Stronger Security and Generic Constructions for Adaptor Signatures

Wei Dai*[1], Tatsuaki Okamoto*[2], and Go Yamamoto[3]

[1]Bain Capital Crypto
[2]NTT
[3]NTT Research

October 23, 2022

Abstract

Adaptor signatures have seen wide applications in layer-2 and peer-to-peer blockchain applications such as atomic swaps and payment channels. We first identify two shortcomings of previous literature on adaptor signatures. (1) Current aim of "script-less" adaptor signatures restricts instantiability, limiting designs based on BLS or current NIST PQC candidates. (2) We identify gaps in current formulations of security. In particular, we show that current notions do not rule out a class of insecure schemes. Moreover, a natural property concerning the on-chain *unlinkability* of adaptor signatures has not been formalized. We then address these shortcomings by providing new and stronger security notions, as well as new generic constructions from *any* signature scheme and hard relation. On definitions:

1. We develop security notions that strictly imply previous notions.
2. We formalize the notion of unlinkability for adaptor signatures.
3. We give modular proof frameworks that facilitate simpler proofs.

On constructions:

1. We give a generic construction of adaptor signature from *any* signature scheme and *any* hard relation, showing that theoretically, (linkable) adaptor signatures can be constructed from any one-way function.
2. We also give an *unlinkable* adaptor signature construction from any signature scheme and any strongly random-self reducible relation, which we show instantiations of using DL, RSA, and LWE.

*Work done while at NTT Research.

Contents

1 Introduction

1.1 Background and Problems

The invention of Bitcoin has ignited a vast amount of research in the area of distributed ledger technologies in the past decade. Scalability and interoperability are two crucial challenges for the mass adoption of blockchain technology. One fruitful direction towards solving these challenges is the study of layer-2 and peer-to-peer (P2P) protocols. For example, atomic swaps allow users to swap asset across different chains without a trusted third party, payment channels allow a group of users to conduct many off-chain payments while only posting a small number of transactions to the blockchain, while payment channel networks (PCNs) generalize payment channels to enable large-scale P2P payments. One important functionality underlying these applications is that of *atomicity*. Roughly, *atomicity* guarantees that a set of transactions should either all be posted (to their respective ledgers) or none are. We briefly review the two currently known techniques to achieve atomicity.

Atomicity in blockchains: scripting vs. adaptor signatures. One seminal technique put forth by the work of the Bitcoin Lightning Network is the so-called "Hash Time Lock Contracts" (HTLC) [20]. Roughly, it is a Bitcoin spending script that, in addition to verification of a signature, also checks the release of a hash pre-image. Atomicity for a set of transactions is achieved by requiring the release of the same pre-image. Hence, if one transaction is processed, then the same pre-image can be used in other transactions. Such techniques are used to build applications such as atomic swaps, payment channels, and payment channel networks.

Aiming to eliminate the use of special scripts for the above applications, Poelstra proposed a technique called "adaptor signatures" [19, 18], which replaces the role of "hash-locks". Roughly, an adaptor signature for some underlying signature scheme achieves the following: If Alice gives to Bob a "pre-signature" $\hat{\sigma}$ on some message m and instance Y, then any signature on message m, that is valid against public key of Alice, given by Bob will allow Alice to learn the witness y for instance Y. Hence, adaptor signature is a "script-less" hash-lock, in the sense that the release of the signature by Bob also releases the witness to Alice. Following the work of Poelstra, there have been numerous work proposing various forms of adaptor signatures [2, 14, 8, 23] and applying them in applications to payment channels [14, 1].

Do we need script-less adaptor signatures? It is understood [19, 18, 7] that eliminating the need for scripts allows "applications" to blockchains that do not support it. Indeed, adaptor signatures allow atomic swap between say Bitcoin and Monero [10]. However, we point out that it is not clear if constructions of more complex applications, e.g. payment channels, can be realized for script-less blockchains such as Monero. Indeed, currently known techniques for payment channels crucially rely on the use of scripts [14, 2, 1]. Hence, we reconsider the requirement of "script-less"—we consider adaptor signatures whose verification requires minimal additions to the underlying signature supported by the blockchain. This leads to the first shortcoming of the current approach.

Problem 1: Restricted design space and limited compatibility with post-quantum signature candidates. Previous works on adaptor signatures have one common theme—the adaptor signature scheme *must* work with a known underlying signature scheme, that is supported directly by the blockchain, e.g. ECDSA [13] and Schnorr [22]. This is due to the desire to eliminate the use of more complex spending scripts such as HTLC. However, one significant downside to such an approach is that adaptor signatures are not possible for all signature schemes. Indeed, it is known [7] that adaptor signature schemes are *impossible* for unique signatures such as BLS [5].

The problem of aiming for "script-less" adaptor signatures is even more pronounced when considering post-quantum (PQ) adaptor signatures. Neither of the two currently known post-quantum constructions, i.e. LAS [8] (based on LWE), and IAS [23] (based on isogenies), are compatible with the NIST PQC Round 3 PQ signature candidates [15]. The underlying signature scheme of LAS is a (strict) variant of Dilithium [6]. The underlying signature scheme of IAS is CSI-FiSh [4], which was proposed after the NIST PQC submission deadline. This means that even if the community arrives at a consensus on PQ standard signatures, further standardization and selection efforts might be required for extensions to adaptor signatures.

Prior works also leave open some natural theoretical questions, which we raise and answer in our work: What are the minimal assumptions required to construct adaptor signatures? In particular, are adaptor signatures in Minicrypt (i.e. can be constructed from one-way functions)?

Problem 2: Definitional gaps. The security definition for adaptor signatures, first proposed by [2], is later adopted by all subsequent follow-up works [8, 23, 1, 7]. However, there is a serious gap in this security definition. In particular, we demonstrate that such notions do not guarantee security against *multiple* queries to the pre-signature oracle with the same message and instance pair (m, Y). Moreover, we show that such gap is not just theoretical—we give a counterexample scheme that is secure against previous notions but renders many applications insecure if they are used in practice. The root cause of this gap is the weak forms of attacks considered. Specifically, the security game (for notions called aEUF-CMA and aWitExt) only allows a single challenge query to pre-sign, and hence does not rule out attacks that access pre-sign more than once.[1] Secondly, previous definitions fail to capture a natural property of adaptor signature schemes—on-chain unlinkability. For example, it is understood that atomic swaps based on Schnorr signatures give more privacy guarantees than HTLC-based solutions. Indeed, we show that such a property does not follow from prior formalization.

1.2 Our contributions

Our work addresses the problems outlined above. First, we give stronger definitions for adaptor signatures, as well as new definitions of unlinkability. We also give a modular proof framework to facilitate simpler proofs. Second, we give generic constructions of adaptor signatures from *any* secure signature scheme and *any* hard relation. The construction is unlinkable if the relation is assumed to be strongly random-self-reducible (SRSR). Our constructions are compatible with any of the current NIST PQC candidates. Answering the theoretical questions, we show (linkable) adaptor signatures can be constructed from one-way functions, and unlinkability can be achieved assuming additionally the existence of SRSR relations.

New security notions and modular proof framework. A significant portion of our technical contribution is regarding the security definition of adaptor signature. First, we close the definitional gap by giving two security notions, called (strong) full extractability, abbreviated as (S)FExt, that exposes rich sets of attack interfaces. We show that full extractability *strictly* implies previous notions. Next, we formalize the notion of unlinkability for adaptor signatures. Lastly, we present a modular proof framework, allowing the security of adaptor signatures to be proved against much simpler notions than full extractability, which we call simple and unique extractability.

Generic construction and unifying perspective. We revisit the main design constraint so far considered for adaptor signatures, namely that they should be "script-less scripts." We allow

[1] We remark that this is *not* a weakness of previous constructions, but rather a gap in the formal security guarantees and the security is expected for applications.

the use of minimal scripts and define "augmented" signature scheme SigR, which is defined against any signature Sig and any relation $\mathbb{R}$. Augmented scheme SigR additionally verifies the release of a witness alongside a valid signature. We then give an adaptor signature scheme GAS_1 for SigR and prove it secure (Theorem 4.1). We remark that adaptor signature GAS_1 is generic in the sense that it is constructed from *any* signature scheme and *any* hard relation. However, we remark that implementing augmented signature schemes for existing blockchains such as Bitcoin amounts to using scripts. Our work can be seen as a formalization of HTLCs as adaptor signatures.

Achieving on-chain unlinkability. We formally define the notion of unlinkability, which has not been formally studied previously. Unlinkability asks adapted signatures to be indistinguishable from standard ones, even if one knows the instance and witness pair used to derive the adapted signature. We show how to add unlinkability to GAS_1, assuming that relation R is strongly random-self-reducible (SRSR), obtaining a new construction which we name GAS_2, whose security proofs are given in Theorem 4.3. We show how to instantiate SRSR from standard number theoretical problems such as DL and RSA, as well as the learning with errors problem (LWE).

Flexible instantiations and minimal assumptions. We remark that our generic constructions can work with *any* signature scheme and *any* hard relation. In particular, our constructions can be instantiated with any of the NIST PQC Round 3 candidate schemes. For example, we could use Rainbow, which is NIST PQC round 3 [15] finalist based on multivariate polynomials, and with any post-quantum-secure SRSR relation, which could be based on lattice problems. More generally, our work shows that linkable adaptor signature is in Minicrypt (Theorem 4.2), meaning it can be constructed assuming the existence of one-way functions (GAS_1). Our construction of GAS_2 shows that the existence of strong random-self reducible relations implies the existence of unlinkable adaptor signatures (Theorem 4.4). Moreover, we remark that the overhead of GAS_1 and GAS_2 are also *minimal* in terms of computational overhead and signature size. GAS_1 has virtually no computational overhead and GAS_2 requires re-randomization of instances from the hard relation. An adapted signature for both schemes contains a standard signature of the underlying signature scheme as well as an instance and witness pair for the hard relation.

2 Preliminary

We use $[n]$ for a positive integer n to denote the set $\{1, \ldots, n\}$. Let S be a finite set. We use $x \twoheadleftarrow S$ to denote sampling from set S uniformly at random and assigning the result to variable x. "PT" denotes polynomial-time, also referred to as efficient. We use $\lambda \in \mathbb{N}$ to denote the security parameter. We recall that a function $f : \mathbb{N} \to \mathbb{R}$ is negligible if for every $c \in \mathbb{N}$, there exists $n_c \in \mathbb{N}$ such that $|f(n)| < n^{-c}$ for all $n > n_c$. Algorithms are probabilistic unless specified otherwise. Suppose $\mathcal{A}$ is an algorithm expecting oracles $\mathsf{O}_1, \ldots$, we use $x \twoheadleftarrow \mathcal{A}^{\mathsf{O}_1,\cdots}(\cdots)$ to denote an execution of algorithm $\mathcal{A}$ and assigning its output to variable x. We use $[\mathcal{A}^{\mathsf{O}_1,\cdots}(\cdots)]$ to denote the set of all possible outputs of algorithm $\mathcal{A}$. We use $S \xleftarrow{\cup} x$ to denote adding an element x to the set S. Integer variables are initialized to 0 and set variables are initialized to the empty set. We adopt the code-based game-playing framework of [3]. A game $\mathbf{G}$ is usually parameterized by some cryptographic scheme S and an adversary $\mathcal{A}$. A game consists of list of named oracles. The execution of a game is the execution of the Main procedure. Variables in game oracles are global by default. An "Assert" statement inside an oracle call will immediately terminate the execution of the oracle call and return False if the given expression evaluates to False. For an example of a game, see Figure 3.

Relations and random self-reducibility. We recall that a relation $\mathsf{R} \subseteq \{0,1\}^* \times \{0,1\}^*$ is an NP relation if there exists PT algorithm $\mathsf{R.Vf}$ such that $\mathsf{R.Vf}(Y, y) = \mathsf{True}$ if and only if $(Y, y) \in \mathsf{R}$. The language L_{R} for relation R is defined as the set $\{Y \in \{0,1\}^* \mid \exists y \in \{0,1\}^* : (Y, y) \in \mathsf{R}\}$. A generator for R is a PT algorithm $\mathsf{R.Gen}$ that on input 1^λ returns a pair $(Y, y) \in \mathsf{R}$. For any positive integer q, we define the q-one-wayness advantage of an adversary $\mathcal{A}$ against R to be $\mathbf{Adv}^{\text{q-ow}}_{\mathsf{R},\mathcal{A}}(\lambda) := \Pr[\mathbf{G}^{\text{q-ow}}_{R,q,\mathcal{A}}(\lambda)]$, where the one-wayness game is given below.

Game $\mathbf{G}^{\text{q-ow}}_{\mathsf{R},\mathcal{A}}(\lambda)$:

1 For $i \in [q]$ do $(Y_i, \cdot) \twoheadleftarrow \mathsf{R.Gen}(1^\lambda)$

2 $(I, y') \twoheadleftarrow \mathcal{A}(Y_1, \ldots, Y_q)$; Return $(Y_I, y') \in \mathsf{R}$

Above, adversary $\mathcal{A}$ returns an index $I \in \{1, \ldots, q\}$ and a guess y'. We say that R is a hard relation if for polynomial q and any PT $\mathcal{A}$, $\mathbf{Adv}^{\text{q-ow}}_{R,\mathcal{A}}(\lambda)$ is negligible. We define the following unique-witness advantage of an adversary $\mathcal{A}$ to be the $\mathbf{Adv}^{\text{uwit}}_{\mathsf{R},\mathcal{A}}(\lambda) := \Pr[(Y, y) \in \mathsf{R}, (Y, y') \in \mathsf{R}, y \neq y' \mid (Y, y, y') \twoheadleftarrow \mathcal{A}(1^\lambda)]$.

We say that a hard relation R is random self-reducible (RSR) if there exists sets $\mathsf{R}.R_\lambda$ and efficient deterministic algorithms $\mathsf{R}.A, \mathsf{R}.B, \mathsf{R}.C$ such that the following holds for any $(Y, y) \in [\mathsf{R.Gen}(1^\lambda)]$ and $r \twoheadleftarrow \mathsf{R}.R_\lambda$: (1) $Y' \leftarrow \mathsf{R}.A(Y, r)$ is distributed identically to $\mathsf{R.Gen}(1^\lambda)$. (2) For $y' \leftarrow \mathsf{R}.B(y, r)$ it holds that $(Y', y') \in R$ where $Y' = \mathsf{R}.A(Y, r)$. (3) For $y \leftarrow \mathsf{R}.C(y', r)$, where $y' = \mathsf{R}.B(y, r)$, it holds that $(Y, y) \in \mathsf{R}$.

Signature schemes. A signature scheme Sig consists of PT algorithms KeyGen, Sign, and Vrf. Via $(\text{pk}, \text{sk}) \twoheadleftarrow \mathsf{KeyGen}(1^\lambda)$, the key generation algorithm generates a public key pk and a secret key sk. Via $\sigma \twoheadleftarrow \mathsf{Sign}(\text{sk}, m)$, the signing algorithms generates a signature σ. Via $b \twoheadleftarrow \mathsf{Vrf}(\text{pk}, m, \sigma)$, the verification algorithm returns a boolean value $b \in \{\mathsf{True}, \mathsf{False}\}$, indicating the validity of the message signature pair. Consider the game $\mathbf{G}^{\text{uf-cma}}_{\mathsf{Sig}}$ given below.

Game $\mathbf{G}^{\text{uf-cma}}_{\mathsf{Sig},\mathcal{A}}(\lambda)$, $\mathbf{G}^{\text{suf-cma}}_{\mathsf{Sig},\mathcal{A}}(\lambda)$

1 $(\text{pk}, \text{sk}) \twoheadleftarrow \mathsf{KeyGen}(1^\lambda)$; $(m, \sigma) \twoheadleftarrow \mathcal{A}^{\text{Sign}}(\text{pk})$; Assert $\mathsf{Vrf}(\text{pk}, m, \sigma)$

2 $\mathbf{G}^{\text{uf-cma}}_{\mathsf{Sig},\mathcal{A}}$: Return $(m \notin S)$

3 $\mathbf{G}^{\text{suf-cma}}_{\mathsf{Sig},\mathcal{A}}$: Return $((m, \sigma) \notin U)$

$\mathsf{Sign}(m)$:

4 $\sigma \twoheadleftarrow \mathsf{Sign}(\text{sk}, m)$; $S \leftarrow S \cup \{m\}$; $U \leftarrow U \cup \{(m, \sigma)\}$; Return σ

We define the (S)UF-CMA advantage of an adversary $\mathcal{A}$ against Sig to be $\mathbf{Adv}^{\text{uf-cma}}_{\mathsf{Sig},\mathcal{A}}(\lambda)$ ($\mathbf{Adv}^{\text{suf-cma}}_{\mathsf{Sig},\mathcal{A}}(\lambda)$). We say that scheme Sig is (S)UF-CMA-secure if $\mathbf{Adv}^{\text{uf-cma}}_{\mathsf{Sig},\mathcal{A}}(\lambda)$ ($\mathbf{Adv}^{\text{suf-cma}}_{\mathsf{Sig},\mathcal{A}}(\lambda)$) is negligible for any PT adversary $\mathcal{A}$.

3 Definitions and Relations

The inception of Adaptor signatures was due to the idea of "script-less scripts" by Polstra, who proposed ways to construct applications of atomic swaps [19] and atomic multi-hop payments [18] without the use of special spending scripts. In a follow-up work, Malavolta el al. implicitly gave constructions of Schnorr and ECDSA adaptor signatures in their work on anonymous multi-hop locks for payment channel networks [14]. The notion of adaptor signature was formally defined and studied by [2] and independently by [9]. Follow-up works have constructed two-party adaptor signatures [7] and post-quantum adaptor signatures [8, 23], as well as building other applications

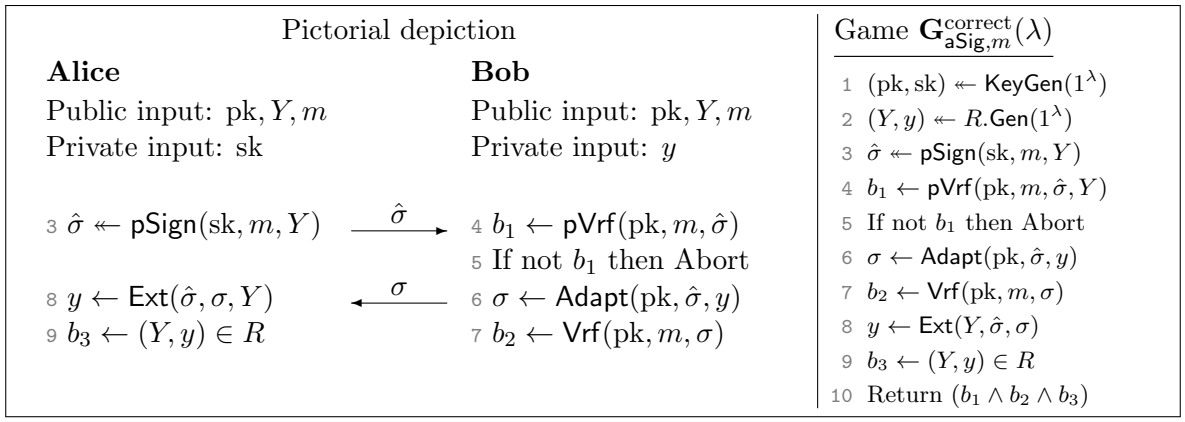


Figure 1: **Left**: Pictorial depiction of an honest execution of adaptor signing protocol between Alice and Bob. Alice holds the secret key sk corresponding to her public key pk. Bob holds the witness y corresponding to his public instance Y. We assume that Alice and Bob have agreed on the message m to be signed before the execution of the protocol. In this depiction, we have simplified the release of σ from Bob. In practical settings, Alice could learn σ from an indirect channel, e.g. a public ledger. We also remark that Bob does not need his private input y for lines 4 and 5, and his private input y is only needed for line 6. **Right**: game defining the correctness of an adaptor signature scheme.

---

on top of adaptor signatures [1]. All follow-up works follow the security definitions of [2]. Our definitions align with that of [2] in terms of syntax, correctness, and basic security (called pre-signature adaptability). Our framework deviates from and significantly improves upon the main security definitions of [2].

Syntax and correctness. Let $\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrf})$ be a signature scheme. Let R be a hard relation with generator Gen. An adaptor signature scheme aSig for signature scheme Sig and relation R specifies (probabilistic) algorithm pSign, and deterministic algorithms pVrf, Adapt, and Ext. We assume that algorithms of Sig, i.e. $\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrf}$, are additionally defined to be algorithms of aSig. Let $(\mathrm{pk}, \mathrm{sk}) \in [\mathsf{Sig.KeyGen}(1^\lambda)]$ and $(Y, y) \in [\mathsf{Gen}(1^\lambda)]$. The adaptor signature algorithms behave as follows.

- Via $\hat{\sigma} \leftarrow \mathsf{pSign}(\mathrm{sk}, m, Y)$ the pre-sign algorithm generates a pre-signature.
- Via $b \leftarrow \mathsf{pVrf}(\mathrm{pk}, m, \hat{\sigma}, Y)$, the pre-signature verification returns a boolean.
- Via $\sigma \leftarrow \mathsf{Adapt}(\mathrm{pk}, \hat{\sigma}, y)$, the adapt algorithm returns a signature σ.
- Via $y \leftarrow \mathsf{Ext}(\sigma, \hat{\sigma}, Y)$, the extract algorithm extracts a witness y.

To introduce correctness, let us first consider an example of honest execution of adaptor signature between two parties Alice and Bob. Consider the protocol given in Figure 1. In typical usage of adaptor signatures, algorithms pSign and Ext are executed by some party (Alice) holding sk, and pVrf and Adapt are executed by a party (Bob) holding secret witness y. We note that Alice and Bob can execute the protocol up to step 5 even if Bob does not know secret witness y; on the other hand, execution of steps 6-9 requires Bob's knowledge of the witness y. We say that aSig is *correctness* if for all message m, $\Pr[\mathbf{G}^{\mathrm{correct}}_{\mathsf{aSig},m}(\lambda)] = 1$. We remark that inputs to extraction are all public, meaning that any external observer can extract out a witness y. This is an intended property of adaptor signatures which allows Alice to delegate witness extraction to third parties.

Notion(s)	Secret key holder Alice	Witness holder Bob	Observer Carol
Correctness	Honest	Honest	-
Adaptability	Malicious	Honest	-
Extractability	Honest	Malicious	-
Unlink (new)	Honest	Honest	Malicious

Relations among extractability notions

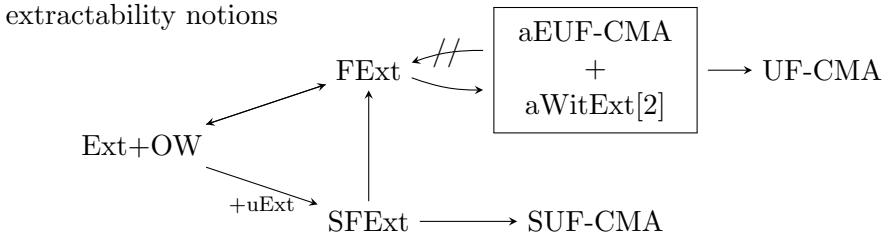


Figure 2: **Top**: Guide to correctness and security notions. **Bottom**: Relations among security notions for the secret-key holder (Alice). An arrow A $\rightarrow$ B means that if an adaptor signature scheme is A-secure, then it is B-secure. Additional assumptions are marked on top of arrows. All implications are tight, meaning reductions preserve running time and success advantage up to small additive constants.

Adaptor signatures with canonical signing. Any adaptor signature scheme aSig gives an alternative way to generating signatures via pSign and Adapt. Specifically, we consider the following signing algorithm.

Algorithm $\mathsf{Sign}'(m)$

1 $(Y, y) \leftarrow\!\!\!\!\$\ \mathsf{R.Gen}(1^\lambda)$; $\hat{\sigma} \leftarrow\!\!\!\!\$\ \mathsf{pSign}(\mathrm{sk}, m, Y)$; Return $\sigma \leftarrow \mathsf{Adapt}(\mathrm{pk}, \hat{\sigma}, y)$

We say that an adaptor signature scheme has *canonical* signing if the above signing algorithm gives signatures that are *identically distributed* to those given by Sign for any secret-key sk and message m. Any adaptor signature scheme can be turned into a canonical one by simply replacing the signing algorithm with the one defined above. All schemes considered in this work are canonical without modifications.

Security of Bob: Adaptability. The first security notion we introduce is called *adaptability*. Intuitively, it guarantees that if Bob is convinced of the validity of the pre-signature $\hat{\sigma}$ and knows a corresponding witness y, then Bob can generate a valid signature σ. More specifically, we ask that for any pk, m, and $\hat{\sigma}$, if $(Y, y) \in R$ and $\mathsf{pVrf}(\mathrm{pk}, m, \hat{\sigma}, Y)$ returns true, it must be that $\mathsf{Adapt}(\mathrm{pk}, \sigma, y)$ returns a valid signature σ on message m wrt to public key pk. Referring back to Figure 1, pre-signature adaptability guarantees the safety of Bob—he can always turn a valid pre-signature into a signature if he learns a corresponding witness. Formally, consider the following game.

Game $\mathbf{G}^{\mathrm{adapt}}_{\mathsf{aSig},\mathcal{A}}(\lambda)$

1 $(\mathrm{pk}, \mathrm{sk}) \leftarrow\!\!\!\!\$\ \mathsf{KeyGen}(1^\lambda)$; $(m, \hat{\sigma}, (Y, y)) \leftarrow\!\!\!\!\$\ \mathcal{A}(\mathrm{pk})$
2 Assert $((Y, y) \in R \ \wedge\ \mathsf{pVrf}(\mathrm{pk}, m, \hat{\sigma}, Y))$
3 Return $\mathsf{Vrf}(\mathrm{pk}, m, \mathsf{Adapt}(\mathrm{pk}, \hat{\sigma}, y))$

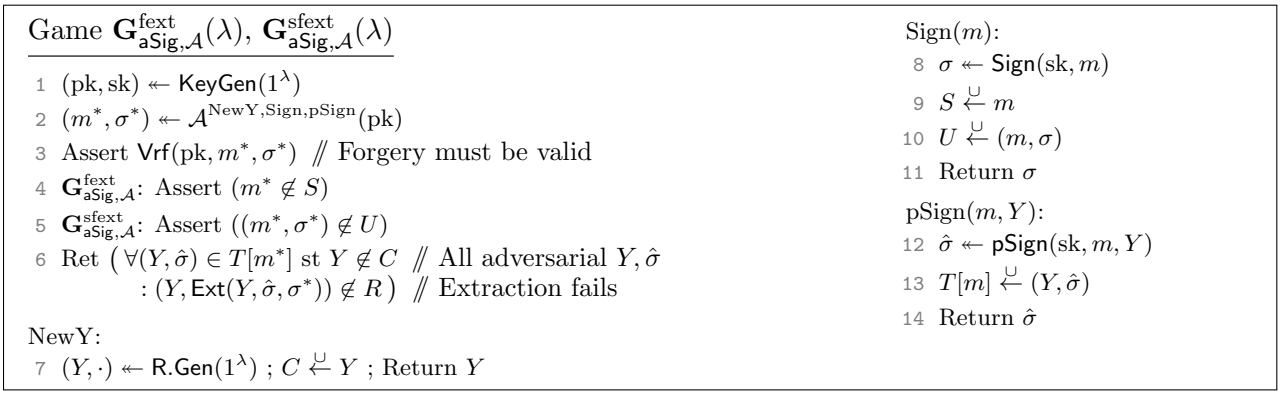


Figure 3: Games defining (strong) full extractability notions.

We say that aSig satisfies *pre-signature adaptability* if $1 - \Pr[\mathbf{G}^{\text{adapt}}_{\mathsf{aSig},\mathcal{A}}(\lambda)]$ is negligible for all PT adversary $\mathcal{A}$. We say that aSig has *perfect pre-signature adaptability* if for all adversary $\mathcal{A}$, $\Pr[\mathbf{G}^{\text{adapt}}_{\mathsf{aSig},\mathcal{A}}(\lambda)] = 1$. Our definition of pre-signature adaptability aligns with that of [2].

Security of Alice: Full Extractability (FExt). The most involved part of the security definition is for the secret-key holder of the signature scheme (Alice). This is where previous definitions fall short. We give a unified definition of security for the safety of the secret-key holder. Before we give the formal definition, we give a high-level description of the available attack surfaces. A secret-key holder, Alice, could potentially expose the following interfaces.

- $\mathsf{Sign}(\text{sk}, m)$ for adversarially chosen m. Exposing such signing oracle models the applications in which honestly generated signatures of Alice are released. Note that we cannot prevent previous honestly generated signatures from being valid. This is similar to the signing oracle that is in UF-CMA and SUF-CMA notions. In fact, we will consider two variants of security for Alice as well.

- $\mathsf{pSign}(\text{sk}, m, Y)$ for adversarially chosen m and instance Y. This models all interactions that Alice could have with external parties where Alice gives out pre-signatures. Each query generates a tuple $(m, Y, \hat{\sigma})$, and we store them in a table indexed by message m, i.e. each query adds the pair $(Y, \hat{\sigma})$ to the set $T[m]$, which is initialized to the empty set.

- Forgery guarantee: For Alice, after given out many signatures and pre-signatures, the following guarantee is desired: if some forgery (m^*, σ^*) is given by an adversary, then one of the following must hold: (1) (m^*, σ^*) must have come from a signing query (2) There must be a corresponding tuple $(Y, \hat{\sigma})$ in table $T[m]$ such that σ^* gives a valid extraction, i.e. $\mathsf{Ext}(Y, \hat{\sigma}, \sigma^*)$ gives some y such that $(Y, y) \in \mathsf{R}$.

Additionally, in the above scenario where Bob (the adversary) returns a signature that extracts, we would like to additionally restrict the *instances* with respect to which extraction could happen. Specifically, we would like to separate instances given to pSign into two categories: (1) those for which Bob knows a witness and (2) those for which Bob does not know a witness. This is to ensure that even if Bob learns a pre-signature on some instance Y for which it does not know the witness, it cannot adapt it into a valid signature. This is achieved in the formal security notion by introducing an oracle NewY that samples honest instances for Bob.

To summarize, extractability guarantees that if Alice gives out signatures and pre-signatures, then the only forgery that some adversary Bob can give is (1) those already given by Alice as

signatures (2) some forgery that leads to a valid extraction of a witness. Formally, consider the games $\mathbf{G}^{\text{fext}}_{\mathsf{aSig},\mathcal{A}}$ and $\mathbf{G}^{\text{sfext}}_{\mathsf{aSig},\mathcal{A}}$ given in Figure 3. In either game, the adversary is given some honestly sampled public key pk and has access to oracles NewY, Sign, pSign (line 2). Each query to Sign is recorded as the allowed forgery budget (set S for the normal case and U for the strong case). Each query m, Y to pSign derives some $\hat{\sigma}$ (line 15) and these values are recorded in table $T[m]$ (line 14). In the end, to win the game, the adversary needs to produce a valid forgery (line 2-3) which is fresh (line 4 or 5) and does not produce any valid extraction (line 6) against instances chosen by Bob (i.e. excluding those Y generated by NewY). Note that the full extractability game only requires the forgery to be on a fresh message (line 4), and the strong full extractability game requires the entire forgery to be fresh (line 5). This differentiation is consistent with the difference between UF-CMA and SUF-CMA security of signatures. We define the (S)FExt advantage of an adversary $\mathcal{A}$ to be $\mathbf{Adv}^{\text{fext}}_{\mathsf{aSig},\mathcal{A}}(\lambda)$ ($\mathbf{Adv}^{\text{sfext}}_{\mathsf{aSig},\mathcal{A}}(\lambda)$), and we say that scheme aSig is (S)FExt-secure if the corresponding advantage function if negligible for efficient adversaries.

Intuitively, if messages to be signed and pre-signed has high entropy and do not repeat, then it suffices to only assume FExt. On the other hand, if applications expect messages to repeat, then it is crucial to additionally aim for SFExt.

Strict implications to notions given by [2]. We show that FExt implies previous notions of unforgeability and extractability (Theorem 3.1), formally aEUF-CMA and aWitExt as defined in [2]. Their notions have since been adopted in following works [1, 7, 8, 23]. Roughly, their security notion guarantees "unforgeability" and "extractability" against a single challenge pre-sign query. Unlike our full extractability notion, their notions specify an *explicit* phase for the challenge message selection and forgery generation. This results in the adversary only learning *exactly one* pre-signature $\hat{\sigma}^* = \mathsf{pSign}(\text{sk}, m^*, Y^*)$ on challenge message m^* and instance Y^*. It is not hard to see that our notion of FExt implies aEUF-CMA and aWitExt. On the other hand, we give a counterexample scheme to show that such implication is *strict* (Theorem 3.2), meaning there are schemes which are aEUF-CMA- and aWitExt-secure that are not FExt-secure. We give a sketch of the counterexample below and give the full analysis in Section 3.1.

Counterexample. We will modify a secure (in the sense of aEUF-CMA and aWitExt) adaptor signature scheme aSig so that pSign leaks an additional signature on message m if and only if pSign is called with the same message m and instance Y more than once. In more detail, pSign will do the following: upon a query $\mathsf{pSign}(\text{sk}, m, Y)$, we first compute a pre-signature $\hat{\sigma}$ exactly as in aSig. Additional to $\hat{\sigma}$, pSign will return some string C_b for a random bit b where (1) C_0 is a random encryption pad and (2) $C_1 = C_0 \oplus \sigma$ is a one-time pad encryption of a signature $\sigma = \mathsf{Sign}(\text{sk}, m; r)$ under randomness r. To make these values consistent across different runs of pSign, we derive values of (C_0, r) via a PRF F (with secret key sk) applied to the input (m, Y), i.e. $(C_0, r) \leftarrow \mathsf{F}(\text{sk}, (m, Y))$. Hence, any single call to $\mathsf{pSign}(m, Y)$ does not reveal any information. Indeed, We will show that the resulting scheme is secure in the sense of aEUF-CMA and aWitExt. However, even two calls $\mathsf{pSign}(m, Y)$ reveals a fresh signature with half probability, hence breaking FExt security.

Our counterexample scheme demonstrates that security guaranteed by aEUF-CMA and aWitExt is weaker than expected for applications where many protocols might be executed concurrently since both notions only guarantee security against a single challenge instance. Moreover, aEUF-CMA is also a *selective* notion, in that the honest instance Y_1 is not known to the adversary until *after* the adversary selects a challenge message m^*. This further weakens the security guaranteed. In contrast, our security notion FE gives the challenge instance Y_1 to the adversary and allows any number of challenge queries to pSign.

Modular proofs from simpler notions. Full extractability and strong full extractability are fairly complex notions, where the adversary is given many attack interfaces. To facilitate simpler proofs and better intuitive understanding, we give a framework in Section 3.2 for proving FExt and SFExt security. In particular, we show that proofs can be modularized if a simplified notion called *simple extractability* is achieved (Theorem 3.3). Roughly, simple extractability removes the interfaces of NewY and Sign. Furthermore, we show that if the adaptor signature scheme is also *uniquely extractable* then it also satisfies strong full extractability (Theorem 3.4). Roughly, unique extractability says that with access to an oracle pSign, an adversary cannot find two valid signatures that both extracts.

New privacy notion: unlinkability. Unlinkability requires adapted signatures, using pSign and Adapt, to be indistinguishable from honestly generated signatures from Sign, even with adversarial access to pre-signatures and signatures. Formally, consider the following game $\mathbf{G}^{\text{unlink}}_{\mathsf{aSig},\mathcal{A}}$.

Game $\mathbf{G}^{\text{unlink}}_{\mathsf{aSig},\mathcal{A}}(\lambda)$
1 $b \leftarrow \{0,1\}$
2 $(\text{pk}, \text{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$
3 $b' \leftarrow \mathcal{A}^{\text{SignChl},\text{Sign},\text{pSign}}(\text{pk})$
4 Return $(b = b')$

SignChl$(m, (Y, y))$:
5 Assert $((Y, y) \in R)$
6 $\hat{\sigma} \leftarrow \mathsf{pSign}(\text{sk}, m, Y)$
7 $\sigma_0 \leftarrow \mathsf{Adapt}(\text{pk}, \hat{\sigma}, y)$
8 $\sigma_1 \leftarrow \mathsf{Sign}(\text{sk}, m)$
9 Return σ_b

Sign(m):
10 $\sigma \leftarrow \mathsf{Sign}(\text{sk}, m)$
11 Return σ

pSign(m, Y):
12 $\hat{\sigma} \leftarrow \mathsf{pSign}(\text{sk}, m, Y)$
13 Return $\hat{\sigma}$

We define the unlink-advantage of an adversary $\mathcal{A}$ against adaptor signature scheme aSig to be $\mathbf{Adv}^{\text{unlink}}_{\mathsf{aSig},\mathcal{A}}(\lambda) = 2\Pr[\mathbf{G}^{\text{unlink}}_{\mathsf{aSig},\mathcal{A}}(\lambda)] - 1$. We say that scheme aSig is (1) unlinkable if $\mathbf{Adv}^{\text{unlink}}_{\mathsf{aSig},\mathcal{A}}(\lambda)$ is negligible for efficient adversaries (2) perfectly unlinkable if the unlink advantage is 0 for any adversary.

We briefly explain how unlinkability guarantees on-chain privacy for the application of atomic swaps. In atomic swaps, Alice and Bob aim to atomically post signature σ_A (for some message m_A) and σ_B (for some message m_B) to a public ledger. To do this, Bob would first generate a pair (Y, y) and gives Alice his pre-signature $\hat{\sigma}_B \leftarrow \mathsf{pSign}(\text{sk}_B, m_B, Y)$. Alice will verify the validity of such pre-signature before giving her pre-signature $\hat{\sigma}_A \leftarrow \mathsf{pSign}(\text{sk}_A, m_A, Y)$ to Bob. Now, Bob can adapt the pre-signature of Alice to a valid signature, via $\sigma_A \leftarrow \mathsf{Adapt}(\text{pk}_A, \hat{\sigma}_A, y)$, and post to the ledger. But if Bob does so, then Alice can extract witness y via $y \leftarrow \mathsf{Ext}(Y, \hat{\sigma}_A, \sigma_A)$ and adapt the pre-signature of Bob to a valid signature using witness y. Unlinkability of the adaptor signature scheme ensures that the adapted signatures σ_A and σ_B to be indistinguishable from honestly generated signatures. Hence, an outside observer cannot deduce that σ_A and σ_B are "linked."

3.1 Relations with previous notions

Restating aEUF-CMA and aWitExt notions of [2]. Their security notions are defined as games where the adversary is run in two stages. We note that it is not clear (in their pseudocode) if the first and second stage of the adversaries are allowed to share any state. Hence, we take the stronger interpretation that implicit state-sharing is allowed. To keep the presentation consistent, we slightly rewrite their security games (while preserving the semantics, assuming the state of first stage adversary is passed to the second stage) so that there is only a single stage. The functionality of the two stages is instead realized via a challenge oracle that can only be called once[2]. Formally, we introduce the notions of [2] by considering games $\mathbf{G}^{\text{aEUFCMA}}$ and $\mathbf{G}^{\text{aWitExt}}$ given in Figure 4.

[2]One can think of the "first stage" as everything leading up to the challenge oracle call and the "second stage" being everything following the challenge oracle call

Game $\mathbf{G}^{\text{aEUFCMA}}_{\mathsf{aSig},\mathcal{A}}(\lambda)$

1 $(\text{pk}, \text{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$
2 $(Y^*, y^*) \leftarrow \mathsf{Gen}(1^\lambda)$
3 $\sigma^* \leftarrow \mathcal{A}^{\text{Sign,pSign,pSignChl}}(\text{pk})$
4 Return $(\mathsf{Vrf}(\text{pk}, m^*, \sigma^*) \wedge m^* \notin S)$

pSignChl(m^*): // Exactly once
5 $\hat{\sigma} \leftarrow \mathsf{pSign}(\text{sk}, m, Y^*)$; Return $(\hat{\sigma}, Y^*)$

Sign(m):
6 $S \leftarrow S \cup \{m\}$
7 Return $\sigma \leftarrow \mathsf{Sign}(\text{sk}, m)$

pSign(m, Y):
8 $S \leftarrow S \cup \{m\}$
9 Return $\hat{\sigma} \leftarrow \mathsf{pSign}(\text{sk}, m, Y)$

Game $\mathbf{G}^{\text{aWitExt}}_{\mathsf{aSig},\mathcal{A}}(\lambda)$

1 $(\text{pk}, \text{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$
2 $\sigma^* \leftarrow \mathcal{A}^{\text{Sign,pSign,pSignChl}}(\text{pk})$
3 Assert $(\mathsf{Vrf}(\text{pk}, m^*, \sigma^*) \wedge m^* \notin S)$
4 Return $((Y^*, \mathsf{Ext}(Y^*, \hat{\sigma}^*, \sigma^*)) \notin R)$

pSignChl(m^*, Y^*): // Exactly once
5 Return $\hat{\sigma}^* \leftarrow \mathsf{pSign}(\text{sk}, m, Y^*)$

Sign(m):
6 $S \leftarrow S \cup \{m\}$
7 Return $\sigma \leftarrow \mathsf{Sign}(\text{sk}, m)$

pSign(m, Y):
8 $S \leftarrow S \cup \{m\}$
9 Return $\hat{\sigma} \leftarrow \mathsf{pSign}(\text{sk}, m, Y)$

Figure 4: Games defining aEUF-CMA and aWitExt notions [2] for adaptor signature scheme aSig.

Full extractability implies previous notions. We first show that our notion of full extractability implies both aEUF-CMA and aWitExt. Specifically, we show that given any adversary attacking aEUF-CMA or aWitExt, then we can give an adversary attacking FExt. Notice that an aEUF-CMA adversary or aWitExt adversary specifies a message m^* and eventually returns σ^*. We will construct a FExt adversary that (1) forwards all Sign and pSign queries (2) simulates pSignChl oracle with pSign while recording m^* and (3) returns exactly m^*, σ^* at the end. The proof below checks that these adversaries win the FExt game whenever the starting adversary wins aEUF-CMA game or aWitExt game.

Theorem 3.1 (FExt $\implies$ aEUF-CMA + aWitExt) *Let $\mathcal{A}_{\text{aEUFCMA}}$ and $\mathcal{A}_{\text{aWitExt}}$ be aEUF-CMA and aWitExt adversaries. The proof constructs adversaries $\mathcal{A}_{\text{fext},1}$ and $\mathcal{A}_{\text{fext},2}$, which have the same running time as the given adversaries, such that*

$$\Pr[\mathbf{G}^{\text{aEUFCMA}}_{\mathsf{aSig},\mathcal{A}_{\text{aEUFCMA}}}(\lambda)] = \mathbf{Adv}^{\text{fext}}_{\mathsf{aSig},\mathcal{A}_{\text{fext},1}}(\lambda)\ , \tag{1}$$

$$\Pr[\mathbf{G}^{\text{aWitExt}}_{\mathsf{aSig},\mathcal{A}_{\text{aWitExt}}}(\lambda)] = \mathbf{Adv}^{\text{fext}}_{\mathsf{aSig},\mathcal{A}_{\text{fext},2}}(\lambda)\ . \tag{2}$$

Proof of Theorem 3.1: Consider the following adversaries $\mathcal{A}_{\text{fext},1}$ and $\mathcal{A}_{\text{fext},2}$.

Adversary $\mathcal{A}^{\text{NewY,Sign,pSign}}_{\text{fext},1}(\text{pk})$:
1 $\sigma^* \leftarrow \mathcal{A}^{\text{Sign,pSign,pSignChl}}_{\text{aEUFCMA}}(\text{pk})$
2 Return (m^*, σ^*)

pSignChl(m):
3 $m^* \leftarrow m$; $Y^* \leftarrow \text{NewY}()$
4 Return $\hat{\sigma}^* \leftarrow \text{pSign}(m^*, Y^*)$

Adversary $\mathcal{A}^{\text{NewY,Sign,pSign}}_{\text{fext},2}(\text{pk})$:
1 $\sigma^* \leftarrow \mathcal{A}^{\text{Sign,pSign,pSignChl}}_{\text{aWitExt}}(\text{pk})$
2 Return (m^*, σ^*)

pSignChl(m, Y):
3 $m^* \leftarrow m$; $Y^* \leftarrow Y$
4 Return $\hat{\sigma}^* \leftarrow \text{pSign}(m^*, Y^*)$

We check that the given adversaries wins if the given reduction adversary does. For aEUF-CMA, $\mathcal{A}_{\text{aEUFCMA}}$ wins if m^* has not been queried previously to either Sign or pSign oracles of game $\mathbf{G}^{\text{aEUFCMA}}$. Furthermore, for the single pSignChl query that $\mathcal{A}_{\text{aEUFCMA}}$ makes, our reduction

adversary $\mathcal{A}_{\text{fext},1}$ uses an instance Y^* from NewY to derive a presignature σ^*. Therefore, for $\mathcal{A}_{\text{fext},1}$ the table at m^* is a singleton set, i.e. $T[m^*] = \{(Y^*, \hat{\sigma}^*)\}$ for which Y^* is in the challenge set of instances C. Hence, our adversary $\mathcal{A}_{\text{fext},1}$ wins as long as signature σ^* is valid on m^*, which is exactly the condition that $\mathcal{A}_{\text{aEUFCMA}}$ needs to satisfy to win as well. This justifies (1).

A similar analysis holds for $\mathcal{A}_{\text{aWitExt}}$, namely that $T[m^*] = \{(Y^*, \hat{\sigma}^*)\}$ at the end of its execution. Notice that $\mathcal{A}_{\text{aWitExt}}$ wins if $\mathsf{Ext}(Y^*, \hat{\sigma}^*, \sigma^*)$ is not a valid extraction, which is exactly what $\mathcal{A}_{\text{fext},2}$ needs to satisfy as well to win. This justifies (2). ∎

Insufficiency of previous notions. To show that previous notions are insufficient, we give a scheme that is secure against previous notions, namely aEUF-CMA and aWitExt, but not FExt-secure. The informal intuition on our scheme is as follows. We will modify a secure adaptor signature scheme aSig so that pSign leaks (depending on a random coin flip) either (1) a one-time encryption pad C_0 or (2) a one-time encryption $C_1 = C_0 \oplus \sigma$, where σ is a signature on m. We use a PRF to derive C_0 and the signing randomness for σ so that they are consistent across different runs of pSign.

Formally, let aSig be any FE-secure adaptor signature for underlying signature scheme Sig and relation R. Suppose that $\mathsf{Sig.KeyGen}(1^\lambda)$ returns secret keys that are uniformly distributed over some set S_λ. Let $L_\lambda = \{Y \mid \exists y : (Y, y) \in [\mathsf{Gen}(1^\lambda)]\}$. Let $(\text{pk}, \cdot) \in [\mathsf{Sig.KeyGen}(1^\lambda)]$. Suppose $\mathsf{Sig.Sign}(\text{pk}, \cdot)$ uses random coins of length at most r_λ and that $[\mathsf{Sig.Sign}(\text{pk}, \cdot)] \subseteq \{0,1\}^{n_\lambda}$. Let F be a pseudo-random function of the form $F_\lambda : S_\lambda \times (\{0,1\}^* \times L_\lambda) \to \{0,1\}^{n_\lambda} \times \{0,1\}^{r_\lambda}$, where S_λ is the key space and $\{0,1\}^* \times L_\lambda$ is the input space. Consider adaptor signature scheme AS_0 given below.

Scheme AS_0

$\mathsf{pSign}(\text{sk}, m, Y)$:
1 $b \twoheadleftarrow \{0,1\}$
2 $\hat{\sigma}' \twoheadleftarrow \mathsf{aSig.pSign}(\text{sk}, m, Y)$
3 $(C_0, r) \leftarrow F_{\text{sk}}(m, Y)$
4 $\sigma \leftarrow \mathsf{Sig.Sign}(\text{sk}, m; r)$
5 $C_1 \leftarrow C_0 \oplus \sigma$
6 Return $(\hat{\sigma}, C_b)$

$\mathsf{pVrf}(\text{pk}, m, \hat{\sigma}, Y)$:
7 $(\hat{\sigma}', C) \leftarrow \hat{\sigma}$
8 Return $\mathsf{aSig.pVrf}(\text{pk}, m, \hat{\sigma}', Y)$

$\mathsf{Adapt}(\text{pk}, \hat{\sigma}, y)$:
9 $(\hat{\sigma}', C) \leftarrow \hat{\sigma}$
10 $\sigma \leftarrow \mathsf{Adapt}(\text{pk}, \hat{\sigma}', y)$
11 Return σ

$\mathsf{Ext}(\hat{\sigma}, \sigma, Y)$:
12 $(\hat{\sigma}', C) \leftarrow \hat{\sigma}$
13 $y \leftarrow \mathsf{aSig.Ext}(\hat{\sigma}', \sigma, Y)$
14 Return y

We claim that AS_0 satisfies aEUF-CMA and aWitExt (if aSig is FExt-secure and F is a secure PRF), but AS_0 is not FE-secure even if aSig is FE-secure. Intuitively, any single run of $\mathsf{pSign}(\text{sk}, m, Y)$ only leaks a presignature, as the second part of the output C_b is random. However, given any two evaluations of $(\hat{\sigma}_i, C_i) \twoheadleftarrow \mathsf{pSign}(\text{sk}, m, Y)$ for $i \in \{1,2\}$, it holds with probability $1/2$ that $\sigma = C_1 \oplus C_2$ is a valid signature on m. This breaks extractability. Since in the FE game, adversaries are allowed to query pSign any number of times for any given Y, even for the challenge instance Y_1. However, for aEUF-CMA and aWitExt, the adversary is only allowed to call pSignChl exactly once. This means that pSignChl does not leak an extra signature and the scheme can be shown to satisfy aEUF-CMA and aWitExt.

Theorem 3.2 (aEUF-CMA+aWitExt $\not\Rightarrow$ FExt) *Scheme AS_0 satisfies aEUF-CMA and aWitExt if aSig does and F is a secure PRF. In particular, for any adversary $\mathcal{A}_{\text{aEUFCMA}}$ and $\mathcal{A}_{\text{aWitExt}}$,*

the proof gives reduction adversaries $\mathcal{A}'_{\text{aEUFCMA}}$, $\mathcal{A}'_{\text{aWitExt}}$, $\mathcal{A}_{\text{prf},1}$, *and* $\mathcal{A}_{\text{prf},2}$, *all about as efficient as the starting adversaries, such that*

$$\Pr[\mathbf{G}^{\text{aEUFCMA}}_{\mathsf{AS}_0,\mathcal{A}_{\text{aEUFCMA}}}(\lambda)] \le \Pr[\mathbf{G}^{\text{aEUFCMA}}_{\mathsf{aSig},\mathcal{A}'_{\text{aEUFCMA}}}(\lambda)] + \mathbf{Adv}^{\text{prf}}_{F,\mathcal{A}_{\text{prf},1}}(\lambda)\ , \tag{3}$$

$$\Pr[\mathbf{G}^{\text{aWitExt}}_{\mathsf{AS}_0,\mathcal{A}_{\text{aWitExt}}}(\lambda)] \le \Pr[\mathbf{G}^{\text{aWitExt}}_{\mathsf{aSig},\mathcal{A}'_{\text{aWitExt}}}(\lambda)] + \mathbf{Adv}^{\text{prf}}_{F,\mathcal{A}_{\text{prf},2}}(\lambda)\ . \tag{4}$$

However, scheme AS_0 *is not Ext-secure even if* aSig *is. In particular, the proof gives efficient adversary* $\mathcal{A}_{\text{ext}}$, *that makes two queries to* pSign, *such that*

$$\mathbf{Adv}^{\text{fext}}_{\mathsf{AS}_0,\mathcal{A}_{\text{fext}}}(\lambda) = \frac{1}{2}\ . \tag{5}$$

Proof of Theorem 3.2: We first show (5). Consider the following adversary

Adversary $\mathcal{A}^{\text{NewY,pSign}}_{\text{fext}}(\text{pk})$:
1 $Y \leftarrow \text{NewY}()$; $(\hat{\sigma}_0, D_0) \leftarrow \mathsf{pSign}(m, Y)$; $(\hat{\sigma}_1, D_1) \leftarrow \mathsf{pSign}(m, Y)$
2 Return $(m, D_0 \oplus D_1)$

Note that, by the construction of $\mathsf{AS}_0.\mathsf{pSign}$, it holds with probability a half that $D_0 \oplus D_1$ is a valid signature on m. Since $\mathcal{A}_{\text{fext}}$ only queries pSign with a challenge Y, it wins as long as it produces a valid signature for m. This justifies (5).

Next, we justify (3). Consider the following games. Game $\mathbf{G}_0$ is the aEUF-CMA game of scheme AS_0. Game $\mathbf{G}_1$ changes all invocations of PRF F to a truly random function. Note that the challenge instance must be fresh for the game to return true, hence we can assume that the point (m^*, Y^*) has not been queried to the PRF before (line 7).

Game $\mathbf{G}_0$, $\boxed{\mathbf{G}_1}$

1 $(\text{pk}, \text{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$
2 $\sigma^* \leftarrow \mathcal{A}^{\text{Sign,pSign,pSignChl}}(\text{pk})$
3 Return $\mathsf{Vrf}(\text{pk}, m^*, \sigma^*) \wedge m^* \notin S$

pSignChl(m^*):
4 $(Y^*, \cdot) \leftarrow \mathsf{R.Gen}(1^\lambda)$
5 $\hat{\sigma}^* \leftarrow \mathsf{aSig.pSign}(\text{sk}, m^*, Y^*)$
6 $(C_0, r) \leftarrow \mathsf{F}(\text{sk}, (m^*, Y^*))$
7 $\boxed{C_0 \leftarrow \{0,1\}^{|n_\lambda|};\ r \leftarrow \{0,1\}^{r_\lambda}}$
8 $\sigma \leftarrow \mathsf{Sig.Sign}(\text{sk}, m^*; r)$
9 $C_1 \leftarrow C_0 \oplus \sigma$; $b \leftarrow \{0,1\}$
10 Return $((\hat{\sigma}^*, C_b), Y^*)$

Sign(m):
11 $\sigma \leftarrow \mathsf{AS}_0.\mathsf{Sign}(\text{sk}, m)$
12 $S \leftarrow S \cup \{m\}$; Return σ

pSign(m, Y):
13 $\hat{\sigma} \leftarrow \mathsf{aSig.pSign}(\text{sk}, m, Y)$
14 If $T[(m, Y)] = \perp$ then
15 $\quad T[(m, Y)] \leftarrow \mathsf{F}(\text{sk}, (m, Y))$
16 $\quad \boxed{T[(m, Y)] \leftarrow \{0,1\}^{n_\lambda + r_\lambda}}$
17 $(C_0, r) \leftarrow T[(m, Y)]$
18 $\sigma \leftarrow \mathsf{Sig.Sign}(\text{sk}, m; r)$
19 $C_1 \leftarrow C_0 \oplus \sigma$; $b \leftarrow \{0,1\}$
20 $S \leftarrow S \cup \{m\}$; Return $(\hat{\sigma}, C_b)$

It is straightforward to give a PRF-adversary whose advantage upper bounds the distance between $\mathbf{G}_0$ and $\mathbf{G}_1$, i.e.

$$\Pr[\mathbf{G}_0] \le \Pr[\mathbf{G}_1] + \mathbf{Adv}^{\text{prf}}_{\mathsf{F},\mathcal{A}_{\text{prf},1}}(\lambda)\ . \tag{6}$$

We omit the details. Finally, note that in game $\mathbf{G}_1$ all oracles pSignChl, Sign, pSign can now be simulated using corresponding oracles for the underlying scheme aSig. This means that we could construct adversary $\mathcal{A}'_{\text{aEUFCMA}}$ such that

$$\Pr[\mathbf{G}_1] \le \Pr[\mathbf{G}^{\text{aEUFCMA}}_{\mathsf{aSig},\mathcal{A}'_{\text{aEUFCMA}}}]\ . \tag{7}$$

In particular, $\mathcal{A}'_{\text{aEUFCMA}}$ needs to invoke both pSign and Sign to simulate one query to pSignSim, but the query to pSignChl that $\mathcal{A}_{\text{aEUFCMA}}$ makes can be simulated with only one query to pSignChl oracle that $\mathcal{A}'_{\text{aEUFCMA}}$ has access to. The pseudocode of $\mathcal{A}'_{\text{aEUFCMA}}$ is given below.

Adversary $\mathcal{A}'^{\text{Sign,pSign,pSignChl}}_{\text{aEUFCMA}}(\text{pk})$:
1 $\sigma^* \leftarrow\!\!\!\$\ \mathcal{A}^{\text{Sign,pSignSim,pSignChlSim}}_{\text{fext}}(\text{pk})$
2 Return σ^*

pSignChlSim(m^*):
3 $(Y^*, \cdot) \leftarrow\!\!\!\$\ \mathsf{R.Gen}(1^\lambda)$
4 $\hat{\sigma}^* \leftarrow\!\!\!\$\ \text{pSign}(\text{sk}, m^*, Y^*)$
5 $C \leftarrow\!\!\!\$\ \{0,1\}^{n_\lambda}$; Return $(\hat{\sigma}^*, C)$

pSignSim(m):
6 $\hat{\sigma} \leftarrow\!\!\!\$\ \text{pSign}(\text{sk}, m, Y)$
7 If $T[(m,Y)] = \perp$ then
8 $\quad T[(m,Y)] \leftarrow\!\!\!\$\ \{0,1\}^{n_\lambda}$
9 $\quad S[(m,Y)] \leftarrow \sigma \leftarrow\!\!\!\$\ \text{Sign}(m)$
10 $C_0 \leftarrow T[(m,Y)]$
11 $C_1 \leftarrow C_0 \oplus S[(m,Y)]$; $b \leftarrow\!\!\!\$\ \{0,1\}$
12 $S \leftarrow S \cup \{m\}$; Return $(\hat{\sigma}, C_b)$

This concludes the proof for (3). The proof for (4) and constructions of $\mathcal{A}_{\text{prf},2}$ and $\mathcal{A}'_{\text{aWitExt}}$ are extremely similar and we omit the details here. ∎

3.2 Modular proofs from simple notions

Full extractability and strong full extractability are fairly complex notions, where the adversary is given many attack interfaces. To facilitate simpler proofs and better intuitive understanding, we give a framework for proving FExt and SFExt security. In particular, we show that proofs can be modularized if a simpler extractability notion (Ext) is achieved.

Simple Extractability (Ext). We first define simple extractability (Ext). The notion eliminates some of the attack surfaces considered in FExt *without* weakening the security guaranteed. The formal definition is given in Figure 5. The adversary is given access to a pre-signature oracle pSign and, to win, must produce a valid signature that does not extract against *any* previous queries to pSign. We show that, assuming R is a hard relation, Ext implies FExt.

Theorem 3.3 (Ext + OW $\implies$ FExt) *Let* aSig *be an adaptor signature scheme for a hard relation* R*. Suppose it satisfies Ext then it also satisfies FExt. Formally, given any FExt-adversary* $\mathcal{A}_{\text{fext}}$*, we can construct* $\mathcal{A}_{\text{ext}}$ *and* $\mathcal{A}_{\text{ow}}$ *such that*

$$\mathbf{Adv}^{\text{fext}}_{\mathsf{aSig},\mathcal{A}_{\text{fext}}}(\lambda) \leq \mathbf{Adv}^{\text{ext}}_{\mathsf{aSig},\mathcal{A}_{\text{ext}}}(\lambda) + \mathbf{Adv}^{\text{q-ow}}_{\mathsf{R},\mathcal{A}_{\text{ow}}}(\lambda)\ . \tag{8}$$

If $\mathcal{A}_{\text{fext}}$ *makes* q_{Sign} *and* q_{pSign} *queries to* Sign *and* pSign*, respectively. Then* $\mathcal{A}_{\text{ext}}$ *makes* $q_{\text{Sign}}+q_{\text{pSign}}$ *queries to the* pSign *oracles. Furthermore,* $\mathcal{A}_{\text{ext}}$ *and* $\mathcal{A}_{\text{ow}}$ *are about as efficient as* $\mathcal{A}_{\text{fext}}$*.*

We first give a high-level proof sketch here before giving the full proof. The reduction adversary $\mathcal{A}_{\text{ext}}$ will need to simulate NewY and Sign, since it only has access to a pre-signature oracle. The adversary will simulate NewY itself and use pSign to simulate queries to Sign. The latter is possible due to the requirement of aSig to be canonical. In particular, each query to Sign is simulated by first sampling a fresh pair $(Y, y) \in \mathsf{R}$ and a signature is then derived using oracle pSign and Adapt algorithm. In doing so, table T for adversary $\mathcal{A}_{\text{ext}}$ becomes larger than that for $\mathcal{A}_{\text{fext}}$. However, it is not hard to see that for the forgery message m^*, the set $T[m^*]$ is the same for both $\mathcal{A}_{\text{fext}}$ and $\mathcal{A}_{\text{ext}}$, assuming $\mathcal{A}_{\text{fext}}$ wins. This is because $\mathcal{A}_{\text{fext}}$ can only win if the returned message m^* is not in the set S, which means that there were no previous queries of the form $\text{Sign}(m^*)$. Finally, we need to make sure that the forgery (m^*, σ^*) does not extract for a challenge instance $Y \in C$ (those Y that was returned by NewY). This event should not happen with high probability since we have

assumed that R is hard. Indeed, it is not hard to give a OW adversary whose OW-advantage can be used to upper-bound the probability of this event.

Proof of Theorem 3.3: The proof is via a sequence of three games, which are given below. Game $\mathbf{G}_0$ is $\mathbf{G}^{\text{fext}}_{\mathsf{aSig},\mathcal{A}}(\lambda)$. Game $\mathbf{G}_1$ derives signatures using pSign and Adapt instead of Sign.

Game $\mathbf{G}_0$, $\mathbf{G}_1$, $\mathbf{G}_2$

1 $(\text{pk}, \text{sk}) \twoheadleftarrow \mathsf{KeyGen}(1^\lambda)$
2 $(m^*, \sigma^*) \twoheadleftarrow \mathcal{A}^{\text{NewY},\text{Sign},\text{pSign}}(\text{pk})$
3 Assert $\mathsf{Vrf}(\text{pk}, m^*, \sigma^*) \wedge m^* \notin S$
4 $b_1 \leftarrow \big(\forall (Y,\hat{\sigma}) \in T[m^*] \text{ st } Y \notin C$
$\quad : (Y, \mathsf{Ext}(Y,\hat{\sigma},\sigma^*)) \notin R\big)$
5 $b_2 \leftarrow \big(\forall (Y,\hat{\sigma}) \in U[m^*] \text{ st } Y \notin C$
$\quad : (Y, \mathsf{Ext}(Y,\hat{\sigma},\sigma^*)) \notin R\big)$
6 $b_3 \leftarrow \big(\forall (Y,\hat{\sigma}) \in T[m^*] \cup U[m^*] \text{ st } Y \in C$
$\quad : (Y, \mathsf{Ext}(Y,\hat{\sigma},\sigma^*)) \notin R\big)$
7 $\mathbf{G}_0, \mathbf{G}_1$: Return b_1
8 $\mathbf{G}_2$: Return $b_1 \wedge b_2$
9 $\mathbf{G}_3$: Return $b_1 \wedge b_2 \wedge b_3$
10 $\mathbf{G}_4$: Return $b_1 \wedge b_2 \wedge \neg b_3$

NewY:
11 $(Y, \cdot) \twoheadleftarrow \mathsf{R.Gen}(1^\lambda)$; $C \xleftarrow{\cup} Y$; Return Y

Sign(m): // $\mathbf{G}_0$
12 $\sigma \twoheadleftarrow \mathsf{Sign}(\text{sk}, m)$
13 $S \leftarrow S \cup \{m\}$; Return σ

Sign(m): // $\mathbf{G}_1, \ldots, \mathbf{G}_4$
14 $(Y, y) \twoheadleftarrow \mathsf{R.Gen}(1^\lambda)$
15 $\hat{\sigma} \twoheadleftarrow \mathsf{pSign}(\text{sk}, m, Y)$
16 $U[m] \leftarrow U[m] \cup \{(Y,\hat{\sigma})\}$
17 $\sigma \leftarrow \mathsf{Adapt}(\text{pk}, \hat{\sigma}, y)$
18 $S \leftarrow S \cup \{m\}$; Return σ

pSign(m, Y):
19 $\hat{\sigma} \twoheadleftarrow \mathsf{pSign}(\text{sk}, m, Y)$
20 $T[m] \leftarrow T[m] \cup \{(Y,\hat{\sigma})\}$
21 Return $\hat{\sigma}$

Game $\mathbf{G}_1$ samples signatures by using the alternative signing algorithm involving pSign and Adapt for freshly sampled pair (Y, y). Since we know that aSig is canonical, we have

$$\Pr[\mathbf{G}_0] = \Pr[\mathbf{G}_1] \ . \tag{9}$$

Next, we would like to rewrite $\mathbf{G}_2$ so that contents of table U is treated in the same way as table T. In other words, we need to make sure that the additional elements added to table T does not change whether a forgery is valid. In particular, consider $\mathbf{G}_2$, where the game additionally checks that not only the forgery σ^* does not extract from set $T[m^*]$, but also $U[m^*]$. We argue that this change does not change the probability of the game returning true, meaning

$$\Pr[\mathbf{G}_1] = \Pr[\mathbf{G}_2] \ . \tag{10}$$

Note that line 3 ensures that $m^* \notin S$. Hence, for the forgery message m^*, the set $U[m^*]$ is the empty set, which implies that flag b_2 is trivially set to true. Moving on, we need to additionally check if the forgery signature σ^* can yield any valid extraction for challenge $Y \in C$. This condition is checked in flag b_3. We separate the winning condition of game $\mathbf{G}_2$, into two games $\mathbf{G}_3$ and $\mathbf{G}_4$, depending on the value of b_3. Clearly,

$$\Pr[\mathbf{G}_2] = \Pr[\mathbf{G}_3] + \Pr[\mathbf{G}_4] \ . \tag{11}$$

We now give adversary $\mathcal{A}_{\text{ext}}$ such that

$$\Pr[\mathbf{G}_3] \leq \mathbf{Adv}^{\text{ext}}_{\mathsf{aSig},\mathcal{A}_{\text{ext}}}(\lambda) \ . \tag{12}$$

Game $\mathbf{G}^{\mathrm{ext}}_{\mathsf{aSig},\mathcal{A}}(\lambda)$

1 $(\mathrm{pk},\mathrm{sk}) \leftarrow\!\!\ast\ \mathsf{KeyGen}(1^\lambda)$; $(m^*,\sigma) \leftarrow\!\!\ast\ \mathcal{A}^{\mathrm{pSign}}(\mathrm{pk})$; Assert $\mathsf{Vrf}(\mathrm{pk},m^*,\sigma)$
2 Return $(\forall (Y,\hat\sigma) \in T[m^*] : (Y,\mathsf{Ext}(Y,\hat\sigma,\sigma)) \notin R)$ // Extraction fails for all $Y,\hat\sigma$

pSign(m,Y):
3 $\hat\sigma \leftarrow\!\!\ast\ \mathsf{pSign}(\mathrm{sk},m,Y)$; $T[m] \xleftarrow{\cup} (Y,\hat\sigma)$; Return $\hat\sigma$

Game $\mathbf{G}^{\mathrm{uext}}_{\mathsf{aSig},\mathcal{A}}(\lambda)$

1 $(\mathrm{pk},\mathrm{sk}) \leftarrow\!\!\ast\ \mathsf{KeyGen}(1^\lambda)$; $(m,Y,\hat\sigma,\sigma,\sigma') \leftarrow\!\!\ast\ \mathcal{A}^{\mathrm{pSign}}(\mathrm{pk})$
2 Assert $(\sigma \neq \sigma' \wedge \mathsf{pVrf}(\mathrm{pk},m,\hat\sigma,Y) \wedge \mathsf{Vrf}(\mathrm{pk},m,\sigma) \wedge \mathsf{Vrf}(\mathrm{pk},m,\sigma'))$
3 $y \leftarrow \mathsf{Ext}(Y,\hat\sigma,\sigma)$; $y' \leftarrow \mathsf{Ext}(Y,\hat\sigma,\sigma')$
4 Return $((Y,y) \in R \wedge (Y,y') \in R)$ // Both extraction succeeds

pSign(m,Y):
5 Return $\hat\sigma \leftarrow\!\!\ast\ \mathsf{pSign}(\mathrm{sk},m,Y)$

Figure 5: Games defining extractability and unique extractability.

Adversary $\mathcal{A}^{\mathrm{pSign}}_{\mathrm{ext}}(\mathrm{pk})$:
1 $(m^*,\sigma^*) \leftarrow\!\!\ast\ \mathcal{A}^{\mathrm{NewY,SignSim,pSign}}_{\mathrm{fext}}(\mathrm{pk})$
2 Assert $\mathsf{Vrf}(\mathrm{pk},m^*,\sigma^*) \wedge m^* \notin S$
3 Return (m^*,σ^*)

NewY(m):
4 $(Y,\cdot) \leftarrow\!\!\ast\ \mathsf{R.Gen}(1^\lambda)$; $C \xleftarrow{\cup} Y$; Return Y

SignSim(m):
5 $(Y,y) \leftarrow\!\!\ast\ \mathsf{R.Gen}(1^\lambda)$
6 $\hat\sigma \leftarrow\!\!\ast\ \mathrm{pSign}(m,Y)$
7 $\sigma \leftarrow\!\!\ast\ \mathsf{Adapt}(\mathrm{pk},\hat\sigma,y)$
8 $S \leftarrow S \cup \{m\}$
9 Return σ

The adversary is very simple, it simulates a signing oracle SignSim for $\mathcal{A}_{\mathrm{fext}}$ and forwards other inputs and outputs accordingly. Finally, we give adversary $\mathcal{A}_{\mathrm{ow}}$ such that

$$\Pr[\mathbf{G}_4] \leq \mathbf{Adv}^{\text{q-ow}}_{\mathsf{R},\mathcal{A}_{\mathrm{ow}}}(\lambda)\ , \tag{13}$$

where q is the maximum number of queries that $\mathcal{A}_{\mathrm{fext}}$ makes to oracle NewY. Consider the following adversary $\mathcal{A}_{\mathrm{ow}}$ against q-OW.

Adversary $\mathcal{A}_{\mathrm{ow}}(Y_1,\ldots,Y_q)$:
1 $(\mathrm{pk},\mathrm{sk}) \leftarrow\!\!\ast\ \mathsf{KeyGen}(1^\lambda)$
2 $(m^*,\sigma^*) \leftarrow\!\!\ast\ \mathcal{A}^{\mathrm{NewY,SignSim,pSign}}_{\mathrm{fext}}(\mathrm{pk})$
3 Assert $\mathsf{Vrf}(\mathrm{pk},m^*,\sigma^*) \wedge m^* \notin S$
4 For $((Y,\hat\sigma) \in T[m^*]$ st $\exists I \in [q] : Y = Y_I)$ do
5 $\quad y \leftarrow \mathsf{Ext}(Y,\hat\sigma,\sigma^*)$
6 $\quad$ If $(Y,y) \in \mathsf{R}$ then Return (I,y)
7 Return $\perp$

NewY(m):
8 $i \leftarrow i+1$; Return Y_i

Sign(m):
9 $(Y,y) \leftarrow\!\!\ast\ \mathsf{R.Gen}(1^\lambda)$
10 $\hat\sigma \leftarrow\!\!\ast\ \mathsf{pSign}(\mathrm{sk},m,Y)$
11 $\sigma \leftarrow\!\!\ast\ \mathsf{Adapt}(\mathrm{pk},\hat\sigma,y)$
12 $T[m] \xleftarrow{\cup} (Y,\hat\sigma)$
13 $S \leftarrow S \cup \{m\}$
14 Return σ

pSign(m,Y):
15 $\hat\sigma \leftarrow\!\!\ast\ \mathsf{pSign}(\mathrm{sk},m,Y)$
16 $T[m] \xleftarrow{\cup} (Y,\hat\sigma)$
17 Return $\hat\sigma$

Equation (8) is obtained by combining the above inequalities. ∎

Next, we show that if aSig is shown to satisfy Ext, then we can also show that it satisfies SFExt security if an additional security notion, which is conceptually simple and easy to verify, is satisfied.

SFExt from unique extractability (uExt). Unique extractability requires that an adversary, with access to a pre-signature oracle, cannot find $(m, Y, \hat{\sigma}, \sigma, \sigma')$ where σ, σ' are two distinct valid signatures on message m that also both extracts against instance Y and pre-signature $\hat{\sigma}$. Formally, consider the game $\mathbf{G}^{\text{uext}}$ given in Figure 5. The adversary has access to a pre-signature oracle returning honestly generated pre-signatures. The adversary wins if it successfully finds two distinct valid signatures σ, σ' that both extracts. We define the uExt advantage of an adversary $\mathcal{A}$ to be $\mathbf{Adv}^{\text{uext}}_{\mathsf{aSig},\mathcal{A}}(\lambda)$. We say that aSig satisfies uExt if the advantage of any efficient adversary is negligible. We show that if an adaptor signature for a hard relation R satisfies Ext, and uExt, then it must also satisfy SFExt.

Theorem 3.4 (Ext + OW + uExt $\implies$ SFExt) *Let* aSig *be a canonical adaptor signature scheme for a hard relation* R*. Suppose it satisfies Ext and uExt, then it also satisfies SFExt. Formally, given any SFExt-adversary* $\mathcal{A}_{\text{sfext}}$*, we can construct* $\mathcal{A}_{\text{ow}}$*,* $\mathcal{A}_{\text{ext}}$*,* $\mathcal{A}_{\text{uext}}$ *such that*

$$\mathbf{Adv}^{\text{sfext}}_{\mathsf{aSig},\mathcal{A}_{\text{sfext}}}(\lambda) \leq \mathbf{Adv}^{\text{ext}}_{\mathsf{aSig},\mathcal{A}_{\text{ext}}}(\lambda) + \mathbf{Adv}^{\text{q-ow}}_{\mathsf{R},\mathcal{A}_{\text{ow}}}(\lambda) + \mathbf{Adv}^{\text{uext}}_{\mathsf{aSig},\mathcal{A}_{\text{uext}}}(\lambda)\ . \tag{14}$$

If $\mathcal{A}_{\text{sfext}}$ *makes* q_{Sign} *and* q_{pSign} *queries to* Sign *and* pSign*, respectively. Then* $\mathcal{A}_{\text{ext}}$ *and* $\mathcal{A}_{\text{uext}}$ *makes* $q_{\text{Sign}} + q_{\text{pSign}}$ *queries to their* pSign *oracles. Furthermore, all adversaries are about as efficient as* $\mathcal{A}_{\text{sfext}}$*.*

We first give a high-level proof sketch here before giving the full proof. Similar to before, the reduction adversary $\mathcal{A}_{\text{ext}}$ will need to simulate oracles NewY and Sign, which is done exactly as in the proof of Theorem 3.3. However, for SFExt, we can no longer assume that table $T[m^*]$ is the same for our reduction adversary. This is because $\mathcal{A}_{\text{sfext}}$ can return a valid forgery message m^* for which there were previous queries of the form $\text{Sign}(m^*)$. However, since for all simulated Sign queries, the reduction adversary $\mathcal{A}_{\text{ext}}$ already knows a valid signature that extracts (the signature that is returned at the end of the Sign oracle call), we can use the notion of unique extractability to upper bound the probability that the forgery signature σ^* also extracts. On the other hand, if the forgery does not extract, then our reduction adversary $\mathcal{A}_{\text{fext}}$ will win by simply forwarding the forgery (m^*, σ^*). Finally, a reduction to one-wayness of R is done similarly as before to bound the probability that extraction succeeds for some $Y \in C$.

In upshot, to prove FExt or SFExt, we can first show that a scheme aSig satisfies Ext, which implies FExt security. If aSig is shown to additionally satisfy uExt, then we know that aSig is also SFExt-secure.

Proof of Theorem 3.4: The proof is via a sequence of five games, which are given below. Game $\mathbf{G}_0$ is $\mathbf{G}^{\text{sfext}}_{\mathsf{aSig},\mathcal{A}}(\lambda)$. Game $\mathbf{G}_1$ derives signatures using pSign and Adapt instead of Sign. Game $\mathbf{G}_2$ to $\mathbf{G}_4$ rewrites the winning conditions of game $\mathbf{G}_1$.

Game $\mathbf{G}_0$ – $\mathbf{G}_4$

1 $(\mathrm{pk}, \mathrm{sk}) \twoheadleftarrow \mathsf{KeyGen}(1^\lambda)$
2 $(m^*, \sigma^*) \twoheadleftarrow \mathcal{A}_{\mathrm{fext}}^{\mathrm{Sign},\mathrm{pSign}}(\mathrm{pk}, Y_1)$
3 Assert $((m^*, \sigma^*) \notin U \wedge \mathsf{Vrf}(\mathrm{pk}, m^*, \sigma^*))$
4 $b_1 \leftarrow \big(\forall (Y, \hat{\sigma}) \in T[m^*]$ st $Y \notin C$
$\quad : (Y, \mathsf{Ext}(Y, \hat{\sigma}, \sigma^*)) \notin R\big)$
5 $b_2 \leftarrow \big(\forall (Y, \hat{\sigma}) \in V[m^*]$ st $Y \notin C$
$\quad : (Y, \mathsf{Ext}(Y, \hat{\sigma}, \sigma^*)) \notin R\big)$
6 $b_3 \leftarrow \big(\forall (Y, \hat{\sigma}) \in T[m^*] \cup U[m^*]$ st $Y \in C$
$\quad : (Y, \mathsf{Ext}(Y, \hat{\sigma}, \sigma^*)) \notin R\big)$
7 $\mathbf{G}_0, \mathbf{G}_1$: Return b_1
8 $\mathbf{G}_2$: Return $\neg b_2$
9 $\mathbf{G}_3$: Return $\neg b_3$
10 $\mathbf{G}_4$: Return $b_1 \wedge b_2 \wedge b_3$

NewY:
11 $(Y, \cdot) \twoheadleftarrow \mathsf{R.Gen}(1^\lambda)$
12 $C \xleftarrow{\cup} Y$; Return Y

Sign(m): // $\mathbf{G}_0$
13 $\sigma \twoheadleftarrow \mathsf{Sign}(\mathrm{sk}, m)$
14 $S \leftarrow S \cup \{m\}$; Return σ

Sign(m): // $\mathbf{G}_1$ to $\mathbf{G}_5$
15 $(Y, y) \twoheadleftarrow \mathsf{R.Gen}(1^\lambda)$
16 $\hat{\sigma} \twoheadleftarrow \mathsf{pSign}(\mathrm{sk}, m, Y)$
17 $V[m] \leftarrow V[m] \cup \{(Y, \hat{\sigma})\}$
18 $\sigma \leftarrow \mathsf{Adapt}(\mathrm{pk}, \hat{\sigma}, y)$
19 $U \leftarrow U \cup \{(m, \sigma)\}$
20 Return σ

pSign(m, Y):
21 $\hat{\sigma} \twoheadleftarrow \mathsf{pSign}(\mathrm{sk}, m, Y)$
22 $T[m] \leftarrow T[m] \cup \{(Y, \hat{\sigma})\}$
23 Return $\hat{\sigma}$

Game $\mathbf{G}_1$ samples signatures by using the alternative signing algorithm involving pSign and Adapt for freshly sampled pair (Y, y). Since we know that aSig is canonical, we have

$$\Pr[\mathbf{G}_0] = \Pr[\mathbf{G}_1] \ . \tag{15}$$

Next, we split winning condition of $\mathbf{G}_1$ into depending on bit b_2 and b_3, whose probability of being false are captured as game $\mathbf{G}_2$ and $\mathbf{G}_3$. We have

$$\begin{aligned}
\Pr[\mathbf{G}_1] &= \Pr[b_1] && (16)\\
&= \Pr[b_1 \wedge b_2] + \Pr[b_1 \wedge \neg b_2] && (17)\\
&\leq \Pr[b_1 \wedge b_2] + \Pr[\neg b_2] && (18)\\
&= \Pr[b_1 \wedge b_2 \wedge b_3] + \Pr[b_1 \wedge b_2 \wedge \neg b_3] + \Pr[\neg b_2] && (19)\\
&\leq \Pr[b_1 \wedge b_2 \wedge b_3] + \Pr[\neg b_3] + \Pr[\neg b_2] && (20)\\
&= \Pr[\mathbf{G}_4] + \Pr[\mathbf{G}_2] + \Pr[\mathbf{G}_3] \ . && (21)
\end{aligned}$$

We first analyze game $\mathbf{G}_2$. Intuitively, if b_2 is false, then there is some $(Y, \hat{\sigma}) \in V[m^*]$ such that $\mathsf{Ext}(Y, \hat{\sigma}, \sigma^*)$ is a valid witness for Y. In this case, we will have obtained two signatures that are both valid and extracts, for the same pair $(Y, \hat{\sigma})$—one being the signature returned in Sign and the other being σ^*. Hence, we will be able to break uExt. We give adversary $\mathcal{A}_{\mathrm{uext}}$ such that

$$\Pr[\mathbf{G}_2] \leq \mathbf{Adv}^{\mathrm{uext}}_{\mathsf{aSig},\mathcal{A}_{\mathrm{uext}}}(\lambda) \ . \tag{22}$$

Adversary $\mathcal{A}_{\mathrm{uext}}$ simulates a signing oracle SignSim for $\mathcal{A}_{\mathrm{sfext}}$ and forwards other inputs and outputs accordingly. At the end, it finds if the forgery (m^*, σ^*) extracts under a previously derived $(Y, \hat{\sigma})$. If it does, then it returns $(m, Y, \hat{\sigma}, \sigma^*, \sigma)$ where σ is the signature generated by SignSim for m^* (with pre-signature $\hat{\sigma}$).

Adversary $\mathcal{A}_{\text{uext}}^{\text{pSign}}(\text{pk})$:
1 $(m^*, \sigma^*) \leftarrow\!\!\$\ \mathcal{A}_{\text{fext}}^{\text{SignSim},\text{pSign}}(\text{pk}, Y_1)$; Assert $(m^* \notin S)$
2 If $(\exists (Y, \hat{\sigma}, \sigma) \in V[m^*] : (Y, \mathsf{Ext}(Y, \hat{\sigma}, \sigma^*)) \in R)$ then
3 Return $(m, Y, \hat{\sigma}, \sigma, \sigma^*)$

NewY:
4 $(Y, \cdot) \leftarrow\!\!\$\ \mathsf{R.Gen}(1^\lambda)$; $C \xleftarrow{\cup} Y$; Return Y

SignSim(m):
5 $(Y, y) \leftarrow\!\!\$\ \mathsf{R.Gen}(1^\lambda)$; $\hat{\sigma} \leftarrow\!\!\$\ \text{pSign}(m, Y)$; $\sigma \leftarrow\!\!\$\ \mathsf{Adapt}(\text{pk}, \hat{\sigma}, y)$
6 $V[m] \leftarrow V[m] \cup \{(Y, \hat{\sigma}, \sigma)\}$; $U \leftarrow U \cup \{(m, \sigma)\}$; Return σ

Next, we analyze game $\mathbf{G}_3$. We will construct $\mathcal{A}_{\text{ext}}$ and $\mathcal{A}_{\text{ow}}$ such that

$$\Pr[\mathbf{G}_3] \leq \mathbf{Adv}_{\mathsf{R},\mathcal{A}_{\text{ow}}}^{\text{q-ow}}(\lambda) \ , \tag{23}$$

where q is the maximum number of queries that $\mathcal{A}_{\text{sfext}}$ makes to oracle NewY. Adversary $\mathcal{A}_{\text{ow}}$ against q-OW is given as follows.

Adversary $\mathcal{A}_{\text{ow}}(Y_1, \ldots, Y_q)$:
1 $(\text{pk}, \text{sk}) \leftarrow\!\!\$\ \mathsf{KeyGen}(1^\lambda)$
2 $(m^*, \sigma^*) \leftarrow\!\!\$\ \mathcal{A}_{\text{fext}}^{\text{NewY},\text{Sign},\text{pSign}}(\text{pk})$
3 Assert $\mathsf{Vrf}(\text{pk}, m^*, \sigma^*) \wedge m^* \notin S$
4 For $((Y, \hat{\sigma}) \in T[m^*]$ st $\exists I \in [q] : Y = Y_I)$ do
5 $y \leftarrow \mathsf{Ext}(Y, \hat{\sigma}, \sigma^*)$
6 If $(Y, y) \in \mathsf{R}$ then Return (I, y)
7 Return $\perp$

NewY(m):
8 $i \leftarrow i + 1$; Return Y_i

Sign(m):
9 $(Y, y) \leftarrow\!\!\$\ \mathsf{R.Gen}(1^\lambda)$
10 $\hat{\sigma} \leftarrow\!\!\$\ \mathsf{pSign}(\text{sk}, m, Y)$
11 $\sigma \leftarrow\!\!\$\ \mathsf{Adapt}(\text{pk}, \hat{\sigma}, y)$
12 $T[m] \xleftarrow{\cup} (Y, \hat{\sigma})$
13 $S \leftarrow S \cup \{m\}$
14 Return σ

pSign(m, Y):
15 $\hat{\sigma} \leftarrow\!\!\$\ \mathsf{pSign}(\text{sk}, m, Y)$
16 $T[m] \xleftarrow{\cup} (Y, \hat{\sigma})$
17 Return $\hat{\sigma}$

Finally, we bound the probability of $\mathbf{G}_4$ returning true. Specifically, we construct $\mathcal{A}_{\text{ext}}$ such that

$$\Pr[\mathbf{G}_4] \leq \mathbf{Adv}_{\mathsf{aSig},\mathcal{A}_{\text{ext}}}^{\text{ext}}(\lambda) \ . \tag{24}$$

Adversary $\mathcal{A}_{\text{ext}}$ simulates a signing oracle SignSim for $\mathcal{A}_{\text{fext}}$ and forwards other inputs and outputs accordingly.

Adversary $\mathcal{A}_{\text{ext}}^{\text{pSign}}(\text{pk})$:
1 $(m^*, \sigma^*) \leftarrow\!\!\$\ \mathcal{A}_{\text{sfext}}^{\text{NewY},\text{SignSim},\text{pSign}}(\text{pk})$
2 Assert $((m^*, \sigma^*) \notin U)$
3 Return (m^*, σ^*)

NewY(m):
4 $(Y, y) \leftarrow\!\!\$\ \mathsf{R.Gen}(1^\lambda)$; $C \xleftarrow{\cup} Y$; Return Y

SignSim(m):
5 $(Y, y) \leftarrow\!\!\$\ \mathsf{R.Gen}(1^\lambda)$
6 $\hat{\sigma} \leftarrow\!\!\$\ \text{pSign}(m, Y)$
7 $\sigma \leftarrow\!\!\$\ \mathsf{Adapt}(\text{pk}, \hat{\sigma}, y)$
8 $U \leftarrow U \cup \{(m, \sigma)\}$
9 Return σ

Equation (14) is obtained by combining the above inequalities. ∎

4 Generic Constructions

In this section, we give generic constructions of adaptor signatures from *any* signature scheme Sig and *any* hard relation R. The reason we use the word "from" but not "for" is that the generated signature σ' is not exactly a standard signature σ for Sig. However, the verification of σ' will require

Scheme	Assumptions		Security		
	on Sig	on R	FExt	SFExt	Unlink
GAS_1	UF-CMA	OW	✓	×	×
GAS_1	SUF-CMA	OW, uWit	✓	✓	×
GAS_2	UF-CMA	OW, SRSR	✓	×	✓
GAS_2	SUF-CMA	OW, SRSR, uWit	✓	✓	✓

Figure 6: Table comparing constructions and their instantiations.

minimal modification to verification of σ, in that it only additionally perform a membership check of relation R.

In more detail, our adaptor signature schemes generate signatures σ' that are a combination of a standard signature σ from Sig and a pair (Y, y) from R, i.e. $\sigma' = (\sigma, Y, y)$. Additionally, the verification must perform checks of the standard signature σ as well as a membership check that $(Y, y) \in \mathsf{R}$. Hence, in terms of applications, our adaptor signatures can be supported by the blockchain as only as it supports signature Sig and relation $\mathbb{R}$ as well as basic scripting capabilities. For example, if Sig is taken to be ECDSA over secp256k1 and R is taken to be the relation induced by hashing 256-bit inputs with sha256, then SigR can be realized via a Bitcoin script, which coincide with the construction of a "hash-lock" contract. We first formalize such an "augmented" signature scheme.

Augmented signature schemes SigR. Let Sig be any signature scheme and R be any hard relation. Roughly, the augmented signature scheme SigR is a signature scheme whose signatures additionally (1) attest to an instance Y alongside a message m and (2) releases a valid witness y of instance Y. Formally, the signing and verification algorithms are given in Figure 7 (key generation is unchanged).

Our constructions can be seen as a generalization to "hash-lock" contracts (coined and used by the Bitcoin Lightning network [20]). We study the notion of adaptor signatures in a general setting where there is no restriction on the underlying signatures schemes. We remark that implementing augmented signature schemes for existing blockchains such as Bitcoin require usage of "scripts." We first present a construction GAS_1 that achieves all security properties, but unlinkability.

Generic Adaptor Signature (GAS) 1. Let Sig be any signature scheme and any hard relation R. Consider construction GAS_1 given in Figure 7. We show that GAS_1 satisfies FExt as long as Sig is Unforgeable, and additionally SFExt if Sig is strongly unforgeable and R has unique witnesses.

Theorem 4.1 *Adaptor signature scheme GAS_1 satisfies correctness and pre-signature adaptability. If Sig is UF-CMA-secure and R is one-way then GAS_1 is FExt-secure. Given adversary $\mathcal{A}_{\text{fext}}$, we can construct adversaries $\mathcal{A}_{\text{uf-cma}}$ and $\mathcal{A}_{\text{ow}}$, with running times similar to that of $\mathcal{A}_{\text{fext}}$ such that*

$$\mathbf{Adv}^{\text{fext}}_{\mathsf{GAS}_1,\mathcal{A}_{\text{fext}}}(\lambda) \leq \mathbf{Adv}^{\text{uf-cma}}_{\mathsf{Sig},\mathcal{A}_{\text{uf-cma}}}(\lambda) + \mathbf{Adv}^{\text{q-ow}}_{R,\mathcal{A}_{\text{ow}}}(\lambda)\ . \tag{25}$$

Furthermore, if Sig is SUF-CMA-secure and R has unique witnesses, then GAS_1 is SFExt-secure. Formally, given adversary $\mathcal{A}_{\text{sfext}}$, we can construct adversaries $\mathcal{A}_{\text{suf-cma}}$, $\mathcal{A}_{\text{ow}}$, and $\mathcal{A}_{\text{uwit}}$ with running times similar to that of $\mathcal{A}_{\text{sfext}}$ such that

$$\mathbf{Adv}^{\text{sfext}}_{\mathsf{GAS}_1,\mathcal{A}_{\text{sfext}}}(\lambda) \leq 2\mathbf{Adv}^{\text{suf-cma}}_{\mathsf{Sig},\mathcal{A}_{\text{uf-cma}}}(\lambda) + \mathbf{Adv}^{\text{q-ow}}_{R,\mathcal{A}_{\text{ow}}}(\lambda) + \mathbf{Adv}^{\text{uwit}}_{\mathsf{R},\mathcal{A}_{\text{uwit}}}\ . \tag{26}$$

We give a rough proof intuition first before giving the full proof. Correctness and adaptability are straightforward to check. We rely on Theorem 3.3 and Theorem 3.4 so that we only need to

Scheme SigR

$\mathsf{Sign}(\mathrm{sk}, m)$:
1 $(Y, y) \twoheadleftarrow R.\mathsf{Gen}(1^\lambda)$
2 $\sigma \twoheadleftarrow \mathsf{Sig.Sign}(\mathrm{sk}, (m, Y))$
3 Return (σ, Y, y)

$\mathsf{Vrf}(\mathrm{pk}, m, \sigma')$:
4 $(\sigma, Y, y) \leftarrow \sigma'$
5 Return $(\mathsf{Sig.Vrf}(\mathrm{pk}, (m, Y), \sigma) \wedge (Y, y) \in R)$

Scheme GAS_1

$\mathsf{pSign}(\mathrm{sk}, m, Y)$:
1 $\sigma' \twoheadleftarrow \mathsf{Sig.Sign}(\mathrm{sk}, (m, Y))$
2 Return (σ', Y)

$\mathsf{pVrf}(\mathrm{pk}, m, \hat{\sigma}, Y)$:
3 Return $\mathsf{Sig.Vrf}(\mathrm{pk}, (m, Y), \hat{\sigma})$

$\mathsf{Adapt}(\mathrm{pk}, \hat{\sigma}, y)$:
4 $(\sigma', Y) \leftarrow \hat{\sigma}$; $\sigma \leftarrow (\sigma', Y, y)$
5 Return σ

$\mathsf{Ext}(\hat{\sigma}, \sigma, Y)$:
6 $(\sigma', Y, y) \leftarrow \sigma$
7 Return y

Scheme GAS_2

$\mathsf{pSign}(\mathrm{sk}, m, Y)$:
1 $r \twoheadleftarrow R.R_\lambda$; $Y' \leftarrow R.A(Y, r)$
2 $\sigma' \twoheadleftarrow \mathsf{Sig.Sign}(\mathrm{sk}, (m, Y'))$
3 Return (σ', Y, r)

$\mathsf{pVrf}(\mathrm{pk}, m, \hat{\sigma}, Y)$:
4 $(\sigma', \cdot, r) \leftarrow \hat{\sigma}$; $Y' \leftarrow R.A(Y, r)$
5 Return $\mathsf{Sig.Vrf}(\mathrm{pk}, (m, Y'), \sigma')$

$\mathsf{Adapt}(\mathrm{pk}, \hat{\sigma}, y)$:
6 $(\sigma', Y, r) \leftarrow \hat{\sigma}$; $y' \leftarrow R.B(y, r)$
7 $Y' \leftarrow \mathsf{R}.A(Y, r)$
8 Return (σ', Y', y')

$\mathsf{Ext}(\hat{\sigma}, \sigma, Y)$:
9 $(\sigma', Y, r) \leftarrow \hat{\sigma}$; $(\sigma', Y', y') \leftarrow \sigma$
10 $y \leftarrow R.C(y', r)$
11 Return y

Figure 7: Top: Augmented signature scheme SigR for any signature scheme Sig and relation R. Bottom: adaptor signature schemes GAS_1 and GAS_2 for signature scheme SigR.

---

verify Ext- and uExt-security of GAS_1. It is not hard to verify that extractability follows from the unforgeability of Sig. For unique extractability, it is not hard to see that we at least need to assume strong extractability of Sig and that R has unique witnesses, since otherwise SigR is not strongly unforgeable. It turns out that these assumptions are also sufficient to show unique extractability.

Proof of Theorem 4.1: First, correctness holds by construction. Next, we check adaptability. Let $(\mathrm{pk}, \mathrm{sk}) \in [\mathsf{KeyGen}(1^\lambda)]$ and $m \in \{0,1\}^*$. Let $\hat{\sigma}, (Y, y)$ be such that $(Y, y) \in R$ and $\mathsf{pVrf}(\mathrm{pk}, m, \hat{\sigma}, Y) = \mathsf{True}$. This means that $\hat{\sigma} = (\sigma, Y)$ and $\mathsf{Sig.Vrf}(\mathrm{pk}, (m, Y), \sigma) = \mathsf{True}$. Hence, by the verification of SigR, it must be that $\mathsf{SigR.Vrf}(\mathrm{pk}, m, (\sigma, Y, y)) = \mathsf{True}$.

We move on to FExt and SFExt. With the help of Theorem 3.3 and Theorem 3.4, we simply need to show that for any adversary $\mathcal{A}_{\mathrm{ext}}$ and $\mathcal{A}_{\mathrm{uext}}$,

$$\mathbf{Adv}^{\mathrm{ext}}_{\mathsf{GAS}_1, \mathcal{A}_{\mathrm{ext}}}(\lambda) \leq \mathbf{Adv}^{\text{uf-cma}}_{\mathsf{Sig}, \mathcal{A}_{\text{uf-cma}}}(\lambda) \,, \tag{27}$$

$$\mathbf{Adv}^{\mathrm{uext}}_{\mathsf{GAS}_1, \mathcal{A}_{\mathrm{uext}}}(\lambda) \leq \mathbf{Adv}^{\text{suf-cma}}_{\mathsf{Sig}, \mathcal{A}_{\text{suf-cma}}}(\lambda) + \mathbf{Adv}^{\mathrm{uwit}}_{\mathsf{R}, \mathcal{A}_{\mathrm{uwit}}}(\lambda) \,, \tag{28}$$

where $\mathcal{A}_{\text{uf-cma}}, \mathcal{A}_{\text{suf-cma}}, \mathcal{A}_{\mathrm{uwit}}$ are reduction adversaries to be constructed.

We first show (27). Consider the following game $\mathbf{G}_0$ and adversary $\mathcal{A}_{\text{uf-cma}}$.

Game $\mathbf{G}_0$

$(\mathrm{pk}, \mathrm{sk}) \twoheadleftarrow \mathsf{Sig.KeyGen}(1^\lambda)$
$(m^*, \sigma^*) \twoheadleftarrow \mathcal{A}_{\mathrm{ext}}^{\mathrm{Sign,pSign}}(\mathrm{pk})$
$(\sigma', Y^*, y^*) \leftarrow \sigma^*$
Assert $(\mathsf{Sig.Vrf}(\mathrm{pk}, (m^*, Y^*), \sigma') \ \wedge \ (Y^*, y^*) \in R)$
Return $(\forall Y \in T[m^*] : (Y, y^*) \notin R)$

pSign(m, Y):
$\sigma \twoheadleftarrow \mathsf{Sig.Sign}(\mathrm{sk}, (m, Y))$; $T[m] \xleftarrow{\cup} Y$
Return (σ, Y)

Adversary $\mathcal{A}_{\text{uf-cma}}^{\mathrm{Sign}}(\mathrm{pk})$:
$(m^*, \sigma^*) \twoheadleftarrow \mathcal{A}_{\mathrm{ext}}^{\mathrm{pSign}}(\mathrm{pk})$
$(\sigma', Y^*, y^*) \leftarrow \sigma^*$
Return $((m^*, Y^*), \sigma')$

pSign(m, Y):
$\sigma \twoheadleftarrow \mathrm{Sign}((m, Y))$
Return (σ, Y)

We claim that

$$\mathbf{Adv}^{\mathrm{ext}}_{\mathsf{Sig},\mathcal{A}_{\mathrm{ext}}}(\lambda) = \Pr[\mathbf{G}_0] \leq \mathbf{Adv}^{\text{uf-cma}}_{\mathsf{Sig},\mathcal{A}_{\text{uf-cma}}}(\lambda) \ . \tag{29}$$

This is straightforward, because if $\mathbf{G}_0$ returns true, then it must be that Y^* returned by the adversary is fresh, meaning it has not queried $\mathsf{pSign}(m^*, Y^*)$ previously. Finally, we note that adversary $\mathcal{A}_{\text{uf-cma}}$ also wins exactly when (m^*, Y^*) is fresh.

Next, we bound (28). Consider the following games $\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3$. Game $\mathbf{G}_1$ is $\mathbf{G}^{\mathrm{uext}}_{\mathsf{GAS}_1,\mathcal{A}_{\mathrm{uext}}}$. Games $\mathbf{G}_2$ and $\mathbf{G}_3$ rewrites the winning condition of $\mathbf{G}_1$ depending on disjoint events b_1 and b_2.

Game $\mathbf{G}_1$, $\mathbf{G}_2$, $\mathbf{G}_3$

$(\mathrm{pk}, \mathrm{sk}) \twoheadleftarrow \mathsf{Sig.KeyGen}(1^\lambda)$
$(m, Y, \hat{\sigma}, \sigma, \sigma') \twoheadleftarrow \mathcal{A}_{\mathrm{uext}}^{\mathrm{pSign}}(\mathrm{pk})$
$(\sigma_0, Y, y) \leftarrow \sigma$; $(\sigma_1, Y', y') \leftarrow \sigma'$
$b_1 \leftarrow (Y \neq Y')$
$b_2 \leftarrow (Y = Y') \wedge (y \neq y')$
$b_3 \leftarrow \mathsf{Vrf}(\mathrm{pk}, (m, Y), \sigma_0) \wedge \mathsf{Vrf}(\mathrm{pk}, (m, Y'), \sigma_1) \wedge (Y, y) \in \mathsf{R} \wedge (Y, y') \in \mathsf{R}$
$\mathbf{G}_1$: Return $(b_1 \vee b_2) \wedge b_3$
$\mathbf{G}_2$: Return $b_1 \wedge b_3$
$\mathbf{G}_3$: Return $b_2 \wedge b_3$

pSign(m, Y):
$\sigma \twoheadleftarrow \mathsf{Sig.Sign}(\mathrm{sk}, (m, Y))$; Return (σ, Y)

Clearly, we have

$$\mathbf{Adv}^{\mathrm{uext}}_{\mathsf{GAS}_1,\mathcal{A}_{\mathrm{uext}}}(\lambda) = \Pr[\mathbf{G}_1] = \Pr[\mathbf{G}_2] + \Pr[\mathbf{G}_3] \ . \tag{30}$$

Next, we construct adversaries $\mathcal{A}_{\text{suf-cma}}$ and $\mathcal{A}_{\mathrm{uwit}}$, such that

$$\Pr[\mathbf{G}_2] \leq \mathbf{Adv}^{\text{suf-cma}}_{\mathsf{Sig},\mathcal{A}_{\text{suf-cma}}}(\lambda) \ , \tag{31}$$

$$\Pr[\mathbf{G}_3] \leq \mathbf{Adv}^{\mathrm{uwit}}_{\mathsf{R},\mathcal{A}_{\mathrm{uwit}}}(\lambda) \ . \tag{32}$$

This is straightforward, $\mathcal{A}_{\text{suf-cma}}$ can simulate pSign with its Sign oracle, and $\mathcal{A}_{\mathrm{uwit}}$ can sample its own key pair to simulate game $\mathbf{G}_3$. The specifications of these adversaries are given below.

Adversary $\mathcal{A}^{\text{Sign}}_{\text{suf-cma}}(\text{pk})$:	Adversary $\mathcal{A}_{\text{uwit}}()$:
1 $(m, Y, \hat{\sigma}, \sigma, \sigma') \twoheadleftarrow \mathcal{A}^{\text{pSign}}_{\text{uext}}(\text{pk})$	1 $(\text{pk}, \text{sk}) \twoheadleftarrow \mathsf{Sig.KeyGen}(1^\lambda)$
2 $(\sigma_0, Y_0, y_0) \leftarrow \sigma$; $(\sigma_1, Y_1, y_1) \leftarrow \sigma'$	2 $(m, Y, \hat{\sigma}, \sigma, \sigma') \twoheadleftarrow \mathcal{A}^{\text{pSign}}_{\text{uext}}(\text{pk})$
3 If $\exists i \in \{1,2\} : (m, Y_i, \sigma_i) \notin U$ then	3 $(\sigma_0, Y_0, y_0) \leftarrow \sigma$; $(\sigma_1, Y_1, y_1) \leftarrow \sigma'$
4 Return $((m, Y_i), \sigma_i)$	4 Return (Y_0, y_0, y_1)
$\text{pSign}(m, Y)$:	$\text{pSign}(m, Y)$:
5 $\sigma \twoheadleftarrow \text{Sign}((m, Y))$	5 $\sigma \twoheadleftarrow \mathsf{Sig.Sign}(\text{sk}, (m, Y))$
6 $U \xleftarrow{\cup} (m, Y, \sigma)$	6 Return (σ, Y)
7 Return (σ, Y)	

This concludes the proof of Theorem 4.1.▮

We observe the following theorem.

Theorem 4.2 *If one-way functions exist then SFExt-secure adaptor signatures exist.*

The proof follows from the fact that one-way functions imply SUF-CMA-secure signatures [16, 21, 11] as well as (e.g. length doubling) pseudo-random generators (PRG) [12], which in turn imply hard relations with computationally unique witnesses consisting of pairs $(\mathsf{PRG}(x), x)$.

Our second construction adds *unlinkability* to GAS_1. To achieve this, we additionally need to assume that relation R is random-self-reducible. For example, hash-preimage relation is *not* RSR while discrete-logarithm relation is.

Generic Adaptor Signature (GAS) 2. First, we assume that R is random-self-reducible. The idea for adding unlinkability is simple: in pSign, we first derive a random instance Y' from the input instance Y and then only use Y' in $\mathsf{Sig.Sign}$; furthermore, we need to return the randomness r for the derivation of Y' as part of the pre-signature. To keep the scheme well specified, other parts of the scheme are modified accordingly. Formally, consider construction GAS_2 given in Figure 7. We will show that in addition to all the properties of GAS_1, scheme GAS_2 also achieves perfect unlinkability.

We will however need a slightly stronger form of RSR called strong RSR, which is captured via the following security game.

Game $\mathbf{G}^{\text{srsr}}_{\mathsf{R},\mathcal{A}}(\lambda)$:

1 $b \twoheadleftarrow \{0,1\}$; $b' \twoheadleftarrow \mathcal{A}^{\text{New}}()$

$\text{New}(Y, y)$:

2 Assert $(Y, y \in \mathsf{R})$; $(Y', y') \twoheadleftarrow \mathsf{R.Gen}(1^\lambda)$; $r \twoheadleftarrow \mathsf{R}.R_\lambda$

3 $Y_0 \leftarrow \mathsf{R}.A(Y', r)$; $y_0 \leftarrow \mathsf{R}.B(y', r)$

4 $Y_1 \leftarrow \mathsf{R}.A(Y, r)$; $y_1 \leftarrow \mathsf{R}.B(y, r)$

5 Return (Y_b, y_b)

We say that R is strongly random self-reducible (SRSR) if the advantage of any PT adversary $\mathcal{A}$, defined to be $\mathbf{Adv}^{\text{srsr}}_{\mathsf{R},\mathcal{A}}(\lambda) := 2\Pr[\mathbf{G}^{\text{srsr}}_{\mathsf{R},\mathcal{A}}(\lambda)] - 1$, is negligible.

Theorem 4.3 *Adaptor signature scheme GAS_2 satisfies correctness and pre-signature adaptability. If R is strongly random-self reducible, then GAS_2 is unlinkable. Specifically, given any adversary $\mathcal{A}_{\text{unlink}}$, the proof gives an adversary $\mathcal{A}_{\text{srsr}}$, as efficient as $\mathcal{A}_{\text{unlink}}$, such that*

$$\mathbf{Adv}^{\text{unlink}}_{\mathsf{GAS}_2,\mathcal{A}_{\text{unlink}}}(\lambda) \leq \mathbf{Adv}^{\text{srsr}}_{\mathsf{R},\mathcal{A}_{\text{srsr}}}(\lambda)\ . \tag{33}$$

Furthermore, if Sig is UF-CMA-secure and R is one-way then GAS_2 is FExt-secure. Given adversary $\mathcal{A}_{\text{fext}}$, we can construct adversaries $\mathcal{A}_{\text{uf-cma}}$ and $\mathcal{A}_{\text{ow}}$, with running times similar to that of

$\mathcal{A}_{\text{uf}}$, *such that*

$$\mathbf{Adv}^{\text{fext}}_{\mathsf{GAS}_2,\mathcal{A}_{\text{fext}}}(\lambda) \le \mathbf{Adv}^{\text{uf-cma}}_{\mathsf{Sig},\mathcal{A}_{\text{uf-cma}}}(\lambda) + \mathbf{Adv}^{\text{q-ow}}_{R,\mathcal{A}_{\text{ow}}}(\lambda)\ . \tag{34}$$

Lastly, if Sig *is SUF-CMA-secure and* R *has unique witnesses, then* GAS_1 *is SFExt-secure. Formally, given adversary* $\mathcal{A}_{\text{sfext}}$*, we can construct adversaries* $\mathcal{A}_{\text{suf-cma}}$*,* $\mathcal{A}_{\text{ow}}$*, and* $\mathcal{A}_{\text{uwit}}$ *with running times similar to that of* $\mathcal{A}_{\text{sfext}}$ *such that*

$$\mathbf{Adv}^{\text{sfext}}_{\mathsf{GAS}_1,\mathcal{A}_{\text{sfext}}}(\lambda) \le 2\mathbf{Adv}^{\text{suf-cma}}_{\mathsf{Sig},\mathcal{A}_{\text{uf-cma}}}(\lambda) + \mathbf{Adv}^{\text{q-ow}}_{R,\mathcal{A}_{\text{ow}}}(\lambda) + \mathbf{Adv}^{\text{uwit}}_{\mathsf{R},\mathcal{A}_{\text{uwit}}}\ . \tag{35}$$

Unlinkability follows from SRSR property in a straightforward manner and the rest of the proofs are very similar to those for Theorem 4.1. Note that any hard relation trivially implies one-way functions: for example, the mapping $f_\lambda(r) := Y$, where $(Y, y) \leftarrow \mathsf{R.Gen}(1^\lambda; r)$ and r is any element of the randomness space of $\mathsf{R.Gen}(1^\lambda)$, is one-way. Hence, similar to Theorem 4.2, we observe the following theorem.

Proof of Theorem 4.3: First, correctness and adaptability holds similar to GAS_1. We give a reduction that turns any unlink adversary to a strong RSR adversary for R. The reduction is very straightforward and we keep the descript at a high-level here. The SRSR adversary sample a key pair $(\text{pk}, \text{sk}) \twoheadleftarrow \mathsf{Sig.KeyGen}(1^\lambda)$, using which it can run pSign and Sign algorithms. It can simulate oracles Sign and pSign honestly. It uses the New oracle given to it from the strong RSR game to simulate SignChl, the pair (Y, y) that is in the input of SignChl is simply forwarded to New.

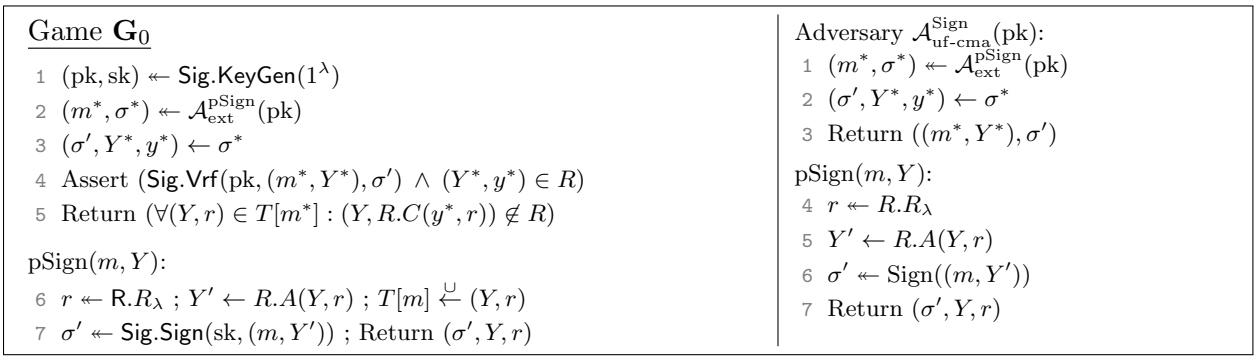
We check extractability. Similar to GAS_1, we need to show that for any adversary $\mathcal{A}_{\text{ext}}$ and $\mathcal{A}_{\text{uext}}$,

$$\mathbf{Adv}^{\text{ext}}_{\mathsf{GAS}_2,\mathcal{A}_{\text{ext}}}(\lambda) \le \mathbf{Adv}^{\text{uf-cma}}_{\mathsf{Sig},\mathcal{A}_{\text{uf-cma}}}(\lambda)\ , \tag{36}$$

$$\mathbf{Adv}^{\text{uext}}_{\mathsf{GAS}_2,\mathcal{A}_{\text{uext}}}(\lambda) \le \mathbf{Adv}^{\text{suf-cma}}_{\mathsf{Sig},\mathcal{A}_{\text{suf-cma}}}(\lambda) + \mathbf{Adv}^{\text{uwit}}_{\mathsf{R},\mathcal{A}_{\text{uwit}}}(\lambda)\ , \tag{37}$$

where $\mathcal{A}_{\text{uf-cma}}, \mathcal{A}_{\text{suf-cma}}, \mathcal{A}_{\text{uwit}}$ are reduction adversaries to be constructed.

We first show (36). Consider the following game $\mathbf{G}_0$ and adversary $\mathcal{A}_{\text{uf-cma}}$.



We claim that

$$\mathbf{Adv}^{\text{ext}}_{\mathsf{Sig},\mathcal{A}_{\text{ext}}}(\lambda) = \Pr[\mathbf{G}_0] \le \mathbf{Adv}^{\text{uf-cma}}_{\mathsf{Sig},\mathcal{A}_{\text{uf-cma}}}(\lambda)\ . \tag{38}$$

We claim that if $\mathbf{G}_0$ returns true, it must be that Y^* returned by adversary is fresh, meaning the adversary has not queried $\mathsf{Sign}((m^*, Y^*))$ previously. Seeking a contradiction, suppose that adversary has incurred a query $\mathsf{Sign}(m^*, Y^*)$, then this query must have come from some query $\mathsf{pSign}(m^*, Y_0)$, where the game has sampled some r_0 such that $\mathsf{R}.A(Y_0, r_0) = Y^*$. By line 4, $(Y^*, y^*) \in R$. So, $\mathsf{R}.C(y^*, r)$ must be a witness of Y_0. This means that the game must return False at line 5. Therefore, there was no signature on message (m^*, Y^*) if the game returns True. We note that adversary $\mathcal{A}_{\text{uf-cma}}$ also wins exactly when (m^*, Y^*) is fresh. This verifies (36).

Next, we bound (37). Consider the following games $\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3$. Game $\mathbf{G}_1$ is $\mathbf{G}^{\text{uext}}_{\mathsf{GAS}_2,\mathcal{A}_{\text{uext}}}$. Games $\mathbf{G}_2$ and $\mathbf{G}_3$ rewrites the winning condition of $\mathbf{G}_1$ depending on disjoint events b_1 and b_2.

Game $\mathbf{G}_1$, $\mathbf{G}_2$, $\mathbf{G}_3$

$(\text{pk}, \text{sk}) \leftarrow\!\!\!\!\$\ \mathsf{Sig.KeyGen}(1^\lambda)$
$(m, Y, \hat{\sigma}, \sigma, \sigma') \leftarrow\!\!\!\!\$\ \mathcal{A}^{\text{pSign}}_{\text{uext}}(\text{pk})$
$(\sigma_0, Y, y) \leftarrow \sigma$; $(\sigma_1, Y', y') \leftarrow \sigma'$
$b_1 \leftarrow (Y \neq Y')$
$b_2 \leftarrow (Y = Y') \wedge (y \neq y')$
$b_3 \leftarrow \mathsf{Vrf}(\text{pk}, (m, Y), \sigma_0) \wedge \mathsf{Vrf}(\text{pk}, (m, Y'), \sigma_1) \wedge (Y, y) \in \mathsf{R} \wedge (Y, y') \in \mathsf{R}$
$\mathbf{G}_1$: Return $(b_1 \vee b_2) \wedge b_3$
$\mathbf{G}_2$: Return $b_1 \wedge b_3$
$\mathbf{G}_3$: Return $b_2 \wedge b_3$

pSign(m, Y):
$r \leftarrow\!\!\!\!\$\ \mathsf{R}.R_\lambda$; $Y' \leftarrow \mathsf{R}.A(Y, r)$; $\sigma' \leftarrow\!\!\!\!\$\ \mathsf{Sign}(\text{sk}, (m, Y'))$; Return (σ', Y, r)

Clearly, we have

$$\mathbf{Adv}^{\text{uext}}_{\mathsf{GAS}_1,\mathcal{A}_{\text{uext}}}(\lambda) = \Pr[\mathbf{G}_1] = \Pr[\mathbf{G}_2] + \Pr[\mathbf{G}_3] \ . \tag{39}$$

Next, we construct adversaries $\mathcal{A}_{\text{suf-cma}}$ and $\mathcal{A}_{\text{uwit}}$, such that

$$\Pr[\mathbf{G}_2] \leq \mathbf{Adv}^{\text{suf-cma}}_{\mathsf{Sig},\mathcal{A}_{\text{suf-cma}}}(\lambda) \ , \tag{40}$$

$$\Pr[\mathbf{G}_3] \leq \mathbf{Adv}^{\text{uwit}}_{\mathsf{R},\mathcal{A}_{\text{uwit}}}(\lambda) \ . \tag{41}$$

This is straightforward, $\mathcal{A}_{\text{suf-cma}}$ can simulate pSign with its Sign oracle, and $\mathcal{A}_{\text{uwit}}$ can sample its own key pair to simulate game $\mathbf{G}_3$. The specifications of these adversaries are given below.

Adversary $\mathcal{A}^{\text{Sign}}_{\text{suf-cma}}(\text{pk})$:
$(m, Y, \hat{\sigma}, \sigma, \sigma') \leftarrow\!\!\!\!\$\ \mathcal{A}^{\text{pSign}}_{\text{uext}}(\text{pk})$
$(\sigma_0, Y_0, y_0) \leftarrow \sigma$; $(\sigma_1, Y_1, y_1) \leftarrow \sigma'$
If $\exists i \in \{1, 2\} : (m, Y_i, \sigma_i) \notin U$ then
  Return $((m, Y_i), \sigma_i)$
pSign(m, Y):
$r \leftarrow\!\!\!\!\$\ \mathsf{R}.R_\lambda$; $Y' \leftarrow \mathsf{R}.A(Y, r)$
$\sigma' \leftarrow\!\!\!\!\$\ \text{Sign}((m, Y'))$
$U \xleftarrow{\cup} (m, Y', \sigma')$; Return (σ', Y, r)

Adversary $\mathcal{A}_{\text{uwit}}()$:
$(\text{pk}, \text{sk}) \leftarrow\!\!\!\!\$\ \mathsf{Sig.KeyGen}(1^\lambda)$
$(m, Y, \hat{\sigma}, \sigma, \sigma') \leftarrow\!\!\!\!\$\ \mathcal{A}^{\text{pSign}}_{\text{uext}}(\text{pk})$
$(\sigma_0, Y_0, y_0) \leftarrow \sigma$; $(\sigma_1, Y_1, y_1) \leftarrow \sigma'$
Return (Y_0, y_0, y_1)
pSign(m, Y):
$r \leftarrow\!\!\!\!\$\ \mathsf{R}.R_\lambda$; $Y' \leftarrow \mathsf{R}.A(Y, r)$
$\sigma' \leftarrow\!\!\!\!\$\ \mathsf{Sign}(\text{sk}, (m, Y'))$; Return (σ', Y, r)

This concludes the proof of Theorem 4.3.∎

Theorem 4.4 *If SRSR relations exist then SFExt-secure and unlinkable adaptor signatures exist.*

Strong RSR relations from any epimorphic (homomorphic and onto) one-way function. Suppose $f_\lambda : D_\lambda \to R_\lambda$ is a homomorphic one-way function, where D_λ and R_λ are both abelian groups (where D_λ has group operation $+$ and R_λ has group operation $\cdot$). For example, two instantiations are $f_{\mathbb{G},g}(y) = g^y$ and $f_{(N,e)}(y) = y^e \mod N$, where $(\mathbb{G}, g)$ is a group instance and (N, e) is an RSA public key. Then it is clear that the relation containing pairs $(f(y), y)$ can be made strongly RSR. In particular, we consider $\mathsf{R}.A(Y, r) := Y \cdot f(r)$, where r is sampled uniformly randomly from D_λ. The corresponding algorithms B and C are defined as $\mathsf{R}.B(y, r) = y + r$ and $\mathsf{R}.C(y', r) = y' - r$. It is easy to check that the relation is SRSR since $(f(y + r), y + r)$ is uniformly random regardless of the value of y, as long as f is homomorphic and onto.

Strong RSR relations from LWE. We sketch how LWE gives rise to a SRSR relation. Recall that for LWE, the dimension n is the LWE security parameter. Take any m that is polynomial in n. We fix[3] a random matrix $A_\lambda \in \mathbb{Z}_q^{m\times n}$ for each security parameter λ. (We can take $n = \mathcal{O}(\lambda)$.) Let q be the modulus and α_e and α_t be parameters that we shall fix at the end. Consider the following relation R consisting of pairs (Y, y) with $y \leftarrow\!\!\!\!\!\!\leftarrow \mathbb{Z}_q^n$ and $Y = Ay + e$, where each component of e is sampled from a discrete Gaussian of width $\alpha_e q$. We define the rerandomize algorithm $\mathsf{R}.A(Y, (r, t)) := Y + A_\lambda r + t$, where r is uniformly random in $\mathbb{Z}_q^n$ and $t \in \mathbb{Z}_q^m$ is such that each component of t is sampled from a discrete Gaussian of width $\alpha_t q$. Lastly, we define $\mathsf{R}.B(y, (r, t)) := y + r$ and $\mathsf{R}.B(y, (r, t)) := y - r$. Above, components of e and t are from (discretized) Gaussian distributions of parameter $\alpha_e q$ and $\alpha_t q$, respectively, where $\alpha_e = \mathcal{O}(1/f^2(n))$, $\alpha_t = \mathcal{O}(1/f(n))$, and $q = \mathcal{O}(f^3(n))$ for a super-polynomial function $f(n)$. For example, the parameter of distribution e is $\mathcal{O}(n^{\log n})$, that of t is $\mathcal{O}(n^{2\log n})$, and modulus q is $\mathcal{O}(n^{3\log n})$. Note that given some value of error e, one cannot distinguish between $e + t$ and $e' + t$ for freshly sampled e' (i.i.d to e) and t. This is because t is "wider" than e by a factor of $n^{\log(n)}$, which is super-polynomial. Finally, by [17, Theorem 4.2.4], the relation R is SRSR if GapSVP_γ and SIVP_γ are hard against quantum adversaries, where $\gamma = \widetilde{\mathcal{O}}(n^{1-\log(n)})$.

References

[1] L. Aumayr, O. Ersoy, A. Erwig, S. Faust, K. Hostáková, M. Maffei, P. Moreno-Sanchez, and S. Riahi. Bitcoin-compatible virtual channels. Cryptology ePrint Archive, Report 2020/554, 2020. `https://eprint.iacr.org/2020/554`. 3, 4, 7, 10

[2] L. Aumayr, O. Ersoy, A. Erwig, S. Faust, K. Hostakova, M. Maffei, P. Moreno-Sanchez, and S. Riahi. Generalized bitcoin-compatible channels. Cryptology ePrint Archive, Report 2020/476, 2020. `https://eprint.iacr.org/2020/476`. 3, 4, 6, 7, 8, 9, 10, 11, 12

[3] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006. 5

[4] W. Beullens, T. Kleinjung, and F. Vercauteren. CSI-FiSh: Efficient isogeny based signatures through class group computations. In S. D. Galbraith and S. Moriai, editors, *ASIACRYPT 2019, Part I*, volume 11921 of *LNCS*, pages 227–247. Springer, Heidelberg, Dec. 2019. 4

[5] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In C. Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Heidelberg, Dec. 2001. 3

[6] L. Ducas, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehle. CRYSTALS – Dilithium: Digital signatures from module lattices. Cryptology ePrint Archive, Report 2017/633, 2017. `https://eprint.iacr.org/2017/633`. 4

[7] A. Erwig, S. Faust, K. Hostáková, M. Maitra, and S. Riahi. Two-party adaptor signatures from identification schemes. In J. Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 451–480. Springer, Heidelberg, May 2021. 3, 4, 6, 10

[3] More formally, A should be sampled as a parameter for each security parameter, but we fix such A here for simplicity.

[8] M. F. Esgin, O. Ersoy, and Z. Erkin. Post-quantum adaptor signatures and payment channel networks. Cryptology ePrint Archive, Report 2020/845, 2020. https://eprint.iacr.org/2020/845. 3, 4, 6, 10

[9] L. Fournier. One-time verifiably encrypted signatures aka adaptor signatures, 2019. 6

[10] J. Gugger. Bitcoin-monero cross-chain atomic swap. Cryptology ePrint Archive, Report 2020/1126, 2020. https://eprint.iacr.org/2020/1126. 3

[11] Q. Huang, D. S. Wong, and Y. Zhao. Generic transformation to strongly unforgeable signatures. In J. Katz and M. Yung, editors, *ACNS 07*, volume 4521 of *LNCS*, pages 1–17. Springer, Heidelberg, June 2007. 24

[12] R. Impagliazzo, L. A. Levin, and M. Luby. Pseudo-random generation from one-way functions (extended abstracts). In *21st ACM STOC*, pages 12–24. ACM Press, May 1989. 24

[13] D. Johnson, A. Menezes, and S. Vanstone. The elliptic curve digital signature algorithm (ECDSA). *International journal of information security*, 1(1):36–63, 2001. 3

[14] G. Malavolta, P. Moreno-Sanchez, C. Schneidewind, A. Kate, and M. Maffei. Anonymous multi-hop locks for blockchain scalability and interoperability. In *NDSS 2019*. The Internet Society, Feb. 2019. 3, 6

[15] D. Moody, G. Alagic, D. C. Apon, D. A. Cooper, Q. H. Dang, J. M. Kelsey, Y.-K. Liu, C. A. Miller, R. C. Peralta, R. A. Perlner, et al. Status report on the second round of the nist post-quantum cryptography standardization process. 2020. 4, 5

[16] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *21st ACM STOC*, pages 33–43. ACM Press, May 1989. 24

[17] C. Peikert. How (not) to instantiate ring-LWE. Cryptology ePrint Archive, Report 2016/351, 2016. https://eprint.iacr.org/2016/351. 27

[18] A. Poelstra. Lightning in scriptless scripts. https://lists.launchpad.net/mimblewimble/msg00086.html, 2017. Accessed: Aug, 2021. 3, 6

[19] A. Poelstra. Scriptless scripts. https://download.wpsoftware.net/bitcoin/wizardry/mw-slides/2017-03-mit-bitcoin-expo/slides.pdf, 2017. Accessed: Aug, 2021. 3, 6

[20] J. Poon and T. Dryja. The bitcoin lightning network: Scalable off-chain instant payments. https://lightning.network/lightning-network-paper.pdf, 2016. Accessed: Aug, 2021. 3, 21

[21] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd ACM STOC*, pages 387–394. ACM Press, May 1990. 24

[22] C.-P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, Jan. 1991. 3

[23] E. Tairi, P. Moreno-Sanchez, and M. Maffei. Post-quantum adaptor signature for privacy-preserving off-chain payments. Cryptology ePrint Archive, Report 2020/1345, 2020. https://eprint.iacr.org/2020/1345. 3, 4, 6, 10