scheme.

The reduction proceeds by simulating the extractability experiment for the enhanced scheme using access to the (pre-)signing oracles of the extractability experiment for the standard scheme. The reduction forwards the key provided by the extractability experiment to the adversary. Whenever $\mathcal{A}$ issues a signing or pre-signing query, we forward this query to the standard (pre-)signing oracle to obtain a valid pre-signature $\widetilde{\sigma}$. We then use the zero-knowledge simulator of the proof of knowledge system to generate a simulated proof of knowledge for $\widetilde{\sigma}$, resulting in a valid pre-signature under the enhanced scheme. Since the proof system is zero-knowledge, this simulation is indistinguishable from a real proof with overwhelming probability.

Eventually, $\mathcal{A}$ outputs a message–signature pair (x, σ) such that σ is a valid Schnorr signature and no pre-signature was queried that allows extraction of the witness. By construction, σ contains a valid signature in the standard Schnorr adaptor signature scheme. Moreover, if the forgery is valid in the enhanced scheme, then x was never signed, and no corresponding pre-signature that extracts was queried in the enhanced scheme. Since any signature–pre-signature pair that extracts in the standard scheme also extracts in the enhanced scheme, the forgery against the enhanced scheme constitutes a valid forgery against extractability for the standard scheme. Our reduction is efficient, as the simulator and oracles are efficient. Hence, extractability in the enhanced Schnorr scheme reduces to extractability in the standard scheme.

Unique Extractability. We show that if unique extractability is violated in the enhanced Schnorr adaptor signature scheme, then it can also be broken in the underlying standard Schnorr adaptor signature scheme. To this end, we construct a reduction that uses an adversary $\mathcal{A}$ against unique extractability in the enhanced scheme to break unique extractability in the standard scheme.

The reduction begins by forwarding the public key provided by the unique extractability experiment for the standard scheme to $\mathcal{A}$. It then simulates the enhanced experiment using access to the (pre-)signing oracles of the standard scheme. Whenever $\mathcal{A}$ issues a signing or pre-signing query, we forward this query to the standard (pre-)signing oracle to obtain a valid pre-signature $\widetilde{\sigma}$. We then use the zero-knowledge simulator of the proof of knowledge system to generate a simulated proof of knowledge for $\widetilde{\sigma}$, producing a valid enhanced pre-signature. Since the proof system is zero-knowledge, this simulation is indistinguishable from a real proof with overwhelming probability.

Eventually, $\mathcal{A}$ outputs a tuple $(x, \widetilde{\sigma}, Y, \sigma_1, \sigma_2)$ such that all (pre-)signatures are valid, and both σ_1 and σ_2 allow extracting valid witnesses for Y using $\widetilde{\sigma}$. This constitutes a violation of unique extractability in the enhanced scheme. By construction, all elements of this tuple—x, $\widetilde{\sigma}$, and σ_i—are also valid in the standard scheme (when ignoring the PoK). Moreover, since the extraction relation remains unchanged between the enhanced and standard schemes, the same pair $(\widetilde{\sigma}, \sigma_i)$ will extract to the same witness in both schemes.

Hence, this tuple also constitutes a valid forgery against unique extractability in the standard Schnorr adaptor signature scheme. The reduction is efficient, as the simulation uses only efficient oracle access and a zero-knowledge simulator. Therefore, unique extractability in the enhanced Schnorr scheme reduces to unique extractability in the standard scheme.

Pre-Signature Adaptability. The same proof carries over to adaptability, since every valid pre-signature in the enhanced scheme contains a valid pre-signature in the standard scheme. In addition, we need to consider that we enhanced adaptability with respect to arbitrary-length messages. This transition works with the same argument as for correctness. So, by pre-signature adaptability of the standard scheme, this can be adapted into a valid signature. Signatures valid under the standard scheme are also valid in the enhanced scheme, since we did not modify the verification algorithm. So, also in the enhanced scheme, this signature is valid, and pre-signature adaptability holds.

Message-Bound Pre-Signatures. Standard Schnorr adaptor signatures have message-bound pre-signatures, since the random oracle is collision resistant: The verification equation for a pre-signature (R', s) on a statement Y has the form

$$Y \cdot g^s = \mathsf{pk}^c \cdot R'$$

with $c = \mathsf{H}_{\mathsf{Sign}}(\mathsf{pk}, x, R')$. For a distinct message $x' \neq x$, we have that $c' = \mathsf{H}_{\mathsf{Sign}}(\mathsf{pk}, x', R') \neq c$ (modulo group order and with overwhelming probability). If $\mathsf{pk} \neq g^0$, then pk is a generator by primality of the group order such that $c' \neq c$ implies $\mathsf{pk}^{c'} \neq \mathsf{pk}^c$ and hence the verification equation does not hold for x'. Since the public key is chosen uniformly random over the group $\mathbb{G}$, the probability of being the neutral element g^0 is negligible. Observe that each pre-signature in our extended Schnorr adaptor signature scheme also contains a standard Schnorr adaptor pre-signature. Furthermore, pre-verification of the extended scheme requires validity of the standard Schnott adaptor signature. Hence, the same argument also shows message-boundedness of our extended scheme.

Pre-signature Unforgeability. Finally, we prove that the enhanced Schnorr adaptor signature scheme satisfies pre-signature unforgeability. We do this by a reduction to the simulation-sound extractability of the PoK and a reduction to the extractability of the standard Schnorr adaptor signature scheme. For this proof, simulation-sound extractability is necessary, since to win pre-signature unforgeability, the adversary is not required to output a valid signature, but just a pre-signature. However, if the adversary outputs a valid pre-signature w.r.t. some statement Y, the reduction might not know a valid witness w.r.t. Y, and hence, cannot adapt such a forgery into a valid signature. However, such a valid signature forgery would be needed to break (adaptor signature) extractability. Since this is not a valid proof strategy, we instead consider a different approach: As above, we build a reduction to the (adaptor signature) extractability and simulate proofs for pre-signatures using the zero-knowledge simulator. When the adversary outputs a valid pre-signature as forgery, this pre-signature contains a proof of knowledge of randomness. Using the extractor of the PoK, we extract this randomness, and by the algebraic structure, this allows us to learn the signing key and break extractability.

We now start our reduction to the extractability of standard Schnorr signatures. As input, the reduction receives a public key pk and has access to pre-sign and sign oracles. Our reduction forwards the public key to $\mathcal{A}$ and simulates signing queries by forwarding them to the underlying signature oracle. Pre-signing queries are handled by obtaining a standard pre-signature and then simulating a PoK using the zero-knowledge simulator. By the simulation-sound extractability of the PoK, this simulation is perfect with overwhelming probability. Eventually, $\mathcal{A}$ outputs a fresh pre-signature on a message x that was never pre-signed. At this point, we try to run the extractor of the PoK system to obtain the underlying randomness r.[4] Now, we

[4] We note that the extractor is allowed to and typically has to rewind the adversary. A reader familiar with UC, where simulators cannot use rewinding, might wonder why this is possible without contradicting our UC security proof, i.e., Theorem 2. The reason is that our UC simulator does not actually have to run this extractor and hence does not have to perform any rewinding. Instead, we use this technique only as part of a contradiction argument to show that certain runs that would break the UC simulation have a negligible chance of occurring.

have two cases: Either the extractor succeeds in extracting the randomness. If this is the case, and given this random value r and the forged pre-signature, we can extract the signing key sk, since the pre-signature contains the value $\mathsf{sk} \cdot h + r$ for a known h. Knowing sk, our reduction can break extractability for the Schnorr scheme, as it can now compute signatures on fresh messages. In the second case, where the extractor does not succeed, we use the power of $\mathcal{A}$ to break the simulation-sound extractability of the PoK system. Contained in the forgery, the adversary has output PoK for which the witness cannot be extracted; we forward the PoK to the extractor experiment for the proof system. This violates the simulation-sound extractability of the PoK system by definition. In both cases, the reduction is efficient and succeeds with non-negligible probability if $\mathcal{A}$ does.

Proof (Proof of Theorem 2). To prove Theorem 2, we have to show the existence of a simulator $\mathcal{S}$ such that the real world protocol that just runs $\mathsf{AS}_{\Sigma,\mathcal{R}}$ (in the presence of the dummy adversary that simply forwards network messages) is indistinguishable from the ideal functionality $\mathcal{F}_{\text{asig}}$ running with the simulator $\mathcal{S}$ for all ppt environments. As is usual for UC security, we only have to show this property for a single session/copy of the protocol (identified by an arbitrary but fixed challenge session ID *sid*) and hence only for one key pair that, however, may be used by an arbitrary number of parties. A general composition theorem then implies that UC security is retained even when an environment uses polynomially many different copies/sessions and hence key pairs (see, e.g., Theorem 3 in [40], which shows this as an explicit and distinct property, or the definition of the UC security experiment and Theorem 22 given in [15], which captures this implicitly along with other properties).

We show this theorem for $\mathcal{S}$ being the dummy adversary, i.e., the adversary in real and ideal world being identical. We thus only have to show that $\mathsf{AS}_{\Sigma,\mathcal{R}}$ and $\mathcal{F}_{\text{asig}}$ are indistinguishable for all environments without having to modify the adversary as part of the proof. The proof itself is structured as a sequence of games that start from the real protocol $\mathsf{AS}_{\Sigma,\mathcal{R}}$ and iteratively modify it until we reach $\mathcal{F}_{\text{asig}}$. We reduce each game hop to one or more of the game-based properties required by Theorem 2 to show that the distinguishing probability for a ppt environment changes only by a negligible probability. Since the number of game hops is constant, this immediately gives the claim, i.e., that the real protocol is indistinguishable from the ideal one.

As mentioned, our starting point is the real protocol that just runs $\mathsf{AS}_{\Sigma,\mathcal{R}}$ as is, which we formalize via the protocol $\mathcal{F}^0_{\text{asig}}$ given in Figure 12. This is basically the same as Figure 2 that we used as a starting point for our technical overview given in Section 4.2. The protocol $\mathcal{F}^0_{\text{asig}}$ is parameterized by a polytime decidable relation $\mathcal{R}$, ppt algorithms $\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{pSign}$, and dpt algorithms $\mathsf{Vrfy}, \mathsf{pVrfy}, \mathsf{Adapt}, \mathsf{Extract}$. When an oracle of the protocol is called, the underlying corresponding algorithm (indicated by the same name) is executed and the result is returned. Since only the owner of the key pair knows the secret key, only he can call the Sign and pSign oracles.

Game $\mathcal{G}_1$. In the first game, we start by modifying our real protocol to obtain a first intermediate ideal functionality that enforces correctness properties in the verification, pre-verification, and extract oracles. To this extent, our functionality deploys bookkeeping. This means, when a signature or pre-signature is successfully created, the functionality stores the corresponding (pre-)signatures into lists $\mathsf{H}_{\mathsf{sign}}$ and $\mathsf{H}_{\mathsf{pSign}}$. When a signature or pre-signature is verified, the functionality accepts all signatures from $\mathsf{H}_{\mathsf{sign}}$ and all pre-signatures from $\mathsf{H}_{\mathsf{pSign}}$. In addition, the functionality stores the results of successfully (pre-)verifying (pre-)signatures in $\mathsf{H}_{\mathsf{verify}}$ and $\mathsf{H}_{\mathsf{pVerify}}$. As an additional list, we introduce the list $\mathsf{H}_{\mathsf{adapt}}$. In $\mathsf{H}_{\mathsf{adapt}}$, we store the inputs and the output of the adapt oracle, i.e., the pairs $(\widetilde{\sigma}, Y, y, \sigma)$, when the pre-siganture–statement pair is stored in $\mathsf{H}_{\mathsf{pVerify}}$, the witness y is a valid witness for Y, and the signature σ is the ouptut of calling the adapt algorithm on $(\widetilde{\sigma}, Y, y)$. Looking ahead, this list will allow us to enforce extract correctness, as well as pre-signature adaptability. Besides this, the functionality $\mathcal{F}^1_{\text{asig}}$ has the same outputs as $\mathcal{F}^0_{\text{asig}}$. We depict $\mathcal{F}^1_{\text{asig}}$ in Figure 13.

Claim 1 *If AS is pre-sign correct (Definition 8), has correct adapting (Definition 9), and Σ is a correct signature scheme (Definition 5), then the gap between $\mathcal{F}^1_{asig}$ and $\mathcal{F}^0_{asig}$ is negligible.*

Proof. To show Claim 1, we note that $\mathcal{F}^1_{\text{asig}}$ and $\mathcal{F}^0_{\text{asig}}$ differ in three cases. The first case is, if a signature on a message x was obtained via $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, x)$ by an oracle, and $\mathsf{Vrfy}(\mathsf{pk}, x, \sigma) = 0$. As the signature scheme Σ is

The functionality is parameterized by a polytime decidable relation $\mathcal{R}$, ppt algorithms $\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{pSign}$ and dpt algorithms $\mathsf{Vrfy}, \mathsf{pVrfy}, \mathsf{Adapt}, \mathsf{Extract}$. Upon first activation, initialize this instance by running and storing $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.

On $\mathtt{PubKey}$? from anyone, return $(\mathtt{PubKey}, \mathsf{pk})$.

On $(\mathtt{Sign}, x)$ for party pid in session $sid = (pid, sid')$, do:
$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, x)$
Return $(\mathtt{Signature}, \sigma)$.

On $(\mathtt{pSign}, x, Y)$ for party pid in session $sid = (pid, sid')$, do:
$\widetilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, x, Y)$
Return $(\mathtt{PreSignature}, \widetilde{\sigma})$

On $(\mathtt{Adapt}, pk, \widetilde{\sigma}, y, Y)$ for party pid, do:
$\sigma \leftarrow \mathsf{Adapt}(pk, \widetilde{\sigma}, y, Y)$
Return $(\mathtt{Adapt}, \sigma)$

On $(\mathtt{Extract}, pk, \widetilde{\sigma}, \sigma, Y)$ for party pid, do:
$y := \mathsf{Extract}(pk, \widetilde{\sigma}, \sigma, Y)$
Return $(\mathtt{Extract}, y)$

On $(\mathtt{pVerify}, pk, x, Y, \widetilde{\sigma})$ for party pid, do:
$b_{\mathsf{pVerify}} := \mathsf{pVrfy}(pk, x, Y, \widetilde{\sigma})$
Return $(\mathtt{VerResult}, b_{\mathsf{pVerify}})$

On $(\mathtt{Verify}, pk, x, \sigma)$ for party pid, do:
$b_{\mathsf{verify}} := \mathsf{Vrfy}(pk, x, \sigma)$
Return $(\mathtt{VerResult}, b_{\mathsf{verify}})$

As usual, the attacker may corrupt any party pid dynamically and gains full control over that party. In addition, if pid is the owner of the key pair modeled by this instance, i.e., the session ID of this instance is $sid = (pid, sid')$, then the attacker also learns sk; for notational convenience, we set $\mathsf{corrupted} = \mathsf{true}$ iff this is the case.

Fig. 12. Ideal adoptor signatures functionality $\mathcal{F}_{\text{asig}}^0$.

The functionality is parameterized by a polytime decidable relation $\mathcal{R}$, ppt algorithms $\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{pSign}$ and dpt algorithms $\mathsf{Vrfy}, \mathsf{pVrfy}, \mathsf{Adapt}, \mathsf{Extract}$. Upon first activation, initialize this instance by running and storing $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.

On $\texttt{PubKey}$? from anyone, return $(\texttt{PubKey}, \mathsf{pk})$.

On $(\texttt{Sign}, x)$ for party pid in session $sid = (pid, sid')$, do:
- $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, x)$
- **if** $\sigma \neq \bot$: **add** σ **to** $\mathsf{H}_{\mathsf{sign}}[x]$
- Return $(\texttt{Signature}, \sigma)$.

On $(\texttt{pSign}, x, Y)$ for party pid in session $sid = (pid, sid')$, do:
- $\widetilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, x, Y)$
- **if** $\widetilde{\sigma} \neq \bot$: **add** $(\widetilde{\sigma}, Y)$ **to** $\mathsf{H}_{\mathsf{pSign}}[x]$
- Return $(\texttt{PreSignature}, \widetilde{\sigma})$

On $(\texttt{Adapt}, pk, \widetilde{\sigma}, y, Y)$ for party pid, do:
- $\sigma \leftarrow \mathsf{Adapt}(pk, \widetilde{\sigma}, y, Y)$
- **if** $pk = \mathsf{pk} \wedge (Y, y) \in \mathcal{R} \wedge \exists x : (\widetilde{\sigma}, Y) \in \mathsf{H}_{\mathsf{pVerify}}[x]$:
  - **add** $(\widetilde{\sigma}, Y, y, \sigma)$ **to** $\mathsf{H}_{\mathsf{adapt}}[x]$
- Return $(\texttt{Adapt}, \sigma)$

On $(\texttt{Extract}, pk, \widetilde{\sigma}, \sigma, Y)$ for party pid, do:
- $y := \mathsf{Extract}(pk, \widetilde{\sigma}, \sigma, Y)$
- **if** $(Y, y) \notin \mathcal{R} \wedge \exists (x, y') : (\widetilde{\sigma}, Y, y', \sigma) \in \mathsf{H}_{\mathsf{adapt}}[x]$: $y := y'$ {*Extract Correctness*
- Return $(\texttt{Extract}, y)$

On $(\texttt{pVerify}, pk, x, Y, \widetilde{\sigma})$ for party pid, do:
- $b_{\mathsf{pVerify}} := \mathsf{pVrfy}(pk, x, Y, \widetilde{\sigma})$
- **if** $pk = \mathsf{pk} \wedge (\widetilde{\sigma}, Y) \in \mathsf{H}_{\mathsf{pSign}}[x]$: $b_{\mathsf{pVerify}} := 1$ {*Pre-Sign Correctness*
- **if** $pk = \mathsf{pk} \wedge b_{\mathsf{pVerify}}$: **add** $(\widetilde{\sigma}, Y)$ **to** $\mathsf{H}_{\mathsf{pVerify}}[x]$
- Return $(\texttt{VerResult}, b_{\mathsf{pVerify}})$

On $(\texttt{Verify}, pk, x, \sigma)$ for party pid, do:
- $b_{\mathsf{verify}} := \mathsf{Vrfy}(pk, x, \sigma)$
- **if** $pk = \mathsf{pk} \wedge (x, \sigma) \in \mathsf{H}_{\mathsf{sign}}$: $b_{\mathsf{verify}} := 1$ {*Correctness*
- **if** $pk = \mathsf{pk} \wedge b_{\mathsf{verify}}$: **add** σ **to** $\mathsf{H}_{\mathsf{verify}}[x]$
- Return $(\texttt{VerResult}, b_{\mathsf{verify}})$

As usual, the attacker may corrupt any party pid dynamically and gains full control over that party. In addition, if pid is the owner of the key pair modeled by this instance, i.e., the session ID of this instance is $sid = (pid, sid')$, then the attacker also learns sk; for notational convenience, we set $\mathsf{corrupted} = \mathsf{true}$ iff this is the case.

Fig. 13. Ideal adoptor signatures functionality $\mathcal{F}_{\mathrm{asig}}^1$.

(perfectly) correct, $\mathsf{Vrfy}(\mathsf{pk}, x, \mathsf{Sign}(\mathsf{sk}, x)) = 0$ cannot happen. The second case is equivalent to the first, but when pre-verifying pre-signatures output by the pre-sign oracle. This second change is again undetectable by the perfect pre-sign correctness of the adaptor signature scheme. In the third case, the functionalities differ if the extract algorithm outputs bot on a pre-signature–signature–statement pair, but this pair is added to $\mathsf{H}_{\mathsf{adapt}}$. A pre-signature–signature–statement pair is added to $\mathsf{H}_{\mathsf{adapt}}$ only if the pre-signature–statement pair was previously stored in $\mathsf{H}_{\mathsf{pVerify}}$. This occurs only if the pre-signature verifies (for that statement and some message x) under the pVrfy algorithm or if it was stored (for that statement and some message x) in $\mathsf{H}_{\mathsf{pSign}}$. By pre-sign correctness, all entries in $\mathsf{H}_{\mathsf{pSign}}$ also verify under the pVrfy algorithm. Altogether we have that in all cases where an element is added to $\mathsf{H}_{\mathsf{adapt}}$ the pre-signature for the statement is valid under the pVrfy algorithm (for some message x). In addition to such a valid pre-signature, adding a pair to $\mathsf{H}_{\mathsf{adapt}}$ also requires providing a valid witness for the statement. Only if a valid pre-signature–statement pair and a valid witness for this statement are provided, we call the adapt algorithm on this input and store the valid pre-signature, the statement, the valid witness, and the resulting adapted pre-signature in $\mathsf{H}_{\mathsf{adapt}}$. When the functionality overwrites a witness, the extract algorithm failed to provide a valid witness for a valid pre-signature, statement, signature pair. If this case occurs, it would thus contradict extract correctness. Since extract correctness is a perfect property, this case cannot happen.

Hence, the gap between $\mathcal{F}^1_{\text{asig}}$ and $\mathcal{F}^0_{\text{asig}}$ is bounded by correctness, pre-sign correctness, and adapting correctness. Since these are perfect properties, this gap is 0.

At this point in our series of game hops, whenever an oracle considers a pre-signature or signature valid w.r.t. some message (and some statement), then the actual algorithms output that the (pre-)signature is valid. Hence, all signatures in the sets $\mathsf{H}_{\mathsf{sign}}[x], \mathsf{H}_{\mathsf{verify}}[x]$ will verify w.r.t. pk and x, and all pre-sigature–statement pairs $(\widetilde{\sigma}, Y)$ in the sets $\mathsf{H}_{\mathsf{pSign}}[x], \mathsf{H}_{\mathsf{pVerify}}[x], \mathsf{H}_{\mathsf{adapt}}[x]$ will pre-verify w.r.t. x. This holds, since from this point forward, we will not add more elements to these lists and assuming correctness and pre-sign correctness, which are perfect properties.

Game $\mathcal{G}_2$. In this game, the functionality enforces message-bound pre-signatures if the adversary did not corrupt the signer. This means, the functionality enforces enforce that each pre-signature can only be valid on a unique message. This ensures unique messages in the list $\mathsf{H}_{\mathsf{adapt}}$ when no corruption happened. Besides this, the functionality $\mathcal{F}^2_{\text{asig}}$ has the same outputs as $\mathcal{F}^1_{\text{asig}}$. We depict $\mathcal{F}^2_{\text{asig}}$ in Figure 14.

Claim 2 *If* AS*, in addition to all the properties required by previous claims, has message-bound pre-signatures (Definition 3), then the gap between* $\mathcal{F}^2_{asig}$ *and* $\mathcal{F}^1_{asig}$ *is negligible.*

Proof. The outputs of the functionalities $\mathcal{F}^2_{\text{asig}}$ and $\mathcal{F}^1_{\text{asig}}$ differ if the signer is uncorrupted, and the pre-signing algorithm pSign, on input a message x, outputs a pre-signature that is also considered valid with respect to a different message x', or if the environment provides a pre-signature that is valid with respect to the public key pk, a message x, and some statement Y, but this pre-signature has already been marked as valid for another message x'. In such cases, $\mathcal{F}^2_{\text{asig}}$ either returns $\perp$ instead of a pre-signature or fails to pre-verify a valid pre-signature.

We now show that the probability of this case occurring is negligible in the security parameter. To this end, we provide a reduction that transforms any environment causing a pre-signature that is valid on two different messages into an adversary that breaks the message-bound pre-signature property of AS.

As input, our reduction receives a public key and has access to a signing and a pre-signing oracle that outputs valid (pre-) signatures w.r.t. the provided public key. To simulate the functionality in the environment, our reduction uses this key and the provided oracles to answer signing and pre-signing queries. Adapting, extracting, and verifying are public algorithms, so our reduction can simulate them. If the simulation comes to a point where the signer is corrupted and the functionality would return the secret key, then the reduction attacker aborts the simulation. Such an abort does not change the distinguishability of $\mathcal{F}^1_{\text{asig}}$ and $\mathcal{F}^2_{\text{asig}}$, since we only enfore message-bound pre-sigantures if the signer is not corrupted, and hence, the functionalities only differ if the signer is not corrupted. So if the signer is corrupted, both functionalities behave identically by definition, and nothing is to show. If no corruption happens, our reduction shows that the gap between the both worlds is at most negligible.

The functionality is parameterized by a polytime decidable relation $\mathcal{R}$, ppt algorithms $\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{pSign}$ and dpt algorithms $\mathsf{Vrfy}, \mathsf{pVrfy}, \mathsf{Adapt}, \mathsf{Extract}$. Upon first activation, initialize this instance by running and storing $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.

On $\texttt{PubKey}$? from anyone, return $(\texttt{PubKey}, \mathsf{pk})$.

On $(\texttt{Sign}, x)$ for party pid in session $sid = (pid, sid')$, do:
- $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, x)$
- **if** $\sigma \neq \bot$: **add** σ **to** $\mathsf{H}_{\mathsf{sign}}[x]$
- Return $(\texttt{Signature}, \sigma)$.

On $(\texttt{pSign}, x, Y)$ for party pid in session $sid = (pid, sid')$, do:
- $\widetilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, x, Y)$
- **if** $\neg\mathsf{corrupted} \wedge \exists x' \neq x : \widetilde{\sigma} \in \mathsf{H}_{\mathsf{pSign}}[x'] \cup \mathsf{H}_{\mathsf{pVerify}}[x']$: $\widetilde{\sigma} := \bot$ {*Message Bounding*
- **if** $\widetilde{\sigma} \neq \bot$: **add** $(\widetilde{\sigma}, Y)$ **to** $\mathsf{H}_{\mathsf{pSign}}[x]$
- Return $(\texttt{PreSignature}, \widetilde{\sigma})$

On $(\texttt{Adapt}, pk, \widetilde{\sigma}, y, Y)$ for party pid, do:
- $\sigma \leftarrow \mathsf{Adapt}(pk, \widetilde{\sigma}, y, Y)$
- **if** $pk = \mathsf{pk} \wedge (Y, y) \in \mathcal{R} \wedge \exists x : (\widetilde{\sigma}, Y) \in \mathsf{H}_{\mathsf{pVerify}}[x]$:
  - **add** $(\widetilde{\sigma}, Y, y, \sigma)$ **to** $\mathsf{H}_{\mathsf{adapt}}[x]$
- Return $(\texttt{Adapt}, \sigma)$

On $(\texttt{Extract}, pk, \widetilde{\sigma}, \sigma, Y)$ for party pid, do:
- $y := \mathsf{Extract}(pk, \widetilde{\sigma}, \sigma, Y)$
- **if** $(Y, y) \notin \mathcal{R} \wedge \exists (x, y') : (\widetilde{\sigma}, Y, y', \sigma) \in \mathsf{H}_{\mathsf{adapt}}[x]$: $y := y'$ {*Extract Correctness*
- Return $(\texttt{Extract}, y)$

On $(\texttt{pVerify}, pk, x, Y, \widetilde{\sigma})$ for party pid, do:
- $b_{\mathsf{pVerify}} := \mathsf{pVrfy}(pk, x, Y, \widetilde{\sigma})$
- **if** $pk = \mathsf{pk} \wedge (\widetilde{\sigma}, Y) \in \mathsf{H}_{\mathsf{pSign}}[x]$: $b_{\mathsf{pVerify}} := 1$ {*Pre-Sign Correctness*
- **else if** $pk = \mathsf{pk} \wedge \neg\mathsf{corrupted}$:
  - **if** $\exists x' \neq x : \widetilde{\sigma} \in \mathsf{H}_{\mathsf{pSign}}[x'] \cup \mathsf{H}_{\mathsf{pVerify}}[x']$: $b_{\mathsf{pVerify}} := 0$ {*Message Bounding*
- **if** $pk = \mathsf{pk} \wedge b_{\mathsf{pVerify}}$: **add** $(\widetilde{\sigma}, Y)$ **to** $\mathsf{H}_{\mathsf{pVerify}}[x]$
- Return $(\texttt{VerResult}, b_{\mathsf{pVerify}})$

On $(\texttt{Verify}, pk, x, \sigma)$ for party pid, do:
- $b_{\mathsf{verify}} := \mathsf{Vrfy}(pk, x, \sigma)$
- **if** $pk = \mathsf{pk} \wedge (x, \sigma) \in \mathsf{H}_{\mathsf{sign}}$: $b_{\mathsf{verify}} := 1$ {*Correctness*
- **else if** $pk = \mathsf{pk} \wedge \neg\mathsf{corrupted}$:
- **if** $pk = \mathsf{pk} \wedge b_{\mathsf{verify}}$: **add** σ **to** $\mathsf{H}_{\mathsf{verify}}[x]$
- Return $(\texttt{VerResult}, b_{\mathsf{verify}})$

As usual, the attacker may corrupt any party pid dynamically and gains full control over that party. In addition, if pid is the owner of the key pair modeled by this instance, i.e., the session ID of this instance is $sid = (pid, sid')$, then the attacker also learns sk; for notational convenience, we set $\mathsf{corrupted} = \mathsf{true}$ iff this is the case.

Fig. 14. Ideal adoptor signatures functionality $\mathcal{F}^2_{\text{asig}}$.

When the pre-signature oracle outputs a pre-signature $\widetilde{\sigma}$ on a message x, but the same pre-signature is already valid on x', or if the environment provides such a fresh, valid pre-signature to the pre-verification oracle, our reduction simply outputs the pre-signature and the matching messages. Since we just call efficient oracles polynomially often, our reduction is efficient. Since we assume that AS is correct, valid pre-signatures in our simulation are also valid w.r.t. the pre-verification algorithm.

Whenever the functionality $\mathcal{F}^2_{\text{asig}}$ differs from $\mathcal{F}^1_{\text{asig}}$, our reduction breaks the message-bound pre-sigature property of AS. This holds since, by correctness, pre-signatures output by the pre-sign algorithm are valid. In addition, our reduction found a single pre-signature that is valid on two distinct messages (and the respective statements). As we assume AS to have message-bound pre-signatures, the gap between $\mathcal{F}^2_{\text{asig}}$ and $\mathcal{F}^1_{\text{asig}}$ is negligible in the security parameter.

Game $\mathcal{G}_3$. In this game, we enforce pre-signature adaptability. This means, in this functionality, each signature that was obtained by adapting a pre-verifying pre-signature is guaranteed to verify. Besides this, the functionality $\mathcal{F}^3_{\text{asig}}$ has the same outputs as $\mathcal{F}^2_{\text{asig}}$. We depict $\mathcal{F}^3_{\text{asig}}$ in Figure 15.

Claim 3 *If AS, in addition to all the properties required by previous claims, satisfies pre-signature adaptability Definition 10, then the gap between $\mathcal{F}^3_{asig}$ and $\mathcal{F}^2_{asig}$ is negligible.*

Proof. The outputs of the functionalities $\mathcal{F}^3_{\text{asig}}$ and $\mathcal{F}^2_{\text{asig}}$ differ if the verification algorithm is called on a signature and a message x, such that the signature should be invalid by the verification algorithm, but the signature is part of the set $\mathsf{H}_{\mathsf{adapt}}[x]$. The signature can be part of $\mathsf{H}_{\mathsf{adapt}}[x]$, if there exists a pre-signature–statement pair, which is a part of $\mathsf{H}_{\mathsf{pVerify}}$, and the adapt oracle was called upon this pre-signature–statement pair and a valid witness for the mentioned statement. A pre-signature–statement pair is added to $\mathsf{H}_{\mathsf{pVerify}}$, if there exists a message x, such that either the pre-verification algorithm verifies on this triple, or the pre-signature was output by the pre-sign oracle, and hence pre-verifies by pre-sign correctness. Since pre-sign correctness is a perfect property, we know that in either case, the pre-verification algorithm on this pre-signature–statement pair, in addition to the message x, outputs 1.

This means, that $\mathcal{F}^3_{\text{asig}}$ differs from $\mathcal{F}^2_{\text{asig}}$, if a valid pre-signature w.r.t. some message x and some statement Y yields into an invalid adapted signature when adapted using a valid witness y for Y–effectively violating pre-signature adaptablity. As we assume AS to have (perfect) pre-signature adaptability, there is no gap between $\mathcal{F}^3_{\text{asig}}$ and $\mathcal{F}^2_{\text{asig}}$.

At this point in the proof, not only the signatures stored in $\mathsf{H}_{\mathsf{sign}}[x]$ and $\mathsf{H}_{\mathsf{verify}}[x]$ verify w.r.t. some message x, but also all signatures stored in the set $\mathsf{H}_{\mathsf{adapt}}[x]$. This holds by pre-signature adaptability which is a perfect property.

Game $\mathcal{G}_4$. In this game, we enforce unique extractability as long as the signer remains uncorrupted. This is done by ensuring that the pre-sign oracle does not output a pre-signature that extracts with two valid signatures, and the sign oracle does not output a signature that extracts with a pre-signature that already has an extracting signature. We do the same in the Adapt Oracle. Furthermore, we ensure in the verification and pre-verification oracles that no (pre-)signature is valid if it allows extracting more than one (pre-)signature. We enforce this functionality by introducing the helper functions $\mathsf{BreakSigUnExt}$ and $\mathsf{BreakpSigUnExt}$ that, on input a message,(pre-)signature pair, output conflicting signatures which are considered valid. If this happens, the candidate (pre-)signatures are either considered invalid (in the verification case), the pre-sign and sign oracles output $\perp$ instead of such a conflicting (pre-)signature, and the adapt oracle outputs the conflicting signature instead. Besides this, the functionality $\mathcal{F}^4_{\text{asig}}$ has the same outputs as $\mathcal{F}^3_{\text{asig}}$. We depict $\mathcal{F}^4_{\text{asig}}$ in Figure 16.

Claim 4 *If AS, in addition to all the properties required by previous claims, achieves unique extractability Definition 12, then the gap between $\mathcal{F}^4_{asig}$ and $\mathcal{F}^3_{asig}$ is negligible.*

Proof. The functionalities $\mathcal{F}^3_{\text{asig}}$ and $\mathcal{F}^4_{\text{asig}}$ differ, if the algorithms $\mathsf{BreakSigUnExt}$ and $\mathsf{BreakpSigUnExt}$ have non-bot outputs. To prove Claim 4, we show that the probability that the algorithms $\mathsf{BreakSigUnExt}$ and $\mathsf{BreakpSigUnExt}$ do not output $\perp$ when queried on valid (pre-)signatures is negligible. Since this is the only

The functionality is parameterized by a polytime decidable relation $\mathcal{R}$, ppt algorithms $\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{pSign}$ and dpt algorithms $\mathsf{Vrfy}, \mathsf{pVrfy}, \mathsf{Adapt}, \mathsf{Extract}$. Upon first activation, initialize this instance by running and storing $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.

On $\mathtt{PubKey}$? from anyone, return $(\mathtt{PubKey}, \mathsf{pk})$.

On $(\mathtt{Sign}, x)$ for party pid in session $sid = (pid, sid')$, do:

$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, x)$
if $\sigma \neq \bot$: **add** σ **to** $\mathsf{H}_{\mathsf{sign}}[x]$
Return $(\mathtt{Signature}, \sigma)$.

On $(\mathtt{pSign}, x, Y)$ for party pid in session $sid = (pid, sid')$, do:

$\widetilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, x, Y)$
if $\neg\mathsf{corrupted} \wedge \exists x' \neq x : \widetilde{\sigma} \in \mathsf{H}_{\mathsf{pSign}}[x'] \cup \mathsf{H}_{\mathsf{pVerify}}[x']$: $\widetilde{\sigma} := \bot$ {*Message Bounding*
if $\widetilde{\sigma} \neq \bot$: **add** $(\widetilde{\sigma}, Y)$ **to** $\mathsf{H}_{\mathsf{pSign}}[x]$
Return $(\mathtt{PreSignature}, \widetilde{\sigma})$

On $(\mathtt{Adapt}, pk, \widetilde{\sigma}, y, Y)$ for party pid, do:

$\sigma \leftarrow \mathsf{Adapt}(pk, \widetilde{\sigma}, y, Y)$
if $pk = \mathsf{pk} \wedge (Y, y) \in \mathcal{R} \wedge \exists x : (\widetilde{\sigma}, Y) \in \mathsf{H}_{\mathsf{pVerify}}[x]$:
  add $(\widetilde{\sigma}, Y, y, \sigma)$ **to** $\mathsf{H}_{\mathsf{adapt}}[x]$
Return $(\mathtt{Adapt}, \sigma)$

On $(\mathtt{Extract}, pk, \widetilde{\sigma}, \sigma, Y)$ for party pid, do:

$y := \mathsf{Extract}(pk, \widetilde{\sigma}, \sigma, Y)$
if $(Y, y) \notin \mathcal{R} \wedge \exists (x, y') : (\widetilde{\sigma}, Y, y', \sigma) \in \mathsf{H}_{\mathsf{adapt}}[x]$: $y := y'$ {*Extract Correctness*
Return $(\mathtt{Extract}, y)$

On $(\mathtt{pVerify}, pk, x, Y, \widetilde{\sigma})$ for party pid, do:

$b_{\mathsf{pVerify}} := \mathsf{pVrfy}(pk, x, Y, \widetilde{\sigma})$
if $pk = \mathsf{pk} \wedge (\widetilde{\sigma}, Y) \in \mathsf{H}_{\mathsf{pSign}}[x]$: $b_{\mathsf{pVerify}} := 1$ {*Pre-Sign Correctness*
else if $pk = \mathsf{pk} \wedge \neg\mathsf{corrupted}$:
  if $\exists x' \neq x : \widetilde{\sigma} \in \mathsf{H}_{\mathsf{pSign}}[x'] \cup \mathsf{H}_{\mathsf{pVerify}}[x']$: $b_{\mathsf{pVerify}} := 0$ {*Message Bounding*
if $pk = \mathsf{pk} \wedge b_{\mathsf{pVerify}}$: **add** $(\widetilde{\sigma}, Y)$ **to** $\mathsf{H}_{\mathsf{pVerify}}[x]$
Return $(\mathtt{VerResult}, b_{\mathsf{pVerify}})$

On $(\mathtt{Verify}, pk, x, \sigma)$ for party pid, do:

$b_{\mathsf{verify}} := \mathsf{Vrfy}(pk, x, \sigma)$
if $pk = \mathsf{pk} \wedge (x, \sigma) \in \mathsf{H}_{\mathsf{sign}}$: $b_{\mathsf{verify}} := 1$ {*Correctness*
else if $pk = \mathsf{pk} \wedge \exists (\widetilde{\sigma}, Y, y, \sigma) \in \mathsf{H}_{\mathsf{adapt}}[x]$: $b_{\mathsf{verify}} := 1$ {*Pre-Sig Adaptability*
else if $pk = \mathsf{pk} \wedge \neg\mathsf{corrupted}$:
if $pk = \mathsf{pk} \wedge b_{\mathsf{verify}}$: **add** σ **to** $\mathsf{H}_{\mathsf{verify}}[x]$
Return $(\mathtt{VerResult}, b_{\mathsf{verify}})$

As usual, the attacker may corrupt any party pid dynamically and gains full control over that party. In addition, if pid is the owner of the key pair modeled by this instance, i.e., the session ID of this instance is $sid = (pid, sid')$, then the attacker also learns sk; for notational convenience, we set $\mathsf{corrupted} = \mathsf{true}$ iff this is the case.

Fig. 15. Ideal adoptor signatures functionality $\mathcal{F}^3_{\mathrm{asig}}$.

The functionality is parameterized by a polytime decidable relation $\mathcal{R}$, ppt algorithms $\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{pSign}$ and dpt algorithms $\mathsf{Vrfy}, \mathsf{pVrfy}, \mathsf{Adapt}, \mathsf{Extract}$. Upon first activation, initialize this instance by running and storing $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.

On $\texttt{PubKey}$? from anyone, return $(\texttt{PubKey}, \mathsf{pk})$.

On $(\texttt{Sign}, x)$ for party pid in session $sid = (pid, sid')$, do:
- $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, x)$
- **if** $\mathsf{BreakSigUnExt}(x, \sigma) \neq \bot$: $\sigma := \bot$ {*Unique Extractability*
- **if** $\sigma \neq \bot$: **add** σ **to** $\mathsf{H}_{\mathsf{sign}}[x]$
- Return $(\texttt{Signature}, \sigma)$.

On $(\texttt{pSign}, x, Y)$ for party pid in session $sid = (pid, sid')$, do:
- $\widetilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, x, Y)$
- **if** $\neg\mathsf{corrupted} \wedge \exists x' \neq x : \widetilde{\sigma} \in \mathsf{H}_{\mathsf{pSign}}[x'] \cup \mathsf{H}_{\mathsf{pVerify}}[x']$: $\widetilde{\sigma} := \bot$ {*Message Bounding*
- **if** $\mathsf{BreakpSigUnExt}(x, \widetilde{\sigma}, Y) \neq \bot$: $\widetilde{\sigma} := \bot$ {*Unique Extractability*
- **if** $\widetilde{\sigma} \neq \bot$: **add** $(\widetilde{\sigma}, Y)$ **to** $\mathsf{H}_{\mathsf{pSign}}[x]$
- Return $(\texttt{PreSignature}, \widetilde{\sigma})$

On $(\texttt{Adapt}, pk, \widetilde{\sigma}, y, Y)$ for party pid, do:
- $\sigma \leftarrow \mathsf{Adapt}(pk, \widetilde{\sigma}, y, Y)$
- **if** $pk = \mathsf{pk} \wedge (Y, y) \in \mathcal{R} \wedge \exists x : (\widetilde{\sigma}, Y) \in \mathsf{H}_{\mathsf{pVerify}}[x]$:
  - **if** $\sigma' := \mathsf{BreakSigUnExt}(x, \sigma) \wedge \sigma' \neq \bot$: $\sigma \leftarrow \sigma'$ {*Unique Extractabilty*
  - **add** $(\widetilde{\sigma}, Y, y, \sigma)$ **to** $\mathsf{H}_{\mathsf{adapt}}[x]$
- Return $(\texttt{Adapt}, \sigma)$

On $(\texttt{Extract}, pk, \widetilde{\sigma}, \sigma, Y)$ for party pid, do:
- $y := \mathsf{Extract}(pk, \widetilde{\sigma}, \sigma, Y)$
- **if** $(Y, y) \notin \mathcal{R} \wedge \exists (x, y') : (\widetilde{\sigma}, Y, y', \sigma) \in \mathsf{H}_{\mathsf{adapt}}[x]$: $y := y'$ {*Extract Correctness*
- Return $(\texttt{Extract}, y)$

On $(\texttt{pVerify}, pk, x, Y, \widetilde{\sigma})$ for party pid, do:
- $b_{\mathsf{pVerify}} := \mathsf{pVrfy}(pk, x, Y, \widetilde{\sigma})$
- **if** $pk = \mathsf{pk} \wedge (\widetilde{\sigma}, Y) \in \mathsf{H}_{\mathsf{pSign}}[x]$: $b_{\mathsf{pVerify}} := 1$ {*Pre-Sign Correctness*
- **else if** $pk = \mathsf{pk} \wedge \neg$corrupted:
  - **if** $\exists x' \neq x : \widetilde{\sigma} \in \mathsf{H}_{\mathsf{pSign}}[x'] \cup \mathsf{H}_{\mathsf{pVerify}}[x']$: $b_{\mathsf{pVerify}} := 0$ {*Message Bounding*
    - **if** $\mathsf{BreakpSigUnExt}(x, \widetilde{\sigma}) \neq \bot$: $b_{\mathsf{verify}} := 0$ {*Unique Extractability*
- **if** $pk = \mathsf{pk} \wedge b_{\mathsf{pVerify}}$: **add** $(\widetilde{\sigma}, Y)$ **to** $\mathsf{H}_{\mathsf{pVerify}}[x]$
- Return $(\texttt{VerResult}, b_{\mathsf{pVerify}})$

On $(\texttt{Verify}, pk, x, \sigma)$ for party pid, do:
- $b_{\mathsf{verify}} := \mathsf{Vrfy}(pk, x, \sigma)$
- **if** $pk = \mathsf{pk} \wedge (x, \sigma) \in \mathsf{H}_{\mathsf{sign}}$: $b_{\mathsf{verify}} := 1$ {*Correctness*
- **else if** $pk = \mathsf{pk} \wedge \exists (\widetilde{\sigma}, Y, y, \sigma) \in \mathsf{H}_{\mathsf{adapt}}[x]$: $b_{\mathsf{verify}} := 1$ {*Pre-Sig Adaptability*
- **else if** $pk = \mathsf{pk} \wedge \neg$corrupted:
  - **if** $\mathsf{BreakSigUnExt}(x, \sigma) \neq \bot$: $b_{\mathsf{verify}} := 0$ {*Unique Extractabilty*
- **if** $pk = \mathsf{pk} \wedge b_{\mathsf{verify}}$: **add** σ **to** $\mathsf{H}_{\mathsf{verify}}[x]$
- Return $(\texttt{VerResult}, b_{\mathsf{verify}})$

As usual, the attacker may corrupt any party pid dynamically and gains full control over that party. In addition, if pid is the owner of the key pair modeled by this instance, i.e., the session ID of this instance is $sid = (pid, sid')$, then the attacker also learns sk; for notational convenience, we set $\mathsf{corrupted} = \mathsf{true}$ iff this is the case.

Fig. 16. Ideal adoptor signatures functionality $\mathcal{F}_{\text{asig}}^4$.

case where $\mathcal{F}^4_{\text{asig}}$ and $\mathcal{F}^3_{\text{asig}}$ differ, this shows the claim. To do this, we provide a reduction to the unique extractability of AS. As input, our reduction receives a verification key and has access to a signing and a pre-signing oracle. Using these oracles and the provided keys, our reduction instantiates our functionality $\mathcal{F}^4_{\text{asig}}$. Since we only ensure unique extractability if the signer is not corrupted, our reduction does not need to provide the corresponding signing key to the adversary. Eventually, either the $\mathsf{BreakSigUnExt}$ algorithm or the $\mathsf{BreakpSigUnExt}$ algorithm does not output $\perp$, but a signature σ. We now describe how our reduction finds values $(x, Y, \widetilde{\sigma}, \sigma, \sigma')$, such that both signatures verify on x, the pre-signature verifies on Y, x, and the pre-signature allows to extract with both signatures to a valid witness for Y.

We start with the algorithm $\mathsf{BreakpSigUnExt}$, which has as input a valid pair $(x, \widetilde{\sigma}, Y)$. Having this input available, what is missing for our forgery against unique extractability are two valid signatures on x that both extract a valid witness with $(\widetilde{\sigma}, Y)$. When the algorithm $\mathsf{BreakpSigUnExt}$ is called, a set $\mathcal{Q}$ is filled with all verifying signatures (i.e. signatures from $\mathsf{H}_{\mathsf{sign}}[x]$ and $\mathsf{H}_{\mathsf{verify}}[x]$ that extract with $(\widetilde{\sigma}, Y)$. We capture all extracting signatures by running the $\mathsf{Extract}$ algorithm on the respective inputs. If $\mathsf{BreakpSigUnExt}$ outputs a signature which is non-$\perp$, this means that our list of extracting signatures contains at least two valid signatures $(\sigma_1 \neq \sigma_2)$ on x. Our reduction can return the pair $(x, \widetilde{\sigma}, Y, \sigma_1, \sigma_2)$ and successfully break unique extractability.

Now we consider the algorithm $\mathsf{BreakSigUnExt}$, which has as input a valid pair (x, σ). This algorithm outputs non-$\perp$ if there exists a valid pre-signature–statement pair $(\widetilde{\sigma}, Y)$ which extracts with σ, but additionally extracts with a signature $\sigma' \neq \sigma$ that is already valid on x. To find valid pre-signature candidates, the algorithm first fills up a set $\mathcal{Q}$ containing all valid pre-signature–statement pairs $(\widetilde{\sigma}, Y)$ contained in $\mathsf{H}_{\mathsf{pSign}}[x]$ and $\mathsf{H}_{\mathsf{pVerify}}[x]$ that extract with σ. Those are valid pre-signatures on x w.r.t. some statement Y. Then, if there exists another signature $\sigma' \neq \sigma$ that was crafted via honest adapting (i.e. $\sigma' \in \mathsf{H}_{\mathsf{adapt}}[x]$), the algorithm returns such σ'. We know by extract correctness (Definition 9), that the extract algorithm yields a valid witness for a statement Y, when called upon a valid pre-signature on Y and the corresponding adapted pre-signature σ (which was obtained by adapting the pre-signature using a valid witness for Y). Therefore, each signature $\sigma' \in \mathsf{H}_{\mathsf{adapt}}[x]$ also extracts a valid witness with the pre-signature in question. Alternatively, if there exists a valid signature $(\sigma' \neq \sigma)$ that allows extracting with any pre-signature in $\mathcal{Q}$ using the $\mathsf{Extract}$ algorithm, the helper algorithm also outputs σ'. To craft a valid forgery, our reduction can return the pair $(x, \widetilde{\sigma}, Y, \sigma, \sigma')$, where σ' is the found signature, and $\widetilde{\sigma}$ is the matching pre-signature from $\mathcal{Q}$.

We have shown how to turn each non-$\perp$ output of the helper functions $\mathsf{BreakSigUnExt}$ and $\mathsf{BreakpSigUnExt}$ into a valid forgery of unique extractability. In addition, our reduction is efficient, since it internally runs a ppt adversary, and simulates oracles using provided oracles from the unique extractability security game. Hence, none of these algorithms can output a non-$\perp$ message during a run of the functionality if the AS satisfies unique extractability with non-negligible probability. Since the functionalities only have a gap if one of the helper algorithms outputs a non-bot message, the gap between $\mathcal{F}^4_{\text{asig}}$ and $\mathcal{F}^3_{\text{asig}}$ is negligible.

Game $\mathcal{G}_5$. In this game, we enforce unforgeability for pre-signatures for as long as the signer is unocrrupted. This means, in this functionality, a pre-signature can just be valid if it was either pre-signed before or if the message on which this pre-signature is pre-verified was signed before. Otherwise, the pre-verification algorithm outputs 0 in $\mathcal{F}^5_{\text{asig}}$. Besides this, the functionality $\mathcal{F}^5_{\text{asig}}$ has the same outputs as $\mathcal{F}^4_{\text{asig}}$. We depict $\mathcal{F}^5_{\text{asig}}$ in Figure 17.

Claim 5 *If AS, in addition to all the properties required by previous claims, has unforgeable pre-signatures (Definition 2), then the gap between $\mathcal{F}^5_{asig}$ and $\mathcal{F}^4_{asig}$ is negligible.*

Proof. The outputs of the functionalities $\mathcal{F}^5_{\text{asig}}$ and $\mathcal{F}^4_{\text{asig}}$ differ only if the signer is uncorrupted, and the pre-verification algorithm is called on a pre-signature and a message x, such that the pre-signature should be valid by the pre-verification algorithm, but the message x was never signed before, and the message-statement pair in question was not pre-signed before. In such cases, $\mathcal{F}^5_{\text{asig}}$ does not qualify the pre-sigature to be valid, while $\mathcal{F}^4_{\text{asig}}$ would. We now show that the probability of such cases occurring is negligible in the security parameter. To this end, we provide a reduction that transforms any environment causing this deviation into an adversary that breaks the pre-signature unforgeability of AS. As input, our reduction

The functionality is parameterized by a polytime decidable relation $\mathcal{R}$, ppt algorithms $\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{pSign}$ and dpt algorithms $\mathsf{Vrfy}, \mathsf{pVrfy}, \mathsf{Adapt}, \mathsf{Extract}$. Upon first activation, initialize this instance by running and storing $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.

On $\texttt{PubKey}$? from anyone, return $(\texttt{PubKey}, \mathsf{pk})$.

On $(\texttt{Sign}, x)$ for party pid in session $sid = (pid, sid')$, do:
- $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, x)$
- **if** $\mathsf{BreakSigUnExt}(x, \sigma) \neq \bot$: $\sigma := \bot$ {*Unique Extractability*
- **if** $\sigma \neq \bot$: **add** σ **to** $\mathsf{H}_{\mathsf{sign}}[x]$
- Return $(\texttt{Signature}, \sigma)$.

On $(\texttt{pSign}, x, Y)$ for party pid in session $sid = (pid, sid')$, do:
- $\widetilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, x, Y)$
- **if** $\neg\mathsf{corrupted} \wedge \exists x' \neq x : \widetilde{\sigma} \in \mathsf{H}_{\mathsf{pSign}}[x'] \cup \mathsf{H}_{\mathsf{pVerify}}[x']$: $\widetilde{\sigma} := \bot$ {*Message Bounding*
- **if** $\mathsf{BreakpSigUnExt}(x, \widetilde{\sigma}, Y) \neq \bot$: $\widetilde{\sigma} := \bot$ {*Unique Extractability*
- **if** $\widetilde{\sigma} \neq \bot$: **add** $(\widetilde{\sigma}, Y)$ **to** $\mathsf{H}_{\mathsf{pSign}}[x]$
- Return $(\texttt{PreSignature}, \widetilde{\sigma})$

On $(\texttt{Adapt}, pk, \widetilde{\sigma}, y, Y)$ for party pid, do:
- $\sigma \leftarrow \mathsf{Adapt}(pk, \widetilde{\sigma}, y, Y)$
- **if** $pk = \mathsf{pk} \wedge (Y, y) \in \mathcal{R} \wedge \exists x : (\widetilde{\sigma}, Y) \in \mathsf{H}_{\mathsf{pVerify}}[x]$:
  - **if** $\sigma' := \mathsf{BreakSigUnExt}(x, \sigma) \wedge \sigma' \neq \bot$: $\sigma \leftarrow \sigma'$ {*Unique Extractabilty*
  - **add** $(\widetilde{\sigma}, Y, y, \sigma)$ **to** $\mathsf{H}_{\mathsf{adapt}}[x]$
- Return $(\texttt{Adapt}, \sigma)$

On $(\texttt{Extract}, pk, \widetilde{\sigma}, \sigma, Y)$ for party pid, do:
- $y := \mathsf{Extract}(pk, \widetilde{\sigma}, \sigma, Y)$
- **if** $(Y, y) \notin \mathcal{R} \wedge \exists (x, y') : (\widetilde{\sigma}, Y, y', \sigma) \in \mathsf{H}_{\mathsf{adapt}}[x]$: $y := y'$ {*Extract Correctness*
- Return $(\texttt{Extract}, y)$

On $(\texttt{pVerify}, pk, x, Y, \widetilde{\sigma})$ for party pid, do:
- $b_{\mathsf{pVerify}} := \mathsf{pVrfy}(pk, x, Y, \widetilde{\sigma})$
- **if** $pk = \mathsf{pk} \wedge (\widetilde{\sigma}, Y) \in \mathsf{H}_{\mathsf{pSign}}[x]$: $b_{\mathsf{pVerify}} := 1$ {*Pre-Sign Correctness*
- **else if** $pk = \mathsf{pk} \wedge \neg\mathsf{corrupted}$:
  - **if** $\exists x' \neq x : \widetilde{\sigma} \in \mathsf{H}_{\mathsf{pSign}}[x'] \cup \mathsf{H}_{\mathsf{pVerify}}[x']$: $b_{\mathsf{pVerify}} := 0$ {*Message Bounding*
    - **if** $\mathsf{BreakpSigUnExt}(x, \widetilde{\sigma}) \neq \bot$: $b_{\mathsf{verify}} := 0$ {*Unique Extractability*
    - **if** $\mathsf{H}_{\mathsf{sign}}[x] = \emptyset$: $b_{\mathsf{pVerify}} := 0$ {*Pre-Sig Unforgeability*
- **if** $pk = \mathsf{pk} \wedge b_{\mathsf{pVerify}}$: **add** $(\widetilde{\sigma}, Y)$ **to** $\mathsf{H}_{\mathsf{pVerify}}[x]$
- Return $(\texttt{VerResult}, b_{\mathsf{pVerify}})$

On $(\texttt{Verify}, pk, x, \sigma)$ for party pid, do:
- $b_{\mathsf{verify}} := \mathsf{Vrfy}(pk, x, \sigma)$
- **if** $pk = \mathsf{pk} \wedge (x, \sigma) \in \mathsf{H}_{\mathsf{sign}}$: $b_{\mathsf{verify}} := 1$ {*Correctness*
- **else if** $pk = \mathsf{pk} \wedge \exists (\widetilde{\sigma}, Y, y, \sigma) \in \mathsf{H}_{\mathsf{adapt}}[x]$: $b_{\mathsf{verify}} := 1$ {*Pre-Sig Adaptability*
- **else if** $pk = \mathsf{pk} \wedge \neg\mathsf{corrupted}$:
  - **if** $\mathsf{BreakSigUnExt}(x, \sigma) \neq \bot$: $b_{\mathsf{verify}} := 0$ {*Unique Extractabilty*
- **if** $pk = \mathsf{pk} \wedge b_{\mathsf{verify}}$: **add** σ **to** $\mathsf{H}_{\mathsf{verify}}[x]$
- Return $(\texttt{VerResult}, b_{\mathsf{verify}})$

As usual, the attacker may corrupt any party pid dynamically and gains full control over that party. In addition, if pid is the owner of the key pair modeled by this instance, i.e., the session ID of this instance is $sid = (pid, sid')$, then the attacker also learns sk; for notational convenience, we set $\mathsf{corrupted} = \mathsf{true}$ iff this is the case.

Fig. 17. Ideal adoptor signatures functionality $\mathcal{F}_{\text{asig}}^5$.

receives a public key and has access to a signing and a pre-signing oracle that outputs valid (pre-)signatures w.r.t. the provided public key. To simulate the functionality in the environment, our reduction uses this key and the provided oracles to answer signing and pre-signing queries. Adapting, extracting, and verifying are public algorithms, so our reduction can simulate them while applying the rules established in $\mathcal{F}_{\text{asig}}^4$. Since we only enforce unforgeable pre-signatures for uncorrupted signers, the reduction does not need to provide the signing key to the adversary. When the pre-verification oracle is called on a pre-signature–statement pair such that the pre-verification algorithm outputs 1, but the corresponding message was never signed nor pre-signed w.r.t. to this statement, our reduction outputs this pre-signature as forgery. This forgery is valid, since it was not output by the pre-sign oracle, and the message x was never signed before. Since we just call efficient oracles polynomially often, our reduction is efficient. Whenever the functionality $\mathcal{F}_{\text{asig}}^5$ differs from $\mathcal{F}_{\text{asig}}^4$, our reduction breaks the pre-signature unforgeability of AS. As we assume AS to have unforgeable pre-signatures, the gap between $\mathcal{F}_{\text{asig}}^5$ and $\mathcal{F}_{\text{asig}}^4$ is negligible in the security parameter.

Game $\mathcal{G}_6$. In this game, we enforce extractability for an uncorrupted signer. This means, in this functionality, each signature that is valid either was previously signed or allows extracting a valid witness together with a pre-signature previously output by the pre-signing oracle. Besides this, the functionality $\mathcal{F}_{\text{asig}}^6$ has the same outputs as $\mathcal{F}_{\text{asig}}^5$. Furthermore, this functionality is also our final functionality. We depict $\mathcal{F}_{\text{asig}}^6$ in Figure 18.

Claim 6 *If AS, in addition to all the properties required by previous claims, satisfies extractability (Definition 11), then the gap between $\mathcal{F}_{asig}^6$ and $\mathcal{F}_{asig}^5$ is negligible.*

Proof. The outputs of the functionalities $\mathcal{F}_{\text{asig}}^6$ and $\mathcal{F}_{\text{asig}}^5$ differ only if the signer is uncorrupted, and the verification algorithm is called on a signature and a message x, such that the signature should be valid by the verification algorithm, but the message was never previously signed, i.e. $\mathsf{H}_{\mathsf{sign}}[x] = \emptyset$, and there exists no pre-signature–statement pair $(\widetilde{\sigma}, Y) \in \mathsf{H}_{\mathsf{pSign}}[x]$ such that the extract algorithm does not output a valid witness for Y when called using the candidate signature and $\widetilde{\sigma}$. In such cases, $\mathcal{F}_{\text{asig}}^6$ does not qualify the signature to be valid, while $\mathcal{F}_{\text{asig}}^5$ would do so. We now show that the probability of such cases occurring is negligible in the security parameter. To this end, we provide a reduction that transforms any environment causing this deviation into an adversary that breaks the extractability of AS. As input, our reduction receives a public key and has access to a signing and a pre-signing oracle that outputs valid (pre-)signatures w.r.t. the provided public key. To simulate the functionality in the environment, our reduction uses this key and the provided oracles to answer signing and pre-signing queries. Adapting, extracting, and verifying are public algorithms, so our reduction can simulate them while applying the rules established in $\mathcal{F}_{\text{asig}}^5$.Since we enforce extractability only for honest signers, the reduction does not need to provide a matching signing key to the adversary. When the environment provokes a gap between $\mathcal{F}_{\text{asig}}^5$ and $\mathcal{F}_{\text{asig}}^6$, our reduction has available a signature–message pair such that the verification algorithm verifies, and there exists no pre-signature that qualifies for extracting a valid witness using the $\mathsf{Extract}$ algorithm, and the message was never signed before.

Since we just call efficient oracles and compute efficient algorithms polynomially often, our reduction is efficient. Whenever the functionality $\mathcal{F}_{\text{asig}}^6$ differs from $\mathcal{F}_{\text{asig}}^5$, our reduction breaks the extractability of AS. As we assume AS to have extractability, the gap between $\mathcal{F}_{\text{asig}}^6$ and $\mathcal{F}_{\text{asig}}^5$ is negligible in the security parameter.

The functionality is parameterized by a polytime decidable relation $\mathcal{R}$, ppt algorithms $\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{pSign}$ and dpt algorithms $\mathsf{Vrfy}, \mathsf{pVrfy}, \mathsf{Adapt}, \mathsf{Extract}$. Upon first activation, initialize this instance by running and storing $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.

On $\texttt{PubKey}$? from anyone, return $(\texttt{PubKey}, \mathsf{pk})$.

On $(\texttt{Sign}, x)$ for party pid in session $sid = (pid, sid')$, do:

$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, x)$
if $\mathsf{BreakSigUnExt}(x, \sigma) \neq \bot$: $\sigma := \bot$ {*Unique Extractability*
if $\sigma \neq \bot$: **add** σ **to** $\mathsf{H}_{\mathsf{sign}}[x]$
Return $(\texttt{Signature}, \sigma)$.

On $(\texttt{pSign}, x, Y)$ for party pid in session $sid = (pid, sid')$, do:

$\widetilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, x, Y)$
if $\neg\mathsf{corrupted} \wedge \exists x' \neq x : \widetilde{\sigma} \in \mathsf{H}_{\mathsf{pSign}}[x'] \cup \mathsf{H}_{\mathsf{pVerify}}[x']$: $\widetilde{\sigma} := \bot$ {*Message Bounding*
if $\mathsf{BreakpSigUnExt}(x, \widetilde{\sigma}, Y) \neq \bot$: $\widetilde{\sigma} := \bot$ {*Unique Extractability*
if $\widetilde{\sigma} \neq \bot$: **add** $(\widetilde{\sigma}, Y)$ **to** $\mathsf{H}_{\mathsf{pSign}}[x]$
Return $(\texttt{PreSignature}, \widetilde{\sigma})$

On $(\texttt{Adapt}, pk, \widetilde{\sigma}, y, Y)$ for party pid, do:

$\sigma \leftarrow \mathsf{Adapt}(pk, \widetilde{\sigma}, y, Y)$
if $pk = \mathsf{pk} \wedge (Y, y) \in \mathcal{R} \wedge \exists x : (\widetilde{\sigma}, Y) \in \mathsf{H}_{\mathsf{pVerify}}[x]$:
  if $\sigma' := \mathsf{BreakSigUnExt}(x, \sigma) \wedge \sigma' \neq \bot$: $\sigma \leftarrow \sigma'$ {*Unique Extractabilty*
  add $(\widetilde{\sigma}, Y, y, \sigma)$ **to** $\mathsf{H}_{\mathsf{adapt}}[x]$
Return $(\texttt{Adapt}, \sigma)$

On $(\texttt{Extract}, pk, \widetilde{\sigma}, \sigma, Y)$ for party pid, do:

$y := \mathsf{Extract}(pk, \widetilde{\sigma}, \sigma, Y)$
if $(Y, y) \notin \mathcal{R} \wedge \exists (x, y') : (\widetilde{\sigma}, Y, y', \sigma) \in \mathsf{H}_{\mathsf{adapt}}[x]$: $y := y'$ {*Extract Correctness*
Return $(\texttt{Extract}, y)$

On $(\texttt{pVerify}, pk, x, Y, \widetilde{\sigma})$ for party pid, do:

$b_{\mathsf{pVerify}} := \mathsf{pVrfy}(pk, x, Y, \widetilde{\sigma})$
if $pk = \mathsf{pk} \wedge (\widetilde{\sigma}, Y) \in \mathsf{H}_{\mathsf{pSign}}[x]$: $b_{\mathsf{pVerify}} := 1$ {*Pre-Sign Correctness*
else if $pk = \mathsf{pk} \wedge \neg\mathsf{corrupted}$:
  if $\exists x' \neq x : \widetilde{\sigma} \in \mathsf{H}_{\mathsf{pSign}}[x'] \cup \mathsf{H}_{\mathsf{pVerify}}[x']$: $b_{\mathsf{pVerify}} := 0$ {*Message Bounding*
    if $\mathsf{BreakpSigUnExt}(x, \widetilde{\sigma}) \neq \bot$: $b_{\mathsf{verify}} := 0$ {*Unique Extractability*
    if $\mathsf{H}_{\mathsf{sign}}[x] = \emptyset$: $b_{\mathsf{pVerify}} := 0$ {*Pre-Sig Unforgeability*
if $pk = \mathsf{pk} \wedge b_{\mathsf{pVerify}}$: **add** $(\widetilde{\sigma}, Y)$ **to** $\mathsf{H}_{\mathsf{pVerify}}[x]$
Return $(\texttt{VerResult}, b_{\mathsf{pVerify}})$

On $(\texttt{Verify}, pk, x, \sigma)$ for party pid, do:

$b_{\mathsf{verify}} := \mathsf{Vrfy}(pk, x, \sigma)$
$b_{\mathsf{extracts}} := \exists(\widetilde{\sigma}, Y) \in \mathsf{H}_{\mathsf{pSign}}[x] : (Y, \mathsf{Extract}(pk, \widetilde{\sigma}, \sigma, Y)) \in \mathcal{R}$
if $pk = \mathsf{pk} \wedge (x, \sigma) \in \mathsf{H}_{\mathsf{sign}}$: $b_{\mathsf{verify}} := 1$ {*Correctness*
else if $pk = \mathsf{pk} \wedge \exists(\widetilde{\sigma}, Y, y, \sigma) \in \mathsf{H}_{\mathsf{adapt}}[x]$: $b_{\mathsf{verify}} := 1$ {*Pre-Sig Adaptability*
else if $pk = \mathsf{pk} \wedge \neg\mathsf{corrupted}$:
  if $\mathsf{BreakSigUnExt}(x, \sigma) \neq \bot$: $b_{\mathsf{verify}} := 0$ {*Unique Extractabilty*
  if $\mathsf{H}_{\mathsf{sign}}[x] = \emptyset \wedge \neg b_{\mathsf{extracts}}$: $b_{\mathsf{verify}} := 0$ {*Extractabilty*
if $pk = \mathsf{pk} \wedge b_{\mathsf{verify}}$: **add** σ **to** $\mathsf{H}_{\mathsf{verify}}[x]$
Return $(\texttt{VerResult}, b_{\mathsf{verify}})$

As usual, the attacker may corrupt any party pid dynamically and gains full control over that party. In addition, if pid is the owner of the key pair modeled by this instance, i.e., the session ID of this instance is $sid = (pid, sid')$, then the attacker also learns sk; for notational convenience, we set $\mathsf{corrupted} = \mathsf{true}$ iff this is the case.

Fig. 18. Ideal adoptor signatures functionality $\mathcal{F}_{\text{asig}}^{6}$.