A preliminary version of this paper [8] (https://doi.org/10.1007/978-3-031-17146-8_24) appeared in the proceedings of the *27th European Symposium on Research in Computer Security (ESORICS) 2022*. This is the full version.*

Deterministic Wallets for Adaptor Signatures

Andreas Erwig and Siavash Riahi

Technische Universität Darmstadt, Darmstadt, Germany
{andreas.erwig, siavash.riahi}@tu-darmstadt.de

Abstract. *Adaptor signatures* are a new cryptographic primitive that binds the authentication of a message to the revelation of a secret value. In recent years, this primitive has gained increasing popularity both in academia and practice due to its versatile use-cases in different Blockchain applications such as atomic swaps and payment channels. The security of these applications, however, crucially relies on users storing and maintaining the secret values used by adaptor signatures in a secure way. For standard digital signature schemes, cryptographic wallets have been introduced to guarantee secure storage of keys and execution of the signing procedure. However, no prior work has considered cryptographic wallets for adaptor signatures.

In this work, we introduce the notion of *adaptor wallets*. Adaptor wallets allow parties to securely use and maintain adaptor signatures in the Blockchain setting. Our adaptor wallets are both deterministic and operate in the hot/cold paradigm, which was first formalized by Das et al. (CCS 2019) for standard signature schemes. We introduce a new cryptographic primitive called *adaptor signatures with rerandomizable keys*, and use it to generically construct adaptor wallets. We further show how to instantiate adaptor signatures with rerandomizable keys from the ECDSA signature scheme and discuss that they can likely be built for Schnorr and Katz-Wang schemes as well. Finally, we discuss the limitations of the existing ECDSA- and Schnorr-based adaptor signatures w.r.t. deterministic wallets in the hot/cold setting and prove that it is impossible to overcome these drawbacks given the current state-of-the-art design of adaptor signatures.

1 Introduction

Blockchains have gained huge popularity in the past decade as they provide a decentralized infrastructure that allows not only to make simple payments but also to execute applications in a secure way. However, most Blockchains, including Bitcoin, only support the execution of simple applications while others, such as Monero or Zcash, are even more restrictive in their functionality and only support simple payments [21, 26]. Nevertheless, virtually all Blockchains rely

*In particular, this version expands the preliminary version by including the formal model of adaptor wallets as well as a security analysis of our generic adaptor wallet construction and the impossibility result of independent statement/witness derivation. The content of the preliminary version is excluded from the CC BY license.

on digital signatures in order to authenticate the origin of a transaction. While the functionality of Blockchains can be extended by appropriately adjusting the mining algorithms, this requires a hard fork of the Blockchain code which can take several years to complete in practice. In order to improve the restricted functionality of many Blockchains without having to change the Blockchain implementation and to allow for the execution of a larger class of applications, a new type of signature scheme called *adaptor signatures* was introduced by the cryptocurrency community [20] and first formally analyzed by Aumayr et al. [3]. At a high level, adaptor signatures allow two parties, say a *signer* and a *publisher* to trade a signature in exchange for a secret, i.e., if the publisher publishes a signature under the signer's secret key on the Blockchain, a secret value is leaked to the signer. More concretely, the publisher first generates an instance of a hard relation, i.e., a statement and witness pair and sends the statement to the signer. Using its secret key and the statement, the signer generates an incomplete signature called *pre-signature* which can be *adapted* by the publisher to a full valid signature using the witness. Once the adapted full signature is published, the signer can *extract* the witness given the pre- and full signature.

Adaptor signatures have proven to be extremely versatile for Blockchain applications. They allow for efficient constructions of two important categories of applications, namely payment channels (e.g., [3, 23]) and atomic swaps (e.g., [7, 25]), while requiring only a minimal functionality from the underlying Blockchain. Payment channels are a so-called off-chain solution, which allows two parties to issue many micropayments to each other without incurring fees for each transaction. Atomic swaps, on the other hand, allow two (or more) parties to atomically exchange tokens, i.e., either the exchange terminates and both parties obtain the other party's token or none does. Both of these applications rely on a technique that allows exchanging a secret value for a signature, which is exactly the functionality that adaptor signatures provide.

As the security of a user's funds in a Blockchain network depends solely on the secure storage of this user's signing secret key (and witnesses of adaptor signatures), it is of utmost importance how users store these secret values. Unfortunately, despite the increasing popularity of adaptor signatures, no prior work tried to address this issue. In other words we would like to answer the following question:

How can parties in practice employ adaptor signatures securely?

A concept known as *cryptographic wallets* has been introduced to use standard signature schemes securely in Blockchain networks. However, it has never been investigated if this concept can be extended to adaptor signatures.

Deterministic Wallets. One of the most promising proposals for cryptographic wallets are so-called deterministic wallets, which at a high level store a master signing key pair from which session key pairs are deterministically derived. Das et al. [5] gave the first formalization of such deterministic wallets in the hot/cold setting and later extended their model [6] to incorporate hierarchical wallets. In a bit more detail, a wallet scheme in the hot/cold setting consists of two separate devices, a hot and a cold wallet, that store the public and secret key respectively. The cold wallet is kept mostly offline and is only used to generate a new signature, whereas the hot wallet is constantly online to receive new transactions. This wallet structure ensures that it is inherently difficult for an attacker to steal the wallet's secret key, as it is stored in the offline cold wallet. Besides a standard unforgeability notion, wallet schemes should typically also satisfy an *unlinkability* property, which ensures that a third party cannot link two transactions issued to the same wallet. A naïve approach to achieve unlinkability is to let the wallet generate a fresh key pair for each transaction. This, however, requires the wallet to store all key pairs, which is not efficient, especially since cold wallets sometimes require special hardware (with limited storage) to securely store the secret keys. As such, deterministic wallets were introduced where the unlinkable keys are deterministically derived from a master key pair. This allows the wallet to derive new keys on the fly when they are needed instead of storing them indefinitely.

To date deterministic wallets have only been analyzed for digital signature schemes (e.g., [5]). Considering that the security of adaptor signatures does not only depend on the secure storage of the secret key but also on the secure handling of witnesses, designing a secure wallet scheme for adaptor signatures becomes even more pressing.

1.1 Our Contribution

In this work, we initiate the study of deterministic wallets in the hot/cold setting for adaptor signatures following the approach of Das et al. [5]. To this end, we first introduce a new notion of adaptor signatures, which we call *adaptor signature with rerandomizable keys*. This primitive extends regular adaptor signatures by key rerandomization algorithms. That is, given an adaptor signature key pair (sk, pk) and some randomness ρ, an adaptor signature with rerandomizable keys allows to deterministically and independently rerandomize sk and pk using ρ to obtain a new key pair (sk', pk') such that (1) (sk', pk') constitutes a valid signing key pair, and (2) (sk', pk') is indistinguishable from a freshly generated key pair. We formally define this primitive and show how to instantiate it by transforming the existing ECDSA-adaptor signature scheme [3, 19] into an adaptor signature with rerandomizable keys.

We provide a formal model for adaptor wallets. Our adaptor wallets are the first cryptographic wallets that are deterministic, in the hot/cold setting and support the use of adaptor signatures. While the hot/cold wallet setting allows to provide strong security guarantees, it is not suitable for all applications in practice. Payment channels, for instance, have a short life span but require a frequent exchange of signatures. As such, storing the secret key in an offline cold wallet seems counterintuitive. Instead, for such applications our model allows to store secret values on one online device while guaranteeing that even if this device gets corrupted, the master key pair and other keys derived from the master key pair remain secure. To achieve this feature, we use the idea of hardened/non-hardened wallets as defined in [6] and adjust it for adaptor wallets (see Section 4.1 for more details).

We then show how to generically construct adaptor wallets from *any* adaptor signature scheme with rerandomizable keys where the hard relation is *witness rerandomizable* and further show how to initiate such a relation for ECDSA-adaptor signatures. Witness rerandomizability of a hard relation R essentially means that for any statement/witness pair $(Y, y) \in R$ the witness y can be rerandomized deterministically using some randomness ρ to a witness y' with corresponding statement Y' such that $(Y', y') \in R$. We require this property to alleviate the storage constraints on the cold wallet, i.e., as explained above, the cold wallet is often a storage restricted device and hence deterministic rerandomization can be useful to generate required values on the fly instead of storing them long-term. Although we do not formally show how adaptor wallets can be instantiated from Schnorr and Katz-Wang signature schemes [22, 14], it seems that our approach can be used in order to transform these schemes to adaptor signatures with rerandomizable keys and use them to instantiate adaptor wallets.

Our final contribution is closely related to witness rerandomizable hard relations. Surprisingly, we show that it is *impossible* to construct an adaptor wallet from fully rerandomizable hard relations, i.e., hard relations where the statement and witness can be rerandomized independently using the same randomness. This is in stark contrast to the secret and public keys which can be rerandomized independently.

We believe that our work paves the way for mainstreaming the usage of adaptor signatures by providing a secure and efficient deterministic wallet framework in the hot/cold setting.

1.2 Related Work

We divide the related work into adaptor signatures and deterministic wallets.

Adaptor Signatures. After being first introduced by Poelstra [20], adaptor signatures have been used in many Blockchain related applications, such as atomic swaps [7], payment channel networks [18] and payment channel hubs [23]. Aumayr et al. [3] later provided a standalone formalization of this primitive. Shortly after, Esgin et al. and Tairi et al. [10, 24] provided instantiations of adaptor signatures in the post-quantum setting where the adversary has access to a quantum computer while the end users do not. Finally, Erwig et al. [9] showed how to generically transform signature schemes built from identification schemes which satisfy certain properties, into single party and two party adaptor signatures. There have been several other recent works on adaptor signatures (e.g., [17, 25]) which have used or extended this primitive in order to build more complex applications.

Deterministic Wallets. There have been many recent works formalizing and analyzing cryptographic wallets, such as [2, 13, 15, 16]. The concept of deterministic wallets in the hot/cold setting was first formalized and instantiated by Das et al. [5]. Alkadri et al. [1] later showed how to realize such wallets with security in the post-quantum setting. In a follow-up work, Das et al. [6] extended the original model by allowing hierarchical derivation of new wallets. In order to guarantee security even in case one of such wallets is corrupted, e.g., when a wallet is not implemented in the hot/cold setting, the authors introduced two different key derivation mechanisms, namely hardened key derivation for keys that might be leaked to the adversary and non-hardened key derivation for keys that are stored securely via the hot/cold wallet paradigm. Later, Yin et al. [27] introduced hierarchical deterministic wallets that support stealth addresses. However, none of these works have considered adaptor signature support for deterministic wallets.

2 Preliminaries

Notation. We denote by $s \leftarrow_\$ H$ the uniform random sampling of a value s from the set H. For an integer l, the notation $[l]$ denotes the set of integers $\{1, \cdots, l\}$ and for a randomized algorithm A, we denote by $y \leftarrow_\$ A(x)$ the execution of A on input x that outputs y. For a deterministic algorithm B, we write $y \leftarrow B(x, \rho)$ to denote the execution of B on input x and ρ that outputs y. By $y \in A(x)$ we denote that y is an element in the set of possible outputs of an execution of A on input x. Throughout our paper, we assume that public parameters par can be used as input to all algorithms. For two strings a and b, we write $a = (b, \cdot)$ if b is a prefix of a. We abbreviate the expressions *deterministic polynomial time* and *probabilistic polynomial time* by DPT and PPT respectively.

2.1 Non-interactive zero knowledge proofs.

A non-interactive zero knowledge proof (NIZK) [4] with respect to a polynomial-time recognizable binary relation R is given by the following tuple of algorithms $\mathsf{NIZK} := (\mathsf{Setup}_R, \mathsf{P}, \mathsf{V})$, where (i) $\mathsf{Setup}_R(1^n)$ outputs a common reference string crs; (ii) $\mathsf{P}(\mathsf{crs}, (Y, y))$ outputs a proof π for $(Y, y) \in R$; (iii) $\mathsf{V}(\mathsf{crs}, Y, \pi)$ outputs a bit $b \in \{0, 1\}$. Further, the NIZK proof of knowledge w.r.t. R should satisfy the properties *completeness*, *soundness*, and *zero knowledge*. We do not go into the details of these properties here.

2.2 (Witness rerandomizable) Hard relation.

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

In this work, we consider only hard relations, where there exists exactly one valid witness for any statement in language L_R. That is, we consider hard relations, where the corresponding language is defined as $L_R := \{Y \in \mathcal{D}_{\mathsf{Y}} \mid \exists! y \in \mathcal{D}_{\mathsf{w}} \text{ s.t. } (Y, y) \in R\}$. Additionally, in this work we require a stronger notion of hard relation namely hard relations that are witness rerandomizable.

Definition 2 (Witness Rerandomizable Hard Relation). *Let $R \subseteq \mathcal{D}_{\mathsf{Y}} \times \mathcal{D}_{\mathsf{w}}$ be a hard relation with statement/witness pairs $(Y, y) \in \mathcal{D}_{\mathsf{Y}} \times \mathcal{D}_{\mathsf{w}}$ and let the public parameters* par *define a randomness space $X := X(\mathsf{par})$. Further, let* $\mathsf{RandWit}$ *be a DPT algorithm which is defined as follows:*
$\mathsf{RandWit}(y, \rho)$*: The deterministic witness randomization algorithm takes as input a witness $y \in \mathcal{D}_{\mathsf{w}}$, a randomness $\rho \in X$ and outputs a rerandomized witness y'.*

We say that R is perfectly witness rerandomizable *if for all $(\cdot, y) \in \mathsf{GenR}(1^n)$ and all $\rho \leftarrow_{\$} X$ the distributions of (Y', y') and (Y'', y'') are identical, where:*

$$(Y', y') \leftarrow (\mathsf{WitToSt}(\mathsf{RandWit}(y, \rho)), \mathsf{RandWit}(y, \rho))$$
$$(Y'', y'') \leftarrow \mathsf{GenR}(1^n)$$

2.3 Adaptor Signatures

We recall the definition of an adaptor signature scheme by Aumayr et al. [3]. We then provide the correctness and security definitions of Adaptor signature.

Definition 3 (Adaptor signature scheme). *An adaptor signature scheme w.r.t. a hard relation R and a signature scheme $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ consists of four algorithms $\mathsf{ASig}_{R,\Sigma} = (\mathsf{pSign}, \mathsf{Adapt}, \mathsf{pVrfy}, \mathsf{Ext})$ with the following syntax:* $\mathsf{pSign}(sk, m, Y)$ *is a PPT algorithm that on input a secret key sk, message $m \in \{0,1\}^*$ and statement $Y \in L_R$, outputs a pre-signature $\tilde{\sigma}$;* $\mathsf{pVrfy}(pk, m, Y, \tilde{\sigma})$ *is a DPT algorithm that on input a public key pk, message $m \in \{0,1\}^*$, statement $Y \in L_R$ and pre-signature $\tilde{\sigma}$, outputs a bit b;* $\mathsf{Adapt}(\tilde{\sigma}, y)$ *is a DPT algorithm that on input a pre-signature $\tilde{\sigma}$ and witness y, outputs a signature σ; and* $\mathsf{Ext}(\sigma, \tilde{\sigma}, Y)$ *is a DPT algorithm that on input a signature σ, pre-signature $\tilde{\sigma}$ and statement $Y \in L_R$, outputs a witness y such that $(Y, y) \in R$, or $\perp$.*

An adaptor signature scheme $\mathsf{ASig}_{R,\Sigma}$ must satisfy *pre-signature correctness* stating that for every $m \in \{0,1\}^*$ and every $(Y, y) \in R$, the following holds:

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

An adaptor signature scheme has to satisfy the following security properties.

Definition 4 (Existential unforgeability). *An adaptor signature scheme $\mathsf{ASig}_{R,\Sigma}$ is unforgeable if for every PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ there exists a negligible function ν such that:* $\Pr[\mathsf{aSigForge}_{\mathcal{A},\mathsf{ASig}_{R,\Sigma}}(n) = 1] \leq \nu(n)$*, where the experiment* $\mathsf{aSigForge}_{\mathcal{A},\mathsf{ASig}_{R,\Sigma}}$ *is defined in Fig. 1.*

```
aWitExt$(n)$
00 $\mathcal{Q} := \emptyset, (sk, pk) \leftarrow \mathsf{Gen}(1^n)$
01 $(m^*, Y^*, \mathsf{st}) \leftarrow \mathcal{A}_1^{\mathtt{SignO}(\cdot),\mathtt{PreSignO}(\cdot,\cdot)}(pk)$
02 $\tilde{\sigma}^* \leftarrow \mathsf{pSign}(sk, m^*, Y^*)$
03 $\sigma^* \leftarrow \mathcal{A}_2^{\mathtt{SignO},\mathtt{PreSignO}}(\tilde{\sigma}^*, \mathsf{st})$
04 $b_1 \leftarrow (Y^*, \mathsf{Ext}(\sigma^*, \tilde{\sigma}^*, Y^*)) \notin R$
05 $b_2 \leftarrow m^* \notin \mathcal{Q}$
06 $b_3 \leftarrow \mathsf{Verify}(pk, m^*, \sigma^*)$
07 $b_4 \leftarrow Y^* \in L_R$
08 Return $(b_1 \land b_2 \land b_3 \land b_4)$

Oracle $\mathtt{PreSignO}(m, Y)$
09 $\tilde{\sigma} \leftarrow \mathsf{pSign}(sk, m, Y)$
10 $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
11 Return $\tilde{\sigma}$
```

```
aSigForge$(n)$
12 $\mathcal{Q} := \emptyset, (sk, pk) \leftarrow \mathsf{Gen}(1^n)$
13 $(Y, y) \leftarrow \mathsf{GenR}(1^n)$
14 $(m^*, \mathsf{st}) \leftarrow \mathcal{A}_1^{\mathtt{SignO},\mathtt{PreSignO}}(pk, Y)$
15 $\tilde{\sigma}^* \leftarrow \mathsf{pSign}(sk, m^*, Y)$
16 $\sigma^* \leftarrow \mathcal{A}_2^{\mathtt{SignO},\mathtt{PreSignO}}(\tilde{\sigma}^*, \mathsf{st})$
17 Return
$(m^* \notin \mathcal{Q} \land \mathsf{Verify}(pk, m^*, \sigma^*))$

Oracle $\mathtt{SignO}(m)$
18 $\sigma \leftarrow \mathsf{Sign}(sk, m)$
19 $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
20 Return $\sigma$
```

Fig. 1. $\mathsf{aSigForge}$ and $\mathsf{aWitExt}$ games for an adaptor signature scheme ASig.

Definition 5 (Pre-signature adaptability). *An adaptor signature scheme $\mathsf{ASig}_{R,\Sigma}$ satisfies pre-signature adaptability if for any message $m \in \{0,1\}^*$, any statement/witness pair $(Y, y) \in R$, any public key pk and any pre-signature $\tilde{\sigma} \in \{0,1\}^*$ with $\mathsf{pVrfy}(pk, m, Y, \tilde{\sigma}) = 1$, we have $\mathsf{Verify}(pk, m, \mathsf{Adapt}(\tilde{\sigma}, y)) = 1$.*

Definition 6 (Witness extractability). *An adaptor signature scheme $\mathsf{ASig}_{R,\Sigma}$ is* witness extractable *if for every PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function ν such that the following holds:* $\Pr[\mathsf{aWitExt}_{\mathcal{A},\mathsf{ASig}_{R,\Sigma}}(n) = 1] \leq \nu(n)$, *where the experiment $\mathsf{aWitExt}_{\mathcal{A},\mathsf{ASig}_{R,\Sigma}}$ is defined in Fig. 1.*

Definition 7. *An adaptor signature scheme $\mathsf{ASig}_{R,\Sigma}$ is secure, if it is unforgeable, pre-signature adaptable and witness extractable.*

2.4 ECDSA-based Adaptor Signature

We briefly recall the ECDSA-based adaptor signature scheme $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}] = (\mathsf{pSign}, \mathsf{Adapt}, \mathsf{pVrfy}, \mathsf{Ext})$ as presented by Aumayr et al. [3], which is defined w.r.t. the positive ECDSA signature scheme $\mathsf{PEC} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ and a hard relation R_g. Recall that the positive ECDSA scheme operates over a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order p and that the key generation outputs a key pair (sk, pk) with $sk \leftarrow_\$ \mathbb{Z}_p$ and $pk \leftarrow g^{sk}$. A message $m \in \{0,1\}^*$ is then signed by first sampling $k \leftarrow_\$ \mathbb{Z}_p$, setting $r \leftarrow f(g^k)$ and computing $s := k^{-1}(\mathsf{H}(m) + r \cdot sk)$, where $\mathsf{H} : \{0,1\} \to \mathbb{Z}_p$ is a hash function and $f : \mathbb{G} \to \mathbb{Z}_p$. The signature is then $\sigma := (r, s)$, which can be verified by checking if $f(g^{s^{-1}\mathsf{H}(m)} pk^{s^{-1}r}) = r$. The hard relation R_g is defined as $R_g := \{((Y, \pi), y) | Y = g^y \land \mathsf{V}(Y, \pi) = 1\}$, i.e., it is the standard dlog relation with an additional non-interactive zero knowledge (NIZK) proof, which proves knowledge of the witness. The additional NIZK proof is required for technical reasons which we do not discuss here. Apart from the

NIZK proof for relation R_g, the $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}]$ construction also includes a NIZK proof for another relation $R_Y := \{((\tilde{K}, K), k) | \tilde{K} = g^k \wedge K = Y^k\}$. For further details we refer the reader to [3]. The construction of $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}]$ is depicted in Fig. 2.

```
$\mathsf{pSign}(sk, m, I_Y)$
    $x := sk, (Y, \pi_Y) := I_Y$
    $k \leftarrow_\$ \mathbb{Z}_q, \tilde{K} := g^k$
    $K := Y^k, r := f(K)$
    $\tilde{s} := k^{-1}(\mathsf{H}(m) + rx)$
    $\pi \leftarrow \mathsf{P}_Y((\tilde{K}, K), k)$
    return $(r, \tilde{s}, K, \pi)$
```

```
$\mathsf{pVrfy}(pk, m, I_Y, \tilde{\sigma})$
    $X := pk, (Y, \pi_Y) := I_Y$
    $(r, \tilde{s}, K, \pi) := \tilde{\sigma}$
    $u := \mathsf{H}(m) \cdot \tilde{s}^{-1}$
    $v := r \cdot \tilde{s}^{-1}, K' := g^u X^v$
    return $(I_Y \in L_R$
        $\wedge (r = f(K)) \wedge \mathsf{V}_Y((K', K), \pi))$
```

```
$\mathsf{Adapt}(\tilde{\sigma}, y)$
    $(r, \tilde{s}, K, \pi) := \tilde{\sigma}$
    $s := \tilde{s} \cdot y^{-1}$
    return $(r, s)$
```

```
$\mathsf{Ext}(\sigma, \tilde{\sigma}, I_Y)$
    $(r, s) := \sigma$
    $(\tilde{r}, \tilde{s}, K, \pi) := \tilde{\sigma}$
    $y' := s^{-1} \cdot \tilde{s}$
    if $(I_Y, y') \in R_g$
        then return $y'$
    else return $\perp$
```

Fig. 2. ECDSA-based adaptor signature scheme $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}]$ instantiated with a hash function $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_p$.

3 Adaptor Signatures with Rerandomizable Keys

In this section we define the notion of adaptor signatures with rerandomizable keys and show how to instantiate it. Later in Sec. 4.1 we will use this primitive to generically construct adaptor wallets.

3.1 Definition

The notion of signature schemes with rerandomizable keys has first been introduced by Fleischhacker et al. [12] and has since been proven to be useful for the construction of deterministic wallet schemes (e.g., [5, 6]). Essentially, a signature scheme with rerandomizable keys extends regular signature schemes by two deterministic algorithms, a public key and a secret key rerandomization algorithm, which on input a public key or a secret key respectively and a randomness, output rerandomized keys. Such keys, if rerandomized with the same randomness, constitute a new signing key pair, which is distributed identically to a freshly and independently generated signing key pair. These properties and the deterministic nature of the rerandomization make such signature schemes good candidates for the construction of deterministic wallets. In our work, we are concerned with adaptor signatures. Therefore, we define in the following the notion of adaptor signatures with rerandomizable keys.

Definition 8 (Adaptor signature scheme with rerandomizable keys). *An adaptor signature scheme with rerandomizable keys w.r.t. a hard relation*

R and a signature scheme $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ consists of six algorithms $\mathsf{RASig}_{R,\Sigma} = (\mathsf{RandSK}, \mathsf{RandPK}, \mathsf{pSign}, \mathsf{Adapt}, \mathsf{pVrfy}, \mathsf{Ext})$ where $(\mathsf{pSign}, \mathsf{Adapt}, \mathsf{pVrfy}, \mathsf{Ext})$ are the same algorithms as defined for adaptor signatures (cf. Def. 3). Assuming that the public parameters par define a randomness space $X := X(\mathsf{par})$, the remaining algorithms are defined as follows:

$\mathsf{RandSK}(sk, \rho)$: *The deterministic secret key rerandomization algorithm takes as input a secret key sk and a randomness $\rho \in X$ and outputs a rerandomized secret key sk'.*

$\mathsf{RandPK}(pk, \rho)$: *The deterministic public key rerandomization algorithm takes as input a public key pk and a randomness $\rho \in X$ and outputs a rerandomized public key pk'.*

An adaptor signature scheme with rerandomizable keys $\mathsf{RASig}_{R,\Sigma}$ must satisfy the following two correctness properties:

1. Pre-signature correctness *stating that for all $(sk, pk) \leftarrow \mathsf{Gen}(1^n)$, all $m \in \{0,1\}^*$, all $\rho \in X$ and all $(Y, y) \in R$, the rerandomized keys $sk' \leftarrow \mathsf{RandSK}(sk, \rho)$ and $pk' \leftarrow \mathsf{RandPK}(pk, \rho)$ satisfy:*

$$\Pr\left[\begin{matrix}\mathsf{pVrfy}(pk', m, Y, \tilde{\sigma}) = 1, \\ \mathsf{Verify}(pk', m, \sigma) = 1, (Y, y') \in R\end{matrix}\ \middle|\ \begin{matrix}\tilde{\sigma} \leftarrow \mathsf{pSign}(sk', m, Y), \\ \sigma := \mathsf{Adapt}(\tilde{\sigma}, y),\end{matrix}\ y' := \mathsf{Ext}(\sigma, \tilde{\sigma}, Y)\right] = 1.$$

2. (Perfect) rerandomizability of keys: *For all $(sk, pk) \in \mathsf{Gen}(1^n)$ and $\rho \leftarrow_\$ X$, the distributions of (sk', pk') and (sk'', pk'') are identical, where:*

$$(sk', pk') \leftarrow (\mathsf{RandSK}(sk, \rho), \mathsf{RandPK}(pk, \rho)),$$
$$(sk'', pk'') \leftarrow_\$ \mathsf{Gen}(1^n).$$

Like adaptor signatures, an $\mathsf{RASig}_{R,\Sigma}$ scheme must satisfy pre-signature adaptability.

Definition 9 (Pre-signature adaptability). *An adaptor signature scheme with perfectly rerandomizable keys $\mathsf{RASig}_{R,\Sigma}$ satisfies pre-signature adaptability if for any message $m \in \{0,1\}^*$, any statement/witness pair $(Y, y) \in R$, any public key pk and any pre-signature $\tilde{\sigma} \in \{0,1\}^*$ with $\mathsf{pVrfy}(pk, m, Y, \tilde{\sigma}) = 1$, we have $\mathsf{Verify}(pk, m, \mathsf{Adapt}(\tilde{\sigma}, y)) = 1$.*

For adaptor signatures with rerandomizable keys, we introduce the notions of *existential unforgeability under honestly rerandomizable keys* and *witness extractability under honestly rerandomizable keys*. These notions extend the respective security notions of adaptor signatures by allowing the adversary to not only obtain (pre-)signatures under sk but also under secret keys that constitute honest rerandomizations of sk. An honest rerandomization is one where the randomness has been chosen uniformly at random from the randomness space X. Further, in our security notions the adversary can win the game by providing a forgery either under sk or under any honestly rerandomized key. We formally describe these security notions in Fig. 3.

Definition 10 (Existential unforgeability under honestly rerandomizable keys). *An adaptor signature scheme with rerandomizable keys* $\mathsf{RASig}_{R,\Sigma}$ *is unforgeable if for every PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *there exists a negligible function* ν *such that:* $\Pr[\mathsf{aSigForge{-}hrk}_{\mathcal{A},\mathsf{RASig}_{R,\Sigma}}(n) = 1] \leq \nu(n)$*, where the experiment* $\mathsf{aSigForge{-}hrk}_{\mathcal{A},\mathsf{RASig}_{R,\Sigma}}$ *is defined as in Fig. 3.*

Definition 11 (Witness extractability under honestly rerandomizable keys). *An adaptor signature scheme with rerandomizable keys* $\mathsf{RASig}_{R,\Sigma}$ *is witness extractable if for every PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there exists a negligible function* ν *such that the following holds:* $\Pr[\mathsf{aWitExt{-}hrk}_{\mathcal{A},\mathsf{RASig}_{R,\Sigma}}(n) = 1] \leq \nu(n)$*, where the experiment* $\mathsf{aWitExt{-}hrk}_{\mathcal{A},\mathsf{RASig}_{R,\Sigma}}$ *is defined as in Fig. 3.*

```
$\mathsf{aSigForge{-}hrk}_{\mathcal{A},\mathsf{RASig}_{R,\Sigma}}(n)$
00 $\mathcal{Q} := \emptyset, \mathcal{R} := \emptyset$
01 $(sk, pk) \leftarrow \mathsf{Gen}(1^n)$
02 $(Y, y) \leftarrow \mathsf{GenR}(1^n)$
03 $(m^*, \rho^*, \mathsf{st}) \leftarrow \mathcal{A}_1^{\mathcal{O}}(pk, Y)$
04 $sk^* \leftarrow \mathsf{RandSK}(sk, \rho^*)$
05 $pk^* \leftarrow \mathsf{RandPK}(pk, \rho^*)$
06 $\tilde{\sigma}^* \leftarrow \mathsf{pSign}(sk^*, m^*, Y)$
07 $\sigma^* \leftarrow \mathcal{A}_2^{\mathcal{O}}(\tilde{\sigma}^*, \mathsf{st})$
08 $b_1 \leftarrow m^* \notin \mathcal{Q}$
09 $b_2 \leftarrow \mathsf{Verify}(pk^*, m^*, \sigma)$
10 $b_3 \leftarrow \rho^* \in \mathcal{R}$
11 Return $(b_1 \wedge b_2 \wedge b_3)$
```

```
$\mathsf{aWitExt{-}hrk}_{\mathcal{A},\mathsf{RASig}_{R,\Sigma}}(n)$
00 $\mathcal{Q} := \emptyset, \mathcal{R} := \emptyset$
01 $(sk, pk) \leftarrow \mathsf{Gen}(1^n)$
02 $(m^*, \rho^*, Y^*, \mathsf{st}) \leftarrow \mathcal{A}_1^{\mathcal{O}}(pk)$
03 $sk^* \leftarrow \mathsf{RandSK}(sk, \rho^*)$
04 $pk^* \leftarrow \mathsf{RandPK}(pk, \rho^*)$
05 $\tilde{\sigma}^* \leftarrow \mathsf{pSign}(sk^*, m^*, Y^*)$
06 $\sigma^* \leftarrow \mathcal{A}_2^{\mathcal{O}}(\tilde{\sigma}^*, \mathsf{st})$
07 $b_1 \leftarrow (Y^*, \mathsf{Ext}(\sigma^*, \tilde{\sigma}^*, Y^*)) \notin R$
08 $b_2 \leftarrow m^* \notin \mathcal{Q}$
09 $b_3 \leftarrow \mathsf{Verify}(pk^*, m^*, \sigma^*)$
10 $b_4 \leftarrow \rho^* \in \mathcal{R}$
11 $b_5 \leftarrow Y^* \in L_R$
12 Return $(b_1 \wedge b_2 \wedge b_3 \wedge b_4 \wedge b_5)$
```

```
Oracle $\mathtt{RSignO}(m, \rho)$
00 If $\rho \notin \mathcal{R}$ : return 0
01 $sk' \leftarrow \mathsf{RandSK}(sk, \rho)$
02 $\sigma \leftarrow \mathsf{Sign}(sk', m)$
03 $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
04 Return $\sigma$

Oracle $\mathtt{PreSignO}(m, Y, \rho)$
05 If $\rho \notin \mathcal{R}$ : return 0
06 $sk' \leftarrow \mathsf{RandSK}(sk, \rho)$
07 $\tilde{\sigma} \leftarrow \mathsf{pSign}(sk', m, Y)$
08 $\mathcal{Q} := \mathcal{Q} \cup \{m\}$
09 Return $\tilde{\sigma}$

Oracle $\mathtt{RandO}$
10 $\rho \leftarrow_\$ X$
11 $\mathcal{R} := \mathcal{R} \cup \{\rho\}$
12 Return $\rho$
```

Fig. 3. $\mathsf{aSigForge{-}hrk}$ and $\mathsf{aWitExt{-}hrk}$ games for an adaptor signature scheme with rerandomizable keys $\mathsf{RASig}_{R,\Sigma}$. In the above games we have $\mathcal{O} := \{\mathtt{RSignO}, \mathtt{PreSignO}, \mathtt{RandO}\}$.

3.2 Construction

In Fig. 4, we present an adaptor signature with rerandomizable keys $\mathsf{REC}_{R,\mathsf{PEC}}[\mathsf{H}]$ from the ECDSA-based adaptor signature $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}]$ from Fig. 2. Similar to the rerandomizable ECDSA construction of Das et al. [5], we use public key-prefixed messages in our construction which is required to ensure security (see the proof sketch of Thm. 1) and we use a hash function $\mathsf{H}\colon \{0,1\}^* \to \mathbb{Z}_p$.

In order to prove the security of our construction, we follow the approach of Das et al. [5], who presented a security proof of the plain ECDSA signature scheme with rerandomizable keys via a reduction to the (non-rerandomizable) ECDSA signature scheme. The main ingredient in their security proof is a related key attack which allows to transform a signature on message m_1 under

```
Algorithm $\mathsf{REC}_{R,\mathsf{PEC}}[\mathsf{H}].\mathsf{pSign}\,(sk, m, Y)$
  $\mathsf{pm} \leftarrow (pk, m)$
  $\tilde{\sigma} \leftarrow \mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}].\mathsf{pSign}\,(sk, \mathsf{pm}, Y)$
  Return $\tilde{\sigma}$

Algorithm $\mathsf{REC}_{R,\mathsf{PEC}}[\mathsf{H}].\mathsf{pVrfy}\,(pk, m, Y, \tilde{\sigma})$
  $\mathsf{pm} \leftarrow (pk, m)$
  Return $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}].\mathsf{pVrfy}\,(pk, \mathsf{pm}, Y, \tilde{\sigma})$

Algorithm $\mathsf{REC}_{R,\mathsf{PEC}}[\mathsf{H}].\mathsf{Adapt}\,(\tilde{\sigma}, y)$
  Return $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}].\mathsf{Adapt}\,(\tilde{\sigma}, y)$

Algorithm $\mathsf{REC}_{R,\mathsf{PEC}}[\mathsf{H}].\mathsf{RandSK}\,(sk, \rho)$
  $sk' \leftarrow sk \cdot \rho \bmod p$
  Return $sk'$

Algorithm $\mathsf{REC}_{R,\mathsf{PEC}}[\mathsf{H}].\mathsf{Sign}\,(sk, m)$
  $\mathsf{pm} \leftarrow (pk, m)$
  $\sigma \leftarrow \mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}].\mathsf{Sign}\,(sk, \mathsf{pm})$
  Return $\sigma$

Algorithm $\mathsf{REC}_{R,\mathsf{PEC}}[\mathsf{H}].\mathsf{Verify}\,(pk, \sigma, m)$
  $\mathsf{pm} \leftarrow (pk, m)$
  Return $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}].\mathsf{Verify}\,(pk, \sigma', \mathsf{pm})$

Algorithm $\mathsf{REC}_{R,\mathsf{PEC}}[\mathsf{H}].\mathsf{Ext}\,(\sigma, \tilde{\sigma}, Y)$
  Return $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}].\mathsf{Ext}\,(\sigma, \tilde{\sigma}, Y)$

Algorithm $\mathsf{REC}_{R,\mathsf{PEC}}[\mathsf{H}].\mathsf{RandPK}\,(pk, \rho)$
  $pk' \leftarrow pk^{\rho}$
  Return $pk'$
```

Fig. 4. Construction of a key-prefixed ECDSA-based adaptor signature scheme with perfectly rerandomizable keys $\mathsf{REC}_{R,\mathsf{PEC}}[\mathsf{H}]$ from the ECDSA-based adaptor signature scheme $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}]$ as described in Fig. 2. Both schemes are instantiated with a hash function $\mathsf{H}\colon \{0,1\}^* \to \mathbb{Z}_p$.

public key pk_1 to a signature for message m_0 under a related public key pk_0. We recall their transformation in the following (and formally in Fig. 5). Let $\mathsf{PEC}[\mathsf{H}_0]$ and $\mathsf{PEC}[\mathsf{H}_1]$ denote two (positive) ECDSA signature schemes instantiated with hash functions H_0 and H_1 respectively. Then the authors show that if $pk_1 = (pk_0)^{\rho}$ where $\rho = \frac{\mathsf{H}_1(m_1)}{\mathsf{H}_0(m_0)} \in \mathbb{Z}_p$ and given a valid signature σ_1 (i.e., $\mathsf{PEC}[\mathsf{H}_1].\mathsf{Verify}(pk_1, m_1, \sigma_1) = 1$), the algorithm $\mathsf{Trf}[\mathsf{H}_0, \mathsf{H}_1](m_0, m_1, \sigma_1, \rho, pk_0, pk_1)$ returns a valid signature σ_0 under pk_0 and m_0, i.e., $\mathsf{PEC}[\mathsf{H}_0].\mathsf{Verify}(pk_0, m_0, \sigma_0) = 1$. Let us now recall the formal lemma of Das et al. for this transformation.

Lemma 1. *Consider the algorithm* $\mathsf{Trf}[\mathsf{H}_0, \mathsf{H}_1]$ *in Figure 5. Suppose that:*

– $\rho = \frac{\mathsf{H}_1(m_1)}{\mathsf{H}_0(m_0)} \in \mathbb{Z}_p$,
– $pk_0, pk_1 \in \mathbb{G}$ *s.t.* $pk_0 = g^{x_0}$ *and* $pk_1 = pk_0^{\rho}$,
– $\mathsf{PEC}[\mathsf{H}_1].\mathsf{Verify}(pk_1, m_1, \sigma_1) = 1$,
– $\sigma_0 \leftarrow \mathsf{Trf}[\mathsf{H}_0, \mathsf{H}_1](m_0, m_1, \sigma_1, \rho, pk_0, pk_1)$.

Then $\mathsf{PEC}[\mathsf{H}_0].\mathsf{Verify}(pk_0, m_0, \sigma_0) = 1$.

This lemma has been previously proven by Das et al. [5].

We show that a similar transformation can be applied to the ECDSA-based adaptor signature scheme $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}]$ to transform pre-signatures. Since pre-signatures in this scheme include a zero-knowledge proof, it is not immediately clear that such a transformation goes through. We next give the lemma for the pre-signature transformation as well as the proof for the lemma.

Lemma 2. *Let* $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}_0]$ *and* $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}_1]$ *denote two ECDSA-based adaptor signature schemes according to Fig. 2 instantiated with hash functions* H_0 *and* H_1*. Consider the algorithm* $\mathsf{ATrf}[\mathsf{H}_0, \mathsf{H}_1]$ *in Figure 5. Suppose that:*

```
$\mathsf{Trf}[\mathsf{H}_0,\mathsf{H}_1](m_0, m_1, \sigma_1, \rho, pk_0, pk_1)$
    $z_0 \leftarrow \mathsf{H}_0(m_0)$
    $z_1 \leftarrow \mathsf{H}_1(m_1)$
    If $\big(\mathsf{PEC}_{R_g,\mathsf{PEC}}[\mathsf{H}_1].\mathsf{Verify}(pk_1, \sigma_1, m_1) = 0\big) \vee \Big(\rho \neq \frac{z_1}{z_0} \vee pk_1 \neq pk_0^{\rho}\Big)$ :
        Return $\perp$
    $(r, s_1) \leftarrow \sigma_1$
    $s_0 \leftarrow \frac{s_1}{\rho} \bmod p$
    $\sigma_0 \leftarrow (r, s_0)$
    Return $\sigma_0$
```

```
$\mathsf{ATrf}[\mathsf{H}_0,\mathsf{H}_1](m_0, m_1, \tilde{\sigma}_1, \rho, pk_0, pk_1, I_Y)$
    $z_0 \leftarrow \mathsf{H}_0(m_0)$
    $z_1 \leftarrow \mathsf{H}_1(m_1)$
    If $\big(\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}_1].\mathsf{pVrfy}(pk_1, m_1, I_Y, \tilde{\sigma}_1\big) \vee \Big(\rho \neq \frac{z_1}{z_0} \vee pk_1 \neq pk_0^{\rho} \vee I_Y \notin L_R\Big)$ :
        Return $\perp$
    $(r, \tilde{s}_1, K, \pi) \leftarrow \tilde{\sigma}_1$
    $\tilde{s}_0 \leftarrow \frac{\tilde{s}_1}{\rho} \bmod p$
    $\tilde{\sigma}_0 \leftarrow (r, \tilde{s}_0, K, \pi)$
    Return $\tilde{\sigma}_0$
```

Fig. 5. Figure shows the $\mathsf{Trf}[\mathsf{H}_0,\mathsf{H}_1]$ and $\mathsf{ATrf}[\mathsf{H}_0,\mathsf{H}_1]$ algorithms for hash functions $\mathsf{H}_0, \mathsf{H}_1 \colon \{0,1\}^* \to \mathbb{Z}_p$.

- $I_Y \in L_{R_Y}$, $\rho = \frac{\mathsf{H}_1(m_1)}{\mathsf{H}_0(m_0)} \in \mathbb{Z}_p$,
- $pk_0, pk_1 \in \mathbb{G}$ *s.t.* $pk_0 = g^{x_0}$ *and* $pk_1 = pk_0^{\rho}$,
- $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}_1].\mathsf{pVrfy}(pk_1, m_1, I_Y, \tilde{\sigma}_1) = 1$,
- $\tilde{\sigma}_0 \leftarrow \mathsf{ATrf}[\mathsf{H}_0,\mathsf{H}_1](m_0, m_1, \tilde{\sigma}_1, \rho, pk_0, pk_1, I_Y)$.

Then $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}_0].\mathsf{pVrfy}(pk_0, m_0, I_Y, \tilde{\sigma}_0) = 1$.

We would like to point out that Lemma 2 requires that after the transformation, the new pre-signature $\tilde{\sigma}_0$ is indeed valid with respect to the same statement I_Y. In other words, given the witness y, both $\tilde{\sigma}_0$ and $\tilde{\sigma}_1$ can be adapted into full signatures under pk_0 and pk_1 respectively.

Proof. The proof of this lemma is similar to the proof of Lemma 1 from [5]. To prove the lemma, we have to show that given a statement $I_Y := (Y, \pi_Y) \in L_R$, a public key $pk_1 = pk_0^{\rho}$ where $\rho = \frac{\mathsf{H}_1(m_1)}{\mathsf{H}_0(m_0)}$ and a pre-signature $\tilde{\sigma}_1$ such that $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}_1].\mathsf{pVrfy}(pk_1, m_1, I_Y, \tilde{\sigma}_1) = 1$, $\mathsf{ATrf}[\mathsf{H}_0,\mathsf{H}_1]$ outputs a pre-signature $\tilde{\sigma}_0$ such that $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}_0].\mathsf{pVrfy}(pk_0, m_0, I_Y, \tilde{\sigma}_0) = 1$. Recall that for the pre-signature $\tilde{\sigma}_1 := (r, \tilde{s}_1, K, \pi)$ it holds that $\tilde{s}_1 = k^{-1}(\mathsf{H}_1(m) + r \cdot sk_1)$, $r := f(K)$, $K := Y^k$ and π is a valid proof that $(\tilde{K}, K)$ is a valid statement in R_Y. Then $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}_0].\mathsf{pVrfy}(pk_0, m_0, Y, \tilde{\sigma}_0)$ computes the following:

$$\begin{aligned} K' &= g^u \cdot pk_0^v = g^{(\mathsf{H}_0(m_0)\cdot \tilde{s}_0^{-1})} \cdot pk_0^{r\cdot \tilde{s}_0^{-1}} = g^{\tilde{s}_0^{-1}\cdot(\mathsf{H}_0(m_0)+x_0\cdot r)} \\ &= g^{\frac{\rho}{\tilde{s}_1}\cdot(\mathsf{H}_1(m_1)\cdot\rho^{-1}+x_1\cdot\rho^{-1}\cdot r)} = g^{\frac{\rho}{k^{-1}(\mathsf{H}_1(m_1)+x_1\cdot r)}\cdot(\mathsf{H}_1(m_1)+x_1\cdot r)\cdot\rho^{-1}} = g^{\frac{\rho\cdot\rho^{-1}}{k^{-1}}} = g^k \end{aligned}$$

Therefore, the zero-knowledge proof π is valid w.r.t. the statement (K', K) where $K' = g^k$ and $K = Y^k$. We can conclude that the pre-signature $\tilde{\sigma}_0 \leftarrow \mathsf{ATrf}[\mathsf{H}_0,\mathsf{H}_1](m_0, m_1, \tilde{\sigma}_1, \rho, pk_0, pk_1, I_Y)$ with $\tilde{\sigma}_0 := (r, \frac{\tilde{s}_1}{\rho}, K, \pi)$ constitutes a valid pre-signature w.r.t. public key pk_0, message m_0 and statement I_Y.

Theorem 1. *Let* $\mathsf{H}_0 \colon \{0,1\}^* \to \mathbb{Z}_p$, $\mathsf{H}_1 \colon \{0,1\}^* \to \mathbb{Z}_p$ *be hash functions modeled as random oracle and let* $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}_0]$ *be the secure ECDSA-based adaptor signature as per Fig. 2. Then the construction* $\mathsf{REC}_{R_g,\mathsf{PEC}}[\mathsf{H}_1]$ *as described in Fig. 4 is existentially unforgeable under honestly rerandomizable keys as per Def. 10.*

Proof (Sketch). The proof of this theorem is similar to the proof of the multiplicatively rerandomizable ECDSA signature scheme as provided by Das et al. [5]. In their proof, the authors show unforgeability of an ECDSA scheme with rerandomizable keys by exhibiting a reduction to the unforgeability of the regular ECDSA signature scheme. The proof of Das et al. relies crucially on the related key attack as depicted by the algorithm $\mathsf{Trf}[\mathsf{H}_0, \mathsf{H}_1]$ in Fig. 5, which allows to transform a signature under a public key pk to a valid signature under a related public key $pk' \leftarrow pk^{\rho'}$, if ρ' has a certain structure. In more details, Das et al. instantiate the ECDSA scheme with a hash function H_0 and the ECDSA scheme with rerandomizable keys with a hash function H_1 (both hash functions are modeled as random oracles). They then program the random oracle H_1 in such a way that on input $m' = (pk', m)$, where pk' is a public key rerandomized with randomness ρ' (i.e., $pk' = pk^{\rho'}$), it holds $\mathsf{H}_1(m') = \mathsf{H}_0(m) \cdot \rho'$. This allows the reduction to transform signatures for rerandomized public keys to signatures for the original public key and vice versa using algorithm $\mathsf{Trf}[\mathsf{H}_0, \mathsf{H}_1]$.

In our proof, we can show unforgeability of the $\mathsf{REC}_{R_g,\mathsf{PEC}}[\mathsf{H}_1]$ scheme via a reduction to the unforgeability of the ECDSA-based adaptor signature scheme $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}_0]$. The main difference in our proof as compared to the proof of Das et al. arises from the fact that we need to apply the related key attack on pre-signatures as well. This transformation requires us to use the algorithm $\mathsf{ATrf}[\mathsf{H}_0, \mathsf{H}_1]$ as described in Fig. 5. To apply this transformation, we program the random oracle H_1 in exactly the same way as is done in the proof of Das et al. and hence, the programming of H_1 is consistent for signatures and pre-signatures.

Theorem 2. *Let* $\mathsf{H}_0 \colon \{0,1\}^* \to \mathbb{Z}_p$, $\mathsf{H}_1 \colon \{0,1\}^* \to \mathbb{Z}_p$ *be hash functions modeled as random oracle and let* $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}_0]$ *be the secure ECDSA-based adaptor signature as per Fig. 2. Then the construction* $\mathsf{REC}_{R_g,\mathsf{PEC}}[\mathsf{H}_1]$ *as described in Fig. 4 is witness extractable under honestly rerandomizable keys as per Def. 11.*

Proof (Sketch). The proof of this theorem is similar to the proof of Thm. 1. Here we must provide a reduction to the witness extractability property $\mathsf{aWitExt}$ of the ECDSA-based adaptor signature scheme $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}_0]$. The main difference, however, is that we have to show that a valid forgery in game $\mathsf{aWitExt{-}hrk}$ for scheme $\mathsf{REC}_{R_g,\mathsf{PEC}}[\mathsf{H}_1]$ can be transformed into a valid forgery in game $\mathsf{aWitExt}$ for scheme $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}_0]$. Recall that for a valid forgery σ^* in game $\mathsf{aWitExt{-}hrk}$ and given the corresponding pre-signature $\tilde{\sigma}^*$, it must hold that $(I_Y^*, \mathsf{REC}_{R_g,\mathsf{PEC}}[\mathsf{H}_1].\mathsf{Ext}(\sigma^*, \tilde{\sigma}^*, I_Y^*)) \notin R_g$. Therefore, we must show that applying the transformations $\mathsf{Trf}[\mathsf{H}_0, \mathsf{H}_1]$ and $\mathsf{ATrf}[\mathsf{H}_0, \mathsf{H}_1]$ from Fig. 5 on σ^* and $\tilde{\sigma}^*$ respectively preserves the above condition w.r.t. scheme $\mathsf{REC}_{R_g,\mathsf{PEC}}[\mathsf{H}_0]$. We show this via the following claim, for which we assume that $m_0, m_1 \in \{0,1\}^*$ are two messages, $\rho^* = \frac{\mathsf{H}_1(m_1)}{\mathsf{H}_0(m_0)} \in \mathbb{Z}_p$ and $pk^* = pk_{\mathsf{aWitExt}}^{\rho^*}$, where pk_{aWitExt} is the public key in game $\mathsf{aWitExt}$.

Claim 1 *If it holds that* $(I_Y^*, \mathsf{REC}_{R_g,\mathsf{PEC}}[\mathsf{H}_1].\mathsf{Ext}(\sigma^*, \tilde{\sigma}^*, I_Y^*)) \notin R_g$ *then we have* $(I_Y^*, \mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}_0].\mathsf{Ext}(\sigma', \tilde{\sigma}', I_Y^*)) \notin R_g$, *where*

$$\sigma' \leftarrow \mathsf{Trf}[\mathsf{H}_0, \mathsf{H}_1](m_0, m_1, \sigma^*, \rho^*, pk_{\mathsf{aWitExt}}, pk^*)$$

$$\tilde{\sigma}' \leftarrow \mathsf{ATrf}[\mathsf{H}_0, \mathsf{H}_1](m_0, m_1, \tilde{\sigma}^*, \rho^*, pk_{\mathsf{aWitExt}}, pk^*, I_Y^*).$$

Let $\sigma^* = (r, s)$ and $\tilde{\sigma}^* = (r, \tilde{s}, K, \pi)$, then we have: $\sigma' := (r, \frac{s}{\rho^*}), \tilde{\sigma}' := (r, \frac{\tilde{s}}{\rho^*}, K, \pi)$. Therefore, we can conclude that:

$$\mathsf{REC}_{R_g,\mathsf{PEC}}[\mathsf{H}_1].\mathsf{Ext}(\sigma^*, \tilde{\sigma}^*, I_Y^*) = s^{-1}\tilde{s} = \left(\frac{s}{\rho^*}\right)^{-1} \frac{\tilde{s}}{\rho^*} = \mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}_0].\mathsf{Ext}(\sigma', \tilde{\sigma}', I_Y^*)$$

Hence, we can conclude that if $(I_Y^*, \mathsf{REC}_{R_g,\mathsf{PEC}}[\mathsf{H}_1].\mathsf{Ext}(\sigma^*, \tilde{\sigma}^*, I_Y^*)) \notin R_g$ then $(I_Y^*, \mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}_0].\mathsf{Ext}(\sigma', \tilde{\sigma}', I_Y^*)) \notin R_g$. And thus, a forgery in game $\mathsf{aWitExt{-}hrk}$ can be transformed into a valid forgery in game $\mathsf{aWitExt}$.

We note that pre-signature adaptability (cf. Def. 9) of $\mathsf{REC}_{R_g,\mathsf{PEC}}$ follows immediately from the pre-signature adaptability property of $\mathsf{EC}_{R_g,\mathsf{PEC}}$.

3.3 Discussion

Note that our ECDSA-based instantiation of an adaptor signature with rerandomizable keys is compatible with a plethora of cryptocurrencies, since many cryptocurrency networks, including Bitcoin and Ethereum, rely on the ECDSA signature scheme. In our instantiation, we use a multiplicative key rerandomization instead of an additive one. This seemingly insignificant difference has a crucial impact on the security of the resulting scheme as shown by Das et al. [6]. More concretely, Das et al. presented an ECDSA scheme with additive key rerandomization, which incurred a security loss in the number of rerandomized keys, whereas the ECDSA scheme with multiplicative rerandomization from [5] does not incur such a loss.[*] In a nutshell, this security loss stems from the related key attack that is required to prove security of the additively rerandomizable scheme. Since the security proof for ECDSA-based adaptor signatures with rerandomizable keys would rely on the same related key attack, a similar security loss can be expected for the additively rerandomizable ECDSA-based adaptor signature. Worse yet, the related key attack for additively rerandomizable ECDSA allows to prove only *one-per-message unforgeability* [11], which is a weaker security notion than standard unforgeability. Therefore, we used multiplicative rerandomization in our instantiation.

While we did not work out the details, it is likely that adaptor signatures with rerandomizable keys can be constructed from Schnorr and Katz-Wang-based adaptor signatures [9] (due to the existing related key attack for Schnorr signatures as presented in [12]). Finally, we believe that it would be an interesting future work to extend the notion of two-party adaptor signatures as presented in [9] to two-party adaptor signatures with rerandomizable keys.

[*]Das et al. show that this loss results in 20 bits less security for certain parameters. We refer the reader to [6] for details.

4 Adaptor Wallets

In this section, we introduce the idea of adaptor wallets, which securely maintain and operate adaptor signature schemes in a cryptocurrency network. We first provide our model and then present a generic wallet construction from any adaptor signature scheme with rerandomizable keys and witness rerandomizable hard relation. Finally, we show that it is impossible to achieve deterministic and independent statement/witness rerandomization in our model. In Appx. A we provide the security arguments for our generic construction.

4.1 Adaptor Wallet Model

We now describe a model for adaptor wallets and we discuss how adaptor signature schemes with rerandomizable keys can be used to instantiate such a wallet. Our notion of adaptor wallets resembles the notion of hierarchical deterministic wallets by Das et al. [6], however, extending their notion to support adaptor signature operations such as pre-signing. We describe here the formal model and show a construction from adaptor signatures with rerandomizable keys.

An adaptor wallet considers one *master wallet*, which is used to deterministically initialize new wallets, so-called *child wallets*. Such child wallets are then used to generate (adaptor) signatures and are identified in our model by an identifier ID. In more detail, the master wallet generates and stores a master key pair $(\mathsf{msk}, \mathsf{mpk})$, a state St and a master statement/witness pair (Y_m, y_m) of a hard relation. However, the master wallet is not used to generate signatures, but only to deterministically initialize child wallets, i.e., in order to initialize a child wallet with identifier ID, the master wallet deterministically derives a new key pair $(sk^{\mathsf{ID}}, pk^{\mathsf{ID}})$ from $(\mathsf{msk}, \mathsf{mpk})$, and a new statement/witness pair $(Y^{\mathsf{ID}}, y^{\mathsf{ID}})$ from (Y_m, y_m). The child wallet can then use its key pair $(sk^{\mathsf{ID}}, pk^{\mathsf{ID}})$ to generate signatures and its statement/witness pair $(Y^{\mathsf{ID}}, y^{\mathsf{ID}})$ and a counter ctr to deterministically derive further statement/witness pairs.

To keep our model simple, we do not allow child wallets to initialize further child wallets (as is done in the fully hierarchical setting [6]). We note, however, that our model can be extended to the fully hierarchical setting.

Similarly to the model of hierarchical deterministic wallets [6], we consider two kinds of child wallets, namely (1) non-hardened wallets, and (2) hardened wallets. Broadly speaking, the difference between these two is that we allow full corruption of hardened wallets, i.e., in our security games we allow the adversary to learn all secret values stored in a hardened wallet, including the session secret key sk^{ID}. For non-hardened wallets, on the other hand, we allow the adversary to only learn the session public key pk^{ID} and statement Y^{ID}. As a motivation for these two kinds of child wallets, recall the main applications of adaptor signatures as mentioned in the Introduction, namely payment channels and atomic swaps. A payment channel is typically used for frequent micropayments, i.e., users deposit only small amounts of money in a channel and use it often to sign transactions. In this case, it would be sensible to assume that the user operates the corresponding wallet on a mobile device, as it has to sign many transactions (possibly at remote locations) and the impact of a wallet corruption is limited. Such a wallet would be represented by a hardened wallet in our model. On the other hand, atomic swaps are used, e.g., to swap coins of one cryptocurrency with coins of another currency. Such swaps are often one-time transactions of large amounts of funds or valuable tokens. In this example, it seems reasonable to implement the corresponding wallet as a hot/cold wallet, as it is crucial to secure such large amounts of funds or valuable tokens in the best possible way. The security goal for an adaptor wallet scheme is that the full corruption of hardened wallets does not compromise the security of any other (child or master) wallet. Additionally, we require that for all uncorrupted wallets, the derived public keys and statement/witness pairs are indistinguishable from freshly generated public keys and statement/witness pairs. Lastly, adaptor wallets must satisfy security notions similar to witness extractability under honestly rerandomizable keys (cf. Def. 11) and pre-signature adaptability (cf. Def. 9) of adaptor signatures with rerandomizable keys. Fig. 6 gives an illustration of our wallet model.

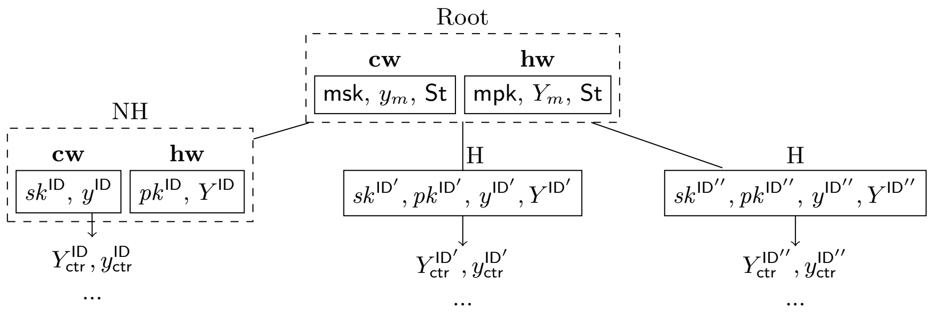
Root
cw
hw
msk, y_m, St
mpk, Y_m, St
NH
cw
hw
sk^{ID}, y^{ID}
pk^{ID}, Y^{ID}
$Y^{\mathsf{ID}}_{\mathsf{ctr}}, y^{\mathsf{ID}}_{\mathsf{ctr}}$
...
H
$sk^{\mathsf{ID}'}, pk^{\mathsf{ID}'}, y^{\mathsf{ID}'}, Y^{\mathsf{ID}'}$
$Y^{\mathsf{ID}'}_{\mathsf{ctr}}, y^{\mathsf{ID}'}_{\mathsf{ctr}}$
...
H
$sk^{\mathsf{ID}''}, pk^{\mathsf{ID}''}, y^{\mathsf{ID}''}, Y^{\mathsf{ID}''}$
$Y^{\mathsf{ID}''}_{\mathsf{ctr}}, y^{\mathsf{ID}''}_{\mathsf{ctr}}$
...

Fig. 6. Exemplatory design of our adaptor wallet scheme with three child wallets. H and NH denote hardened and non-hardened nodes respectively, **cw** and **hw** denote cold and hot wallets respectively and the values below the child wallets (e.g. $y^{\mathsf{ID}}_{\mathsf{ctr}}, Y^{\mathsf{ID}}_{\mathsf{ctr}}$) illustrate the statement/witness pairs that are being derived within each child wallet.

Statement/Witness rerandomization. According to the hot/cold wallet setting, it would be ideal if the deterministic derivation of statements and witnesses can be done independently. That is, we would like to store and derive statements exclusively on the hot wallet and witnesses only on the cold wallet. This would allow the cold wallet to stay entirely inactive (and therefore secure) in applications where it suffices to derive statements first and the corresponding witnesses only at a later time. Surprisingly, we show in Sec. 4.3 that for any multiplicative or additive statement/witness derivation, such an independent derivation is impossible. In our model and construction, we therefore resort to a joint statement/witness derivation.

Let us now formally define the notion of an adaptor wallet. An adaptor wallet scheme consists of a Setup algorithm, which initializes the master wallet, derivation algorithms for hardened and non-hardened keys ($\mathsf{SKDer}_{\mathsf{H}}, \mathsf{PKDer}_{\mathsf{H}}, \mathsf{SKDer}_{\mathsf{NH}}$,

$\mathsf{PKDer_{NH}}$) as well as for statement/witness pairs RDer, adaptor signature algorithms ($\mathsf{pSign}, \mathsf{pVrfy}, \mathsf{Adapt}, \mathsf{Ext}$) and signing and verification algorithms (Sign, Verify). In the following, we assume that public parameters par are known to all algorithms and we define secret and public key sets $\mathcal{SK}$ and $\mathcal{PK}$ respectively. Formally we have:

Definition 12 (Adaptor wallet). *An adaptor wallet scheme is defined w.r.t. a hard relation R and consists of algorithms $\mathsf{ADWal}_R = (\mathsf{Setup}, \mathsf{SKDer_H}, \mathsf{SKDer_{NH}}, \mathsf{PKDer_H}, \mathsf{PKDer_{NH}}, \mathsf{RDer}, \mathsf{pSign}, \mathsf{pVrfy}, \mathsf{Adapt}, \mathsf{Ext}, \mathsf{Sign}, \mathsf{Verify})$ which are defined as follows:*

– *The following algorithm describes the setup procedure of the adaptor wallet scheme.*

  $\mathsf{Setup}(1^n)$**:** *The probabilistic setup algorithm takes as input a security parameter n and outputs a master key pair $(\mathsf{msk}, \mathsf{mpk}) \in \mathcal{SK} \times \mathcal{PK}$, a state $\mathsf{St} \in \{0,1\}^*$ and a master statement/witness pair $(Y_m, y_m) \in R$.*

– *The following algorithms describe the deterministic derivation of keys and statement/witness pairs.*

  $\mathsf{SKDer_H}(\mathsf{msk}, \mathsf{St}, \mathsf{ID})$**:** *The deterministic hardened secret key derivation algorithm takes as input a master secret key $\mathsf{msk} \in \mathcal{SK}$, a state $\mathsf{St} \in \{0,1\}^*$ and an identifier $\mathsf{ID} \in \{0,1\}^*$. It outputs a secret key $sk^{\mathsf{ID}} \in \mathcal{SK}$.*

  $\mathsf{SKDer_{NH}}(\mathsf{msk}, \mathsf{mpk}, \mathsf{St}, \mathsf{ID})$**:** *The deterministic non-hardened secret key derivation algorithm takes as input a master secret key $\mathsf{msk} \in \mathcal{SK}$, a master public key $\mathsf{mpk} \in \mathcal{PK}$, a state $\mathsf{St} \in \{0,1\}^*$ and an identifier $\mathsf{ID} \in \{0,1\}^*$. It outputs a secret key $sk^{\mathsf{ID}} \in \mathcal{SK}$.*

  $\mathsf{PKDer_H}(\mathsf{msk}, \mathsf{mpk}, \mathsf{St}, \mathsf{ID})$**:** *The deterministic hardened public key derivation algorithm takes as input a master secret key $\mathsf{msk} \in \mathcal{SK}$, a master public key $\mathsf{mpk} \in \mathcal{PK}$, a state $\mathsf{St} \in \{0,1\}^*$ and an identifier $\mathsf{ID} \in \{0,1\}^*$. It outputs a public key $pk^{\mathsf{ID}} \in \mathcal{PK}$.*

  $\mathsf{PKDer_{NH}}(\mathsf{mpk}, \mathsf{St}, \mathsf{ID})$**:** *The deterministic non-hardened public key derivation algorithm takes as input a master public key $\mathsf{mpk} \in \mathcal{PK}$, a state $\mathsf{St} \in \{0,1\}^*$ and an identifier $\mathsf{ID} \in \{0,1\}^*$. It outputs a public key $pk^{\mathsf{ID}} \in \mathcal{PK}$.*

  $\mathsf{RDer}(Y, y, \mathbf{ctr}, \mathsf{ID})$**:** *The deterministic statement/witness derivation algorithm takes as input a statement/witness pair $(Y, y) \in R$, a counter $\mathsf{ctr} \in \{0,1\}^*$ and an identifier $\mathsf{ID} \in \{0,1\}^*$. It outputs a statement/witness $(Y^{\mathsf{ID}}_{\mathsf{ctr}}, y^{\mathsf{ID}}_{\mathsf{ctr}})$.*

– *The following algorithms describe the adaptor signature procedures.*

  $\mathsf{pSign}(sk^{\mathsf{ID}}, m, Y^{\mathsf{ID}}_{\mathbf{ctr}})$**:** *The probabilistic pre-signing algorithm takes as input a secret key $sk^{\mathsf{ID}} \in \mathcal{SK}$, a message $m \in \{0,1\}^*$ and a statement $Y^{\mathsf{ID}}_{\mathsf{ctr}} \in L_R$. It outputs a pre-signature $\tilde{\sigma}$.*

  $\mathsf{pVrfy}(pk^{\mathsf{ID}}, m, Y^{\mathsf{ID}}_{\mathbf{ctr}}, \tilde{\sigma})$**:** *The deterministic pre-verification algorithm takes as input a public key $pk^{\mathsf{ID}} \in \mathcal{PK}$, a message $m \in \{0,1\}^*$, a statement $Y^{\mathsf{ID}}_{\mathsf{ctr}} \in L_R$ and a pre-signature $\tilde{\sigma}$. It outputs 0 or 1.*

$\mathsf{Adapt}(\tilde{\sigma}, y^{\mathsf{ID}}_{\mathsf{ctr}})$**:** *The deterministic adapting algorithm takes as input a pre-signature* $\tilde{\sigma}$ *and a witness* $y^{\mathsf{ID}}_{\mathsf{ctr}}$*. It outputs a signature* σ*.*

$\mathsf{Ext}(\sigma, \tilde{\sigma}, Y^{\mathsf{ID}}_{\mathsf{ctr}})$**:** *The deterministic extracting algorithm takes as input a signature* σ*, a pre-signature* $\tilde{\sigma}$ *and a statement* $Y^{\mathsf{ID}}_{\mathsf{ctr}} \in L_R$*. It outputs a witness* $y^{\mathsf{ID}}_{\mathsf{ctr}}$ *such that* $(Y^{\mathsf{ID}}_{\mathsf{ctr}}, y^{\mathsf{ID}}_{\mathsf{ctr}}) \in R$*, or* $\perp$*.*

– *The following algorithms describe the relevant procedures for signing and verification.*

$\mathsf{Sign}(sk^{\mathsf{ID}}, m)$**:** *The probabilistic signing algorithm takes as input a secret key* $sk^{\mathsf{ID}} \in \mathcal{SK}$ *and a message* $m \in \{0,1\}^*$*. It outputs a signature* σ*.*

$\mathsf{Verify}(pk^{\mathsf{ID}}, m, \sigma)$**:** *The deterministic verification algorithm takes as input a public key* pk^{ID}*, a message* $m \in \{0,1\}^*$ *and a signature* σ*. It outputs* 0 *or* 1*.*

An adaptor wallet scheme is *correct*, if (1) statement/witness pairs that are derived by the algorithm RDer form valid statement/witness pairs for the hard relation R, (2) secret/public key pairs that are derived by the algorithms $\mathsf{SKDer_H}, \mathsf{PKDer_H}$ and $\mathsf{SKDer_{NH}}, \mathsf{PKDer_{NH}}$ respectively form valid signing key pairs, and (3) derived statement/witness pairs and derived secret/public key pairs satisfy pre-signature correctness in the sense of an adaptor signature scheme.

We denote keys with subscript nh (e.g., $sk^{\mathsf{ID}}_{\mathsf{nh}}$ or $pk^{\mathsf{ID}}_{\mathsf{nh}}$) as *non-hardened* keys and keys with subscript h (e.g., $sk^{\mathsf{ID}}_{\mathsf{h}}$ or $pk^{\mathsf{ID}}_{\mathsf{h}}$) as *hardened* keys.

Definition 13 (Correctness of adaptor wallets). *Let* ADWal_R *be an adaptor wallet scheme. For* $n \in \mathbb{N}$*, any* $\mathsf{ID} \in \{0,1\}^*$ *and any* $(\mathsf{msk}, \mathsf{mpk}, \mathsf{St}, Y_m, y_m) \in \mathsf{Setup}(1^n)$*, we define tuples* $(sk^{\mathsf{ID}}_{\mathsf{h}}, pk^{\mathsf{ID}}_{\mathsf{h}})$ *as*

$$\begin{aligned} sk^{\mathsf{ID}}_{\mathsf{h}} &:= \mathsf{SKDer_H}(\mathsf{msk}, \mathsf{St}, \mathsf{ID}) \\ pk^{\mathsf{ID}}_{\mathsf{h}} &:= \mathsf{PKDer_H}(\mathsf{msk}, \mathsf{mpk}, \mathsf{St}, \mathsf{ID}) \end{aligned}$$

and tuples $(sk^{\mathsf{ID}}_{\mathsf{nh}}, pk^{\mathsf{ID}}_{\mathsf{nh}})$ *as*

$$\begin{aligned} sk^{\mathsf{ID}}_{\mathsf{nh}} &:= \mathsf{SKDer_{NH}}(\mathsf{msk}, \mathsf{mpk}, \mathsf{St}, \mathsf{ID}) \\ pk^{\mathsf{ID}}_{\mathsf{nh}} &:= \mathsf{PKDer_{NH}}(\mathsf{mpk}, \mathsf{St}, \mathsf{ID}). \end{aligned}$$

Further, for any $\mathit{ctr} \in \{0,1\}^*$ *we define tuples* $\left(Y^{\mathsf{ID}}_{\mathit{ctr}}, y^{\mathsf{ID}}_{\mathit{ctr}}\right) := \mathsf{RDer}(Y^{\mathsf{ID}}, y^{\mathsf{ID}}, \mathit{ctr}, \mathsf{ID})$ *where* $(Y^{\mathsf{ID}}, y^{\mathsf{ID}}) := \mathsf{RDer}(Y_m, y_m, 0, \mathsf{ID})$*.*

ADWal_R *is* correct *if for all* $m \in \{0,1\}^*$*, all* $\mathsf{ID} \in \{0,1\}^*$*, all* $\mathit{ctr} \in \{0,1\}^*$ *and all* $(\mathsf{msk}, \mathsf{mpk}, \mathsf{St}, Y_m, y_m) \in \mathsf{Setup}(1^n)$ *the following four conditions hold:*

$$\Pr\left[\begin{array}{l} \mathsf{pVrfy}(pk^{\mathsf{ID}}_{\mathsf{h}}, m, Y^{\mathsf{ID}}_{\mathit{ctr}}, \tilde{\sigma}) = 1, \\ \mathsf{Verify}(pk^{\mathsf{ID}}_{\mathsf{h}}, m, \sigma) = 1, (Y^{\mathsf{ID}}_{\mathit{ctr}}, y') \in R \end{array}\middle|\begin{array}{l} \tilde{\sigma} \leftarrow \mathsf{pSign}(sk^{\mathsf{ID}}_{\mathsf{h}}, m, Y^{\mathsf{ID}}_{\mathit{ctr}}) \\ \sigma := \mathsf{Adapt}(pk^{\mathsf{ID}}_{\mathsf{h}}, \tilde{\sigma}, y^{\mathsf{ID}}_{\mathit{ctr}}), \\ y' := \mathsf{Ext}(pk^{\mathsf{ID}}_{\mathsf{h}}, \sigma, \tilde{\sigma}, Y^{\mathsf{ID}}_{\mathit{ctr}}) \end{array}\right] = 1. \quad (1)$$

$$\Pr\left[\begin{array}{l}\mathsf{pVrfy}(pk_{\mathsf{nh}}^{\mathsf{ID}}, m, Y_{ctr}^{\mathsf{ID}}, \tilde{\sigma}) = 1, \\ \mathsf{Verify}(pk_{\mathsf{nh}}^{\mathsf{ID}}, m; \sigma) = 1, (Y_{ctr}^{\mathsf{ID}}, y') \in R\end{array}\middle|\begin{array}{l}\tilde{\sigma} \leftarrow \mathsf{pSign}(sk_{\mathsf{nh}}^{\mathsf{ID}}, m, Y_{ctr}^{\mathsf{ID}}) \\ \sigma := \mathsf{Adapt}(pk_{\mathsf{nh}}^{\mathsf{ID}}, \tilde{\sigma}, y_{ctr}^{\mathsf{ID}}), \\ y' := \mathsf{Ext}(pk_{\mathsf{nh}}^{\mathsf{ID}}, \sigma, \tilde{\sigma}, Y_{ctr}^{\mathsf{ID}})\end{array}\right] = 1. \quad (2)$$

$$\Pr\left[\mathsf{Verify}(pk_{\mathsf{h}}^{\mathsf{ID}}, m, \sigma) = 1 \,\middle|\, \sigma \leftarrow \mathsf{Sign}(sk_{\mathsf{h}}^{\mathsf{ID}}, m)\right] = 1. \quad (3)$$

$$\Pr\left[\mathsf{Verify}(pk_{\mathsf{nh}}^{\mathsf{ID}}, m, \sigma) = 1 \,\middle|\, \sigma \leftarrow \mathsf{Sign}(sk_{\mathsf{nh}}^{\mathsf{ID}}, m)\right] = 1. \quad (4)$$

Additionally, an adaptor wallet scheme must satisfy the properties *wallet unforgeability*, *wallet witness extractability*, *wallet unlinkability* and *wallet signature adaptability*. We formalize these properties via the games **wUfcma**, **wWitExt**, **wUnl** and **wAdapt** in Figures 8 and 9. The required oracles for these games are described in Fig. 7.

Oracles Before we formally define the security notions that an adaptor wallet scheme has to satisfy, we describe the oracles to which the adversary will obtain access in the security games. We give a formal description of the oracles in Fig. 7. The oracles `HKeyO` and `NHKeyO` allow the adversary to adaptively create new hardened or non-hardened child wallets for an adaptively chosen ID. The `LeakO` oracle allows the adversary to adaptively corrupt a hardened wallet and thereby to learn all secret values of this wallet, including its session secret key. Note that, according to our model, this oracle can only be called for *hardened wallets*. The `StLeakO` oracle leaks the state of the scheme. The oracles `SignO` and `PreSignO` allow the adversary to receive signatures and pre-signatures respectively for adaptively chosen messages, wallet identifiers and statements. Finally, the `HardRelO` oracle gives the adversary the ability to learn derived statement/witness pairs for adaptively chosen wallets and counters. This models the adversary's capability in certain use cases, such as the revocation process in payment channels, to learn statement/witness pairs of an uncorrupted wallet.

For all our games, we assume that the lists `HK`, `NHK`, $\mathcal{C}$, $\mathcal{Q}$ and $\mathcal{W}$ have been initialized to the emptyset $\emptyset$ (i.e., `HK` $:= \emptyset$, `NHK` $:= \emptyset$ etc.). The lists `HK` and `NHK` are used throughout the oracles and games to bookkeep the internal values of all hardened and non-hardened wallets, whereas $\mathcal{C}$ keeps track of all corrupted hardened wallets and $\mathcal{Q}$ stores for each wallet on which messages a (pre-) signature has already been generated. Finally, $\mathcal{W}$ keeps track of the statement/witness pairs per wallet that have been leaked to the adversary.

Wallet Unforgeability and Witness Extractability We now describe the security notions of wallet unforgeabilty and wallet witness extractability that an adaptor wallet has to satisfy. These two notions are formally defined in Fig. 8.

```
Oracle $\mathtt{HKeyO}(\mathsf{ID})$
If $\mathtt{NHK}[\mathsf{ID}] \neq \bot$: return $\bot$
If $\mathtt{HK}[\mathsf{ID}] \neq \bot$:
    $(\cdot, pk_{\mathsf{h}}^{\mathsf{ID}}, \cdot, \cdot, \cdot) := \mathtt{HK}[\mathsf{ID}]$
    return $pk_{\mathsf{h}}^{\mathsf{ID}}$
$sk_{\mathsf{h}}^{\mathsf{ID}} \leftarrow \mathsf{SKDer}_{\mathsf{H}}(\mathsf{msk}, \mathsf{St}, \mathsf{ID})$
$pk_{\mathsf{h}}^{\mathsf{ID}} \leftarrow \mathsf{PKDer}_{\mathsf{H}}(\mathsf{msk}, \mathsf{mpk}, \mathsf{St}, \mathsf{ID})$
$(Y^{\mathsf{ID}}, y^{\mathsf{ID}}) := \mathsf{RDer}(Y_m, y_m, 0, \mathsf{ID})$
$\mathtt{HK}[\mathsf{ID}] = (sk_{\mathsf{h}}^{\mathsf{ID}}, pk_{\mathsf{h}}^{\mathsf{ID}}, y^{\mathsf{ID}}, Y^{\mathsf{ID}})$
Return $pk_{\mathsf{h}}^{\mathsf{ID}}$

Oracle $\mathtt{NHKeyO}(\mathsf{ID})$
If $\mathtt{HK}[\mathsf{ID}] \neq \bot$: return $\bot$
If $\mathtt{NHK}[\mathsf{ID}] \neq \bot$:
    $(\cdot, pk_{\mathsf{nh}}^{\mathsf{ID}}, \cdot, \cdot, \cdot) := \mathtt{NHK}[\mathsf{ID}]$
    return $pk_{\mathsf{nh}}^{\mathsf{ID}}$
$sk_{\mathsf{nh},\mathsf{ID}} \leftarrow \mathsf{SKDer}_{\mathsf{NH}}(\mathsf{msk}, \mathsf{mpk}, \mathsf{St}, \mathsf{ID})$
$pk_{\mathsf{nh}}^{\mathsf{ID}} \leftarrow \mathsf{PKDer}_{\mathsf{NH}}(\mathsf{mpk}, \mathsf{St}, \mathsf{ID})$
$(Y^{\mathsf{ID}} y^{\mathsf{ID}}) := \mathsf{RDer}(Y_m, y_m, 0, \mathsf{ID})$
$\mathtt{NHK}[\mathsf{ID}] = (sk_{\mathsf{nh}}^{\mathsf{ID}}, pk_{\mathsf{nh}}^{\mathsf{ID}}, y^{\mathsf{ID}}, Y^{\mathsf{ID}})$
Return $(pk_{\mathsf{nh}}^{\mathsf{ID}}, Y^{\mathsf{ID}})$

Oracle $\mathtt{LeakO}(\mathsf{ID})$
If $\mathtt{HK}[\mathsf{ID}] = \bot$: return $\bot$
$(sk_{\mathsf{h}}^{\mathsf{ID}}, pk_{\mathsf{h}}^{\mathsf{ID}}, y^{\mathsf{ID}}, Y^{\mathsf{ID}}) := \mathtt{HK}[\mathsf{ID}]$
$\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathsf{ID}\}$
Return $(sk_{\mathsf{h}}^{\mathsf{ID}}, y^{\mathsf{ID}})$

Oracle $\mathtt{StLeakO}$
$\mathtt{stLeak} = 1$
Return $\mathsf{St}$
```

```
Oracle $\mathtt{SignO}(m, \mathsf{ID})$
If $\mathtt{HK}[\mathsf{ID}] = \bot \wedge \mathtt{NHK}[\mathsf{ID}] = \bot$: return $\bot$
If $\mathtt{HK}[\mathsf{ID}] = \bot$:
    $(sk, \cdot, \cdot, \cdot, \cdot) := \mathtt{NHK}[\mathsf{ID}]$
Else $(sk, \cdot, \cdot, \cdot, \cdot) := \mathtt{HK}[\mathsf{ID}]$
$\sigma \leftarrow \mathsf{Sign}(sk, m)$
$\mathcal{Q}[\mathsf{ID}] := \mathcal{Q}[\mathsf{ID}] \cup \{m\}$
Return $\sigma$

Oracle $\mathtt{PreSignO}(m, \mathsf{ID}, Y)$
If $\mathtt{HK}[\mathsf{ID}] = \bot \wedge \mathtt{NHK}[\mathsf{ID}] = \bot$: return $\bot$
If $\mathtt{HK}[\mathsf{ID}] = \bot$:
    $(sk, \cdot, \cdot, \cdot, \cdot) := \mathtt{NHK}[\mathsf{ID}]$
Else $(sk, \cdot, \cdot, \cdot, \cdot) := \mathtt{HK}[\mathsf{ID}]$
$\tilde{\sigma} \leftarrow \mathsf{pSign}(sk, m, Y)$
$\mathcal{Q}[\mathsf{ID}] := \mathcal{Q}[\mathsf{ID}] \cup \{m\}$
Return $\tilde{\sigma}$

Oracle $\mathtt{HardRelO}(\mathsf{ID}, \mathsf{ctr})$
If $\mathsf{ctr} \in \mathcal{W}[\mathsf{ID}]$: return $\bot$
If $\mathtt{HK}[\mathsf{ID}] = \bot \wedge \mathtt{NHK}[\mathsf{ID}] = \bot$: return $\bot$
If $\mathtt{HK}[\mathsf{ID}] \neq \bot$: $(\cdot, \cdot, \cdot, y^{\mathsf{ID}}, Y^{\mathsf{ID}}) := \mathtt{HK}[\mathsf{ID}]$
Else: $(\cdot, \cdot, \cdot, y^{\mathsf{ID}}, Y^{\mathsf{ID}}) := \mathtt{NHK}[\mathsf{ID}]$
$(Y_{\mathsf{ctr}}^{\mathsf{ID}}, y_{\mathsf{ctr}}^{\mathsf{ID}}) := \mathsf{RDer}(Y^{\mathsf{ID}}, y^{\mathsf{ID}}, \mathsf{ctr}, \mathsf{ID})$
$\mathcal{W}[\mathsf{ID}] \leftarrow \mathcal{W}[\mathsf{ID}] \cup \{\mathsf{ctr}\}$
Return $(Y_{\mathsf{ctr}}^{\mathsf{ID}}, y_{\mathsf{ctr}}^{\mathsf{ID}})$
```

Fig. 7. Oracles for the security games in Figures 8 and 9.

Wallet Unforgeability In a nutshell, unforgeability for adaptor wallets guarantees that an adversary cannot forge a signature of any uncorrupted wallet instance even if the adversary receives a pre-signature on a fresh message of its choice for a fresh yet deterministically generated statement/witness pair. Recall that in our model, only hardened secret keys can be corrupted.

More concretely, in the unforgeability game, the challenger generates a master key pair, a state and a master statement/witness pair via the execution of $\mathsf{Setup}(1^n)$. The adversary receives the master public key, the master statement and the state as input and obtains access to the oracles described in Fig. 7. Eventually, the adversary outputs an identifier ID^* and a counter ctr^*. If a hardened or non-hardened wallet exists for this ID^* and no statement/witness pair was returned to the adversary for the given ID^* and ctr^*, the challenger derives a new statement/witness pair using ID^* and ctr^* and returns the derived statement to the adversary. Note that the adversary does not receive the witness corresponding to this statement and cannot query it from the `HardRelO` oracle. At this point the adversary provides a message for which the challenger generates and returns a pre-signature using the derived statement. Finally, the adversary outputs a forgery. It wins the game if (1) the forgery is valid, (2) the message has not been queried to the (pre-) signing oracles `PreSignO` and `SignO` for this specific ID^* before, and (3) the session secret key of this wallet has not been leaked to the adversary.

Definition 14. *An adaptor wallet scheme* ADWal_R *is wallet unforgeable if for every PPT adversary* $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ *there exists a negligible function* ν *in the security parameter* n *such that* $\Pr[\mathbf{wUfcma}_{\mathcal{A},\mathsf{ADWal}_R}(n) = 1] \leq \nu(n)$*, where the experiment* $\mathbf{wUfcma}_{\mathcal{A},\mathsf{ADWal}_R}$ *is defined in Fig. 8.*

Wallet Witness Extractability Witness extracatbility for adaptor wallets is similar to the unforgeability notion with the main difference that the adversary chooses the challenge statement/witness pair itself. The adversary then receives a pre-signature on the chosen statement and message and its goal is to output a full signature such that given the pre-signature and full signature, no valid witness for the challenge statement can be extracted.

Definition 15. *An adaptor wallet scheme* ADWal_R *is wallet witness extractable if for every PPT adversary* $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2)$ *there exists a negligible function* ν *in the security parameter* n *such that* $\Pr[\mathbf{wWitExt}_{\mathcal{A},\mathsf{ADWal}_R}(n) = 1] \leq \nu(n)$*, where the experiment* $\mathbf{wWitExt}_{\mathcal{A},\mathsf{ADWal}_R}$ *is defined in Fig. 8.*

Wallet Unlinkability At a high level, the notion of wallet unlinkability describes that an adversary, upon receiving the master public key mpk of an adaptor wallet scheme, cannot distinguish whether a session public key pk^{ID^*} (hardened or non-hardened) was derived from mpk or from a freshly generated mpk'. This should hold, even if the adversary learns all previously generated session public keys of the adaptor wallet scheme and if it corrupts some or all hardened wallets.

The game is formally described in Fig. 9. In its essence, the game allows the adversary, on input the master public key and the master statement, to query the oracles of Fig. 7. Eventually, the adversary outputs an ID^* and a value c, which indicate for which child wallet the adversary wishes to be challenged and if the wallet should be hardened or non-hardened. The game then uniformly at random chooses a bit b and proceeds as follows depending on b. If $b = 0$, the game derives the challenge session public key from mpk corresponding to the value of c. Otherwise, if $b = 1$ the game chooses a fresh master public key mpk' and state St' and derives the challenge session public key from mpk'. Upon receiving the challenge session public key, the adversary has to decide whether $b = 0$ or $b = 1$.

Definition 16. *An adaptor wallet scheme* ADWal_R *is wallet unlinkable if for every PPT adversary* $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2)$ *there exists a negligible function* ν *in the security parameter* n *such that* $\Pr[\mathbf{wUnl}_{\mathcal{A},\mathsf{ADWal}_R}(n) = 1] \leq \nu(n)$*, where the experiment* $\mathbf{wUnl}_{\mathcal{A},\mathsf{ADWal}_R}$ *is defined in Fig. 9.*

```
$\mathbf{wUfcma}_{\mathcal{A},\mathsf{ADWal}_R}(n)$
00 $(\mathsf{msk}, \mathsf{mpk}, \mathsf{St}, Y_m, y_m) \leftarrow \mathsf{Setup}(1^n)$
01 $(\mathsf{ID}^*, \mathsf{ctr}^*, \mathsf{st}_1) \leftarrow \mathcal{A}_1^{\mathcal{O}}(\mathsf{mpk}, \mathsf{St}, Y_m)$
02 If $\mathsf{ctr}^* \in \mathcal{W}[\mathsf{ID}^*]$: return 0
03 If $\mathtt{HK}[\mathsf{ID}] \neq \bot$: $(sk, pk, y^{\mathsf{ID}^*}, Y^{\mathsf{ID}^*}) := \mathtt{HK}[\mathsf{ID}^*]$
04 If $\mathtt{NHK}[\mathsf{ID}] \neq \bot$: $(sk, pk, y^{\mathsf{ID}^*}, Y^{\mathsf{ID}^*}) := \mathtt{NHK}[\mathsf{ID}^*]$
05 Else: return 0
06 $\mathcal{W}[\mathsf{ID}^*] \leftarrow \mathcal{W}[\mathsf{ID}^*] \cup \mathsf{ctr}^*$
07 $(Y^{\mathsf{ID}^*}_{\mathsf{ctr}^*}, y^{\mathsf{ID}^*}_{\mathsf{ctr}^*}) \leftarrow \mathsf{RDer}(Y^{\mathsf{ID}^*}, y^{\mathsf{ID}^*}, \mathsf{ctr}^*, \mathsf{ID}^*)$
08 $(m^*, \mathsf{st}_2) \leftarrow \mathcal{A}_2^{\mathcal{O}}(\mathsf{st}_1, Y^{\mathsf{ID}^*}_{\mathsf{ctr}^*})$
09 $\tilde{\sigma} \leftarrow \mathsf{pSign}(sk, m, Y^{\mathsf{ID}^*}_{\mathsf{ctr}^*})$
10 $\sigma^* \leftarrow \mathcal{A}_3^{\mathcal{O}}(\tilde{\sigma}, \mathsf{st}_2)$
11 Return $(m^* \notin \mathcal{Q}[\mathsf{ID}^*] \wedge \mathsf{Verify}(pk, m^*, \sigma^*) \wedge$
           $\mathsf{ID}^* \notin \mathcal{C})$
```

```
$\mathbf{wWitExt}_{\mathcal{A},\mathsf{ADWal}_R}(n)$
00 $(\mathsf{msk}, \mathsf{mpk}, \mathsf{St}, Y_m, y_m) \leftarrow \mathsf{Setup}(1^n)$
01 $(m^*, \mathsf{ID}^*, Y^*, \mathsf{st}) \leftarrow \mathcal{A}_1^{\mathcal{O}}(\mathsf{mpk}, \mathsf{St}, Y_m)$
02 If $\mathtt{HK}[\mathsf{ID}^*] \neq \bot$:
       $(sk, pk, y^{\mathsf{ID}^*}, Y^{\mathsf{ID}^*}) := \mathtt{HK}[\mathsf{ID}^*]$
03 If $\mathtt{NHK}[\mathsf{ID}^*] \neq \bot$:
       $(sk, pk, y^{\mathsf{ID}^*}, Y^{\mathsf{ID}^*}) := \mathtt{NHK}[\mathsf{ID}^*]$
04 Else: return 0
05 $\tilde{\sigma} \leftarrow \mathsf{pSign}(sk, m^*, Y^*)$
06 $\sigma^* \leftarrow \mathcal{A}_2^{\mathcal{O}}(\tilde{\sigma}, \mathsf{st})$
07 $b_1 \leftarrow (Y^*, \mathsf{Ext}(\sigma^*, \tilde{\sigma}, Y^*)) \notin R$
08 $b_2 \leftarrow m^* \notin \mathcal{Q}[\mathsf{ID}^*]$
09 $b_3 \leftarrow \mathsf{Verify}(pk, m^*, \sigma^*)$
10 $b_4 \leftarrow \mathsf{ID}^* \notin \mathcal{C}$
11 Return $(b_1 \wedge b_2 \wedge b_3 \wedge b_4)$
```

Fig. 8. Wallet unforgeability (**wUfcma**) and wallet witness extractability (**wWitExt**) games for an adaptor scheme. In the above games we have $\mathcal{O} := \{\mathtt{HKeyO}, \mathtt{NHKeyO}, \mathtt{LeakO}, \mathtt{SignO}, \mathtt{PreSignO}, \mathtt{HardRelO}\}$.

```
$\mathbf{wUnl}_{\mathcal{A},\mathsf{ADWal}_R}(n)$
00 $\mathtt{stLeak} = 0$
01 $(\mathsf{msk}, \mathsf{mpk}, \mathsf{St}, Y_m, y_m) \leftarrow \mathsf{Setup}(1^n)$
02 $(\mathsf{ID}^*, c, \mathsf{st}) \leftarrow \mathcal{A}_1^{\mathcal{O}}(\mathsf{mpk}, Y_m)$
03 If $\mathtt{HK}[\mathsf{ID}^*] \neq \bot \vee \mathtt{NHK}[\mathsf{ID}^*] \neq \bot$: return 0
04 $(\mathsf{msk}', \mathsf{mpk}', \mathsf{St}', Y_m', y_m') \leftarrow \mathsf{Setup}(1^n)$
05 $b \leftarrow_{\$} \{0, 1\}$
06 If $b = 0 \wedge c = \mathsf{h}$:
07    $pk_c^{\mathsf{ID}^*} \leftarrow \mathtt{HKeyO}(\mathsf{ID}^*)$
08 If $b = 0 \wedge c = \mathsf{nh}$:
09    $pk_c^{\mathsf{ID}^*} \leftarrow \mathtt{NHKeyO}(\mathsf{ID}^*)$
10 If $b = 1 \wedge c = \mathsf{h}$:
      $pk_c^{\mathsf{ID}^*} \leftarrow \mathtt{HKeyO}(\mathsf{ID}^*, \mathsf{msk}', \mathsf{mpk}', \mathsf{St}', Y_m', y_m')$
11 If $b = 1 \wedge c = \mathsf{nh}$:
      $pk_c^{\mathsf{ID}^*} \leftarrow \mathtt{NHKeyO}(\mathsf{ID}^*, \mathsf{msk}', \mathsf{mpk}', \mathsf{St}', Y_m', y_m')$
12 $b' \leftarrow \mathcal{A}_2^{\mathcal{O}}(pk_c^{\mathsf{ID}^*}, \mathsf{st})$
13 If $c = \mathsf{nh} \wedge \mathtt{stLeak} = 1$:
14    return 0
15 Return $(b = b' \wedge \mathsf{ID}^* \notin \mathcal{C})$
```

```
$\mathbf{wAdapt}_{\mathcal{A},\mathsf{ADWal}_R}(n)$
00 $(\mathsf{msk}, \mathsf{mpk}, \mathsf{St}, Y_m, y_m) \leftarrow \mathsf{Setup}(1^n)$
01 $(pk_{\mathcal{A}}, m^*, \tilde{\sigma}^*, \mathsf{ID}^*, \mathsf{ctr}^*)$
       $\leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{msk}, \mathsf{mpk}, \mathsf{St}, Y_m, y_m)$
02 If $\mathtt{HK}[\mathsf{ID}^*] \neq \bot$:
       $(sk, pk, y^{\mathsf{ID}^*}, Y^{\mathsf{ID}^*}) := \mathtt{HK}[\mathsf{ID}^*]$
03 If $\mathtt{NHK}[\mathsf{ID}^*] \neq \bot$:
       $(sk, pk, y^{\mathsf{ID}^*}, Y^{\mathsf{ID}^*}) := \mathtt{NHK}[\mathsf{ID}^*]$
04 Else: return 0
05 $(Y^{\mathsf{ID}^*}_{\mathsf{ctr}^*}, y^{\mathsf{ID}^*}_{\mathsf{ctr}^*}) := \mathsf{RDer}(Y^{\mathsf{ID}^*}, y^{\mathsf{ID}^*}, \mathsf{ctr}^*, \mathsf{ID}^*)$
06 $\sigma^* \leftarrow \mathsf{Adapt}(\tilde{\sigma}^*, y^{\mathsf{ID}^*}_{\mathsf{ctr}^*})$
07 $b_1 \leftarrow pk_{\mathcal{A}} \in PK$
08 $b_2 \leftarrow \mathsf{pVrfy}(pk_{\mathcal{A}}, m^*, Y^{\mathsf{ID}^*}_{\mathsf{ctr}^*}, \tilde{\sigma}^*)$
09 $b_3 \leftarrow \mathsf{Verify}(pk_{\mathcal{A}}, m^*, \sigma^*)$
10 Return $b_1 \wedge b_2 \wedge \neg b_3$
```

Fig. 9. Wallet unlinkability (**wUnl**) and wallet signature adaptability (**wAdapt**) games for a adaptor scheme. In the above games we have $\mathcal{O} := \{\mathtt{HKeyO}, \mathtt{NHKeyO}, \mathtt{LeakO}, \mathtt{SignO}, \mathtt{PreSignO}, \mathtt{HardRelO}, \mathtt{StLeakO}\}$
and we denote by $\mathtt{NHKeyO}(\mathsf{ID}, \mathsf{msk}', \mathsf{mpk}', \mathsf{St}', Y_m', y_m')$ (and for $\mathtt{HKeyO}$ respectively) the execution of oracle $\mathtt{NHKeyO}$ w.r.t. the values $(\mathsf{msk}', \mathsf{mpk}', \mathsf{St}', Y_m', y_m')$ instead of $(\mathsf{msk}, \mathsf{mpk}, \mathsf{St}, Y_m, y_m)$.

Wallet Signature Adaptability The final property of an adaptor wallet scheme, wallet signature adaptability, intuitively says that an adversary cannot produce a pre-signature for a statement under a self-chosen key pair, such that the pre-signature verifies but the adapted full signature does not verify. The formal game is depicted in Fig. 9 and proceeds as follows: The game initiates an adaptor wallet scheme and gives the entire internal state of the master wallet to the adversary, which is allowed to make queries to oracles as described in Fig. 7. Eventually, the adversary outputs its own public key $pk_{\mathcal{A}}$, a message m^*, a pre-signature $\tilde{\sigma}^*$, an ID^* and a counter ctr^*, upon which the game derives the statement/witness pair corresponding to ID^* and ctr^*. The adversary wins the game if the pre-signature $\tilde{\sigma}^*$ verifies w.r.t. to m^* and the derived statement, but the adapted full signature (adapted with the derived witness) does not verify.

Definition 17. *An adaptor wallet scheme* ADWal_R *is wallet signature adaptable if for every PPT adversary* $\mathcal{A}$ *there exists a negligible function* ν *in the security parameter* n *such that* $\Pr[\mathbf{wAdapt}_{\mathcal{A},\mathsf{ADWal}_R}(n) = 1] \leq \nu(n)$*, where the experiment* $\mathbf{wAdapt}_{\mathcal{A},\mathsf{ADWal}_R}$ *is defined in Fig. 9.*

Definition 18. *An adaptor wallet scheme* ADWal_R *is secure, if satisfies wallet unforgeability, wallet witness extractability, wallet unlinkability and wallet signature adaptability.*

4.2 Construction

We now provide our generic construction of adaptor wallets, from an adaptor signature scheme with rerandomizable keys $\mathsf{RASig}_{R,\Sigma} = (\mathsf{RandSK}, \mathsf{RandPK}, \mathsf{pSign}, \mathsf{Adapt}, \mathsf{pVrfy}, \mathsf{Ext})$. This construction uses a hash function $\mathsf{H} : \{0,1\}^* \to X$ and we require that the hard relation R is witness rerandomizable as per Def. 2. Our construction can be found in Figure 10.

Theorem 3. *Let* $\mathsf{RASig}_{R,\Sigma}$ *be an adaptor signature scheme with rerandomizable keys as per Def. 8, let* R *be a witness rerandomizable hard relation as per Def. 2, and let* $\mathsf{H} : \{0,1\}^* \to X$ *be a hash function modeled as random oracle. Then the construction from Fig. 10 is a secure adaptor wallet scheme.*

In order to prove this theorem, we have to show that the construction from Fig. 10 satisfies wallet unforgeability, wallet witness extractability, wallet unlinkability and wallet signature adaptability. We provide security arguments for these properties in Appx. A.

4.3 Impossibility of Independent Statement/Witness Derivation

As mentioned above, one main question that arises when modeling derivation of statement/witness pairs in a deterministic fashion is whether an independent derivation of statement/witness pairs in hot and cold wallets respectively is possible. Surprisingly, unlike the secret and public key derivation mechanism, we show that this is not necessarily the case. At a high level, this is because unlike

```
Algorithm $\mathsf{Setup}(1^n)$
  $\mathsf{St} \leftarrow_\$ \{0,1\}^n$
  $(Y_m, y_m) \leftarrow R.\mathsf{GenR}(1^n)$
  $(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{RASig}_{R,\Sigma}.\mathsf{Gen}(1^n)$
  Return $(\mathsf{msk}, \mathsf{mpk}, \mathsf{St}, Y_m, y_m)$

Algorithm $\mathsf{pSign}(sk^{\mathsf{ID}}, m, Y)$
  $\tilde{\sigma} \leftarrow \mathsf{RASig}_{R,\Sigma}.\mathsf{pSign}(sk^{\mathsf{ID}}, m, Y)$
  Return $\tilde{\sigma}$

Algorithm $\mathsf{pVrfy}(pk^{\mathsf{ID}}, m, Y, \tilde{\sigma})$
  Return $\mathsf{RASig}_{R,\Sigma}.\mathsf{pVrfy}(pk^{\mathsf{ID}}, m, Y, \tilde{\sigma})$

Algorithm $\mathsf{Adapt}(\tilde{\sigma}, y^{\mathsf{ID}}_{\mathsf{ctr}})$
  $\sigma \leftarrow \mathsf{RASig}_{R,\Sigma}.\mathsf{Adapt}(\tilde{\sigma}, y^{\mathsf{ID}}_{\mathsf{ctr}})$
  Return $\sigma$

Algorithm $\mathsf{Ext}(\sigma, \tilde{\sigma}, Y^{\mathsf{ID}}_{\mathsf{ctr}})$
  Return $\mathsf{RASig}_{R,\Sigma}.\mathsf{Ext}(\sigma, \tilde{\sigma}, Y^{\mathsf{ID}}_{\mathsf{ctr}})$

Algorithm $\mathsf{Sign}(sk^{\mathsf{ID}}, m)$
  $\sigma \leftarrow \mathsf{RASig}_{R,\Sigma}.\mathsf{Sign}(sk^{\mathsf{ID}}, m)$
  Return $\sigma$

Algorithm $\mathsf{Verify}(pk^{\mathsf{ID}}, m, \sigma)$
  Return $\mathsf{RASig}_{R,\Sigma}.\mathsf{Verify}(pk^{\mathsf{ID}}, m, \sigma)$

Algorithm $\mathsf{SKDer}_{\mathsf{H}}(\mathsf{msk}, \mathsf{St}, \mathsf{ID})$
  $\rho \leftarrow \mathsf{H}(\mathsf{msk}, \mathsf{St}, \mathsf{ID})$
  $sk^{\mathsf{ID}} \leftarrow \mathsf{RASig}_{R,\Sigma}.\mathsf{RandSK}(\mathsf{msk}, \rho)$
  Return $sk^{\mathsf{ID}}$

Algorithm $\mathsf{SKDer}_{\mathsf{NH}}(\mathsf{msk}, \mathsf{mpk}, \mathsf{St}, \mathsf{ID})$
  $\rho \leftarrow \mathsf{H}(\mathsf{mpk}, \mathsf{St}, \mathsf{ID})$
  $sk^{\mathsf{ID}} \leftarrow \mathsf{RASig}_{R,\Sigma}.\mathsf{RandSK}(\mathsf{msk}, \rho)$
  Return $sk^{\mathsf{ID}}$

Algorithm $\mathsf{PKDer}_{\mathsf{H}}(\mathsf{msk}, \mathsf{mpk}, \mathsf{St}, \mathsf{ID})$
  $\rho \leftarrow \mathsf{H}(\mathsf{msk}, \mathsf{St}, \mathsf{ID})$
  $pk^{\mathsf{ID}} \leftarrow \mathsf{RASig}_{R,\Sigma}.\mathsf{RandPK}(\mathsf{mpk}, \rho)$
  Return $pk^{\mathsf{ID}}$

Algorithm $\mathsf{PKDer}_{\mathsf{NH}}(\mathsf{mpk}, \mathsf{St}, \mathsf{ID})$
  $\rho \leftarrow \mathsf{H}(\mathsf{mpk}, \mathsf{St}, \mathsf{ID})$
  $pk^{\mathsf{ID}} \leftarrow \mathsf{RASig}_{R,\Sigma}.\mathsf{RandPK}(\mathsf{mpk}, \rho)$
  Return $pk^{\mathsf{ID}}$

Algorithm $\mathsf{RDer}(Y, y, \mathsf{ctr}, \mathsf{ID})$
  $\rho \leftarrow \mathsf{H}(y, \mathsf{ctr}, \mathsf{ID})$
  $y^{\mathsf{ID}}_{\mathsf{ctr}} \leftarrow R.\mathsf{RandWit}(y, \rho)$
  $Y^{\mathsf{ID}}_{\mathsf{ctr}} \leftarrow R.\mathsf{WitToSt}(y^{\mathsf{ID}}_{\mathsf{ctr}})$
  Return $(Y^{\mathsf{ID}}_{\mathsf{ctr}}, y^{\mathsf{ID}}_{\mathsf{ctr}})$
```

Fig. 10. Generic construction of adaptor wallets w.r.t. an adaptor signature scheme with rerandomizable keys $\mathsf{RASig}_{R,\Sigma}$, where R is a witness rerandomizable hard relation as per Def. 2 and a hash function $\mathsf{H} : \{0,1\}^* \to X$.

session secret keys, derived witnesses do not remain secret but are typically revealed in adaptor signature applications.

More formally, we say that a hard relation $R \subseteq \mathcal{D}_{\mathsf{Y}} \times \mathcal{D}_{\mathsf{w}}$ has independently rerandomizable statement/witness pairs w.r.t. a randomness space X, if there exist two functions $f_{\mathsf{STDer}} : \mathcal{D}_{\mathsf{Y}} \times X \to \mathcal{D}_{\mathsf{Y}}$ and $f_{\mathsf{WitDer}} : \mathcal{D}_{\mathsf{w}} \times X \to \mathcal{D}_{\mathsf{w}}$ where for any $\rho \in X$ and any $(Y, y) \in R$ we have: $Y' \leftarrow f_{\mathsf{STDer}}(Y, \rho)$, $y' \leftarrow f_{\mathsf{WitDer}}(y, \rho)$, and $(Y', y') \in R$.

Translating the above to the hot/cold wallet setting, means that the cold wallet executes function f_{WitDer} and the hot wallet function f_{STDer} where ρ is computed as $\rho \leftarrow \omega(Y, \mathsf{ctr}, \mathsf{ID})$ for some deterministic and publicly known function $\omega(\cdot)$ which is typically instantiated with a hash function and modeled as a random oracle. An adversary in this setting can corrupt the hot wallet but not the cold wallet, and hence can learn the statements Y and Y' as well as the respective randomness ρ. In addition, as required by certain adaptor signature applications, the adversary eventually learns a derived witness $y' \leftarrow f_{\mathsf{WitDer}}(y, \rho)$. Therefore, if there exists a function $f^{-1} : \mathcal{D}_{\mathsf{w}} \times X \to \mathcal{D}_{\mathsf{w}}$ which on input y' and ρ returns y, i.e., $y \leftarrow f^{-1}(y', \rho)$, then we cannot construct deterministic and independent statement/witness derivation from f_{WitDer} and f_{STDer}. This is, because an adversary could compute y and thereby break unforgeability of the adaptor wallet scheme.

For the following (informal) theorem, we introduce the notion of an *adaptor wallet with independent statement/witness derivation.* This notion differs from adaptor wallets in the sense that statement/witness pairs are derived via functions f_{STDer} and f_{WitDer} respectively (instead of RDer) and randomness is derived via ω for a randomness space X as described above.

Theorem 4 (informal). *Let* AW *be an adaptor wallet scheme with independent statement/witness derivation and assume that the function* f^{-1} *as described above exists. Then* AW *does not satisfy wallet unforgability.*

Proof (Sketch). We can prove this theorem by exhibiting an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ that breaks the wallet unforgeability of AW. The attack proceeds as follows:

In the wallet unforgeability game, $\mathcal{A}_1$ chooses an ID^* and queries oracle $\mathtt{NHKeyO}$ on input ID^* and receives $(pk_{\mathsf{nh}}^{\mathsf{ID}^*}, Y^{\mathsf{ID}^*})$. It then queries oracle $\mathtt{HardRelO}$ on input ID^* and a counter ctr and receives $(Y_{\mathsf{ctr}}^{\mathsf{ID}^*}, y_{\mathsf{ctr}}^{\mathsf{ID}^*})$. Note that, since $\mathcal{A}_1$ knows Y^{ID^*}, it can compute the randomness $\rho_{\mathsf{ctr}}^{\mathsf{ID}^*} \leftarrow \omega(Y^{\mathsf{ID}^*}, \mathsf{ctr}, \mathsf{ID}^*)$ that was used for the derivation of $(Y_{\mathsf{ctr}}^{\mathsf{ID}^*}, y_{\mathsf{ctr}}^{\mathsf{ID}^*})$. Finally, $\mathcal{A}_1$ executes $y^{\mathsf{ID}^*} \leftarrow f^{-1}(y_{\mathsf{ctr}}^{\mathsf{ID}^*}, \rho_{\mathsf{ctr}}^{\mathsf{ID}^*})$, which allows the adversary to run f_{WitDer} on input y^{ID^*} and any randomness ρ. $\mathcal{A}_1$ now outputs $(\mathsf{ID}^*, \mathsf{ctr}^*, \mathsf{st}_1)$ where $\mathsf{st}_1 := (y^{\mathsf{ID}^*}, \mathsf{ID}^*, \mathsf{ctr}^*)$. Upon receiving $Y_{\mathsf{ctr}^*}^*$, $\mathcal{A}_2$ chooses any message m^* from the message space and outputs (m^*, st_2) where $\mathsf{st}_2 := \mathsf{st}_1$. Upon receiving $\tilde{\sigma}$, $\mathcal{A}_3$, executes $y_{\mathsf{ctr}^*}^{\mathsf{ID}^*} \leftarrow f_{\mathsf{WitDer}}(y^{\mathsf{ID}^*}, \rho_{\mathsf{ctr}^*}^{\mathsf{ID}^*})$, for $\rho_{\mathsf{ctr}^*}^{\mathsf{ID}^*} \leftarrow \omega(Y^{\mathsf{ID}^*}, \mathsf{ctr}^*, \mathsf{ID}^*)$. Finally, $\mathcal{A}_3$ executes $\mathsf{Adapt}(pk, \tilde{\sigma}, y_{\mathsf{ctr}^*}^{\mathsf{ID}^*})$ and receives the full signature σ^* which constitutes a valid forgery.

Let us now see how this result affects existing adaptor signature constructions. For the ECDSA-based adaptor signature construction $\mathsf{EC}_{R_g,\mathsf{PEC}}[\mathsf{H}]$ as described in Sec. 2.4 it is not possible to define f_{STDer} without providing the witness as input. This is mainly because the hard relation $R_g := \{((Y, \pi), y) \mid Y = g^y \wedge \mathsf{V}_g(Y, \pi) = 1\}$ requires a zero-knowledge proof alongside the statement Y, that proves knowledge of the witness y. Naturally, generating this proof without the witness is not possible. Now consider the "pure" dlog hard relation $R^{dlog} := \{(Y, y) \mid Y = g^y\}$, which is required for adaptor signature schemes based on Schnorr and Katz-Wang [9]. The statement/witness pairs for this relation can be rerandomized either multiplicatively or additively. Both of these operations, however, can easily be inverted. For instance, for a statement/witness pair $(g^y, y) \in R^{dlog}$, an additive rerandomization would instantiate the functions f_{STDer} and f_{WitDer} as $f_{\mathsf{STDer}}(g^y, \rho) := g^y \cdot g^\rho = Y'$ and $f_{\mathsf{WitDer}}(y, \rho) := y + \rho = y'$. Naturally, the function f^{-1} can simply be instantiated as $f^{-1}(y', \rho) := y' - \rho = y$.

Impact of the impossibility result. Due to the above impossibility result of independent statement/witness derivation we cannot construct an adaptor wallet scheme with statement derivation in the hot wallet. However, for certain applications of adaptor signatures, this restriction is tolerable as the cold wallet does not need to generate many signatures and/or statement/witness pairs and therefore does not need to be activated frequently. Further, in practice one can minimize the number of times a cold wallet must be activated by batching the generation of statement/witness pairs, i.e., the cold wallet can generate multiple pairs and send all statements at once to the hot wallet. For other applications with frequent transactions, such as payment channels, an adaptor wallet user can use a hardened wallet as explained in Sec. 4.1.

An interesting open problem is to design an adaptor wallet scheme that overcomes this impossibility result.

Acknowledgments

This work is supported by the German Research Foundation DFG - SFB 1119 - 236615297 (CROSSING Project S7), by the German Federal Ministry of Education and Research (BMBF) *iBlockchain Project* (grant nr. 16KIS0902), by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts within their joint support of the *National Research Center for Applied Cybersecurity ATHENE.*

References

[1] N. Alkeilani Alkadri et al. "Deterministic Wallets in a Quantum World". In: *ACM CCS 2020*. Ed. by J. Ligatti et al. ACM Press, Nov. 2020, pp. 1017–1031. DOI: 10.1145/3372297.3423361.

[2] M. J. Atallah et al. "Dynamic and Efficient Key Management for Access Hierarchies". In: *ACM Trans. Inf. Syst. Secur.* 12.3 (Jan. 2009). ISSN: 1094-9224. DOI: 10.1145/1455526.1455531. URL: https://doi.org/10.1145/1455526.1455531.

[3] L. Aumayr et al. "Generalized Channels from Limited Blockchain Scripts and Adaptor Signatures". In: *Advances in Cryptology – ASIACRYPT 2021*. Ed. by M. Tibouchi and H. Wang. Cham: Springer International Publishing, 2021, pp. 635–664.

[4] M. Blum et al. "Non-Interactive Zero-Knowledge and Its Applications". In: *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*. STOC '88. Chicago, Illinois, USA: Association for Computing Machinery, 1988, 103–112. ISBN: 0897912640. DOI: 10.1145/62212.62222. URL: https://doi.org/10.1145/62212.62222.

[5] P. Das et al. "A Formal Treatment of Deterministic Wallets". In: *ACM CCS 2019*. Ed. by L. Cavallaro et al. ACM Press, Nov. 2019, pp. 651–668. DOI: 10.1145/3319535.3354236.

[6] P. Das et al. "The Exact Security of BIP32 Wallets". In: *ACM CCS 2021*. Ed. by G. Vigna and E. Shi. ACM Press, Nov. 2021, pp. 1020–1042. DOI: 10.1145/3460120.3484807.

[7] A. Deshpande and M. Herlihy. "Privacy-Preserving Cross-Chain Atomic Swaps". In: *FC 2020*. Ed. by M. Bernhard et al. Springer International Publishing, 2020.

[8] A. Erwig and S. Riahi. “Deterministic Wallets for Adaptor Signatures”. In: *Computer Security – ESORICS 2022*. Ed. by V. Atluri et al. Cham: Springer Nature Switzerland, 2022, pp. 487–506. ISBN: 978-3-031-17146-8. DOI: `https://doi.org/10.1007/978-3-031-17146-8_24`.

[9] A. Erwig et al. “Two-Party Adaptor Signatures from Identification Schemes”. In: *PKC 2021, Part I*. Ed. by J. Garay. Vol. 12710. LNCS. Springer, Heidelberg, May 2021, pp. 451–480. DOI: `10.1007/978-3-030-75245-3_17`.

[10] M. F. Esgin et al. “Post-Quantum Adaptor Signatures and Payment Channel Networks”. In: *ESORICS 2020, Part II*. Ed. by L. Chen et al. Vol. 12309. LNCS. Springer, Heidelberg, Sept. 2020, pp. 378–397. DOI: `10.1007/978-3-030-59013-0_19`.

[11] M. Fersch et al. “On the One-Per-Message Unforgeability of (EC)DSA and Its Variants”. In: *TCC 2017, Part II*. Ed. by Y. Kalai and L. Reyzin. Vol. 10678. LNCS. Springer, Heidelberg, Nov. 2017, pp. 519–534. DOI: `10.1007/978-3-319-70503-3_17`.

[12] N. Fleischhacker et al. “Efficient Unlinkable Sanitizable Signatures from Signatures with Re-randomizable Keys”. In: *PKC 2016, Part I*. Ed. by C.-M. Cheng et al. Vol. 9614. LNCS. Springer, Heidelberg, Mar. 2016, pp. 301–330. DOI: `10.1007/978-3-662-49384-7_12`.

[13] G. Gutoski and D. Stebila. “Hierarchical Deterministic Bitcoin Wallets that Tolerate Key Leakage”. In: *FC 2015*. Ed. by R. Böhme and T. Okamoto. Vol. 8975. LNCS. Springer, Heidelberg, Jan. 2015, pp. 497–504. DOI: `10.1007/978-3-662-47854-7_31`.

[14] J. Katz and N. Wang. “Efficiency Improvements for Signature Schemes with Tight Security Reductions”. In: *ACM CCS 2003*. Ed. by S. Jajodia et al. ACM Press, Oct. 2003, pp. 155–164. DOI: `10.1145/948109.948132`.

[15] Y. Kondi et al. “Refresh When You Wake Up: Proactive Threshold Wallets with Offline Devices”. In: *2021 IEEE Symposium on Security and Privacy (SP)*. 2021, pp. 608–625. DOI: `10.1109/SP40001.2021.00067`.

[16] A. D. Luzio et al. “Arcula: A Secure Hierarchical Deterministic Wallet for Multi-asset Blockchains”. In: *CANS 20*. Ed. by S. Krenn et al. Vol. 12579. LNCS. Springer, Heidelberg, Dec. 2020, pp. 323–343. DOI: `10.1007/978-3-030-65411-5_16`.

[17] V. Madathil et al. *Practical Decentralized Oracle Contracts for Cryptocurrencies*. Cryptology ePrint Archive, Report 2022/499. `https://ia.cr/2022/499`. 2022.

[18] G. Malavolta et al. “Anonymous Multi-Hop Locks for Blockchain Scalability and Interoperability”. In: *NDSS 2019*. The Internet Society, Feb. 2019.

[19] P. Moreno-Sanchez and A. Kate. *Scriptless Scripts with ECDSA*. `https://lists.linuxfoundation.org/pipermail/lightning-dev/attachments/20180426/fe978423/attachment-0001.pdf`. 2018.

[20] A. Poelstra. *Scriptless scripts*. `https://download.wpsoftware.net/bitcoin/wizardry/mw-slides/2017-03-mit-bitcoin-expo/slides.pdf`. 2017. Visited 10/2020.

[21] E. B. Sasson et al. "Zerocash: Decentralized anonymous payments from bitcoin". In: *2014 IEEE symposium on security and privacy*. IEEE. 2014, pp. 459–474.

[22] C.-P. Schnorr. "Efficient Signature Generation by Smart Cards". In: *Journal of Cryptology* 4.3 (Jan. 1991), pp. 161–174. DOI: 10.1007/BF00196725.

[23] E. Tairi et al. "A^{2}L: Anonymous Atomic Locks for Scalability in Payment Channel Hubs". In: *2021 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2021, pp. 1834–1851. DOI: 10.1109/SP40001.2021.00111.

[24] E. Tairi et al. "Post-Quantum Adaptor Signature for Privacy-Preserving Off-Chain Payments". In: *Financial Cryptography and Data Security*. Ed. by N. Borisov and C. Diaz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2021, pp. 131–150. ISBN: 978-3-662-64331-0.

[25] S. A. Thyagarajan et al. *Universal Atomic Swaps: Secure Exchange of Coins Across All Blockchains*. Cryptology ePrint Archive, Report 2021/1612. https://ia.cr/2021/1612. 2021.

[26] N. Van Saberhagen. "CryptoNote v 2.0". In: (2013).

[27] X. Yin et al. *Secure Hierarchical Deterministic Wallet Supporting Stealth Address*. Cryptology ePrint Archive, Paper 2022/627. https://eprint.iacr.org/2022/627. 2022. URL: https://eprint.iacr.org/2022/627.

A Proof of Thm. 3

A.1 Wallet Unforgeability

Lemma 3. *The adaptor wallet construction from Fig. 10 satisfies wallet unforgeability as per Def. 14.*

Proof (sketch). The proof of this lemma works in a similar way as the wallet unforgability proof of Das et al. [6] for hierarchical deterministic wallets. In their proof, Das et al. provide a reduction to the unforgeability property of a signature scheme with rerandomizable keys. In our case, we reduce the wallet unforgeability of our adaptor wallet construction to the unforgeability of adaptor signatures with rerandomizable keys. Therefore, we will discuss how to adjust the reduction of Das et al. in our setting. At a high level, in addition to their proof, we have to show how the reduction can simulate pre-signing queries and we have to exhibit reductions to the witness rerandomizable hard relation.

The main issue in the proof arises from the fact that the reduction on input only a public key pk from game $\mathsf{aSigForge{-}hrk}$ has to simulate game $\mathbf{wUfcma}_{\mathcal{A},\mathsf{ADWal}_R}$ to adversary $\mathcal{A}$. At the beginning of game $\mathbf{wUfcma}_{\mathcal{A},\mathsf{ADWal}_R}$, the reduction generates its own master statement/witness pair as well as a state, which it can use to derive all required witnesses itself. In order to answer queries to the non-hardened wallet generation oracle NHKeyO, the reduction simply queries a rerandomized public key from game $\mathsf{aSigForge{-}hrk}$ and honestly generates statement/witness pairs itself. The derivation of hardened wallets is more challenging, because $\mathcal{A}$ can query the `LeakO` oracle, upon which the reduction must provide the secret key sk^{ID} and witness y^{ID} of a hardened wallet. As mentioned before, revealing y^{ID} is not a problem, but it is not clear how the reduction can reveal sk^{ID}. Therefore, the reduction simulates the derivation of hardened wallets by sampling fresh key pairs that are independent of pk. This allows the reduction to answer $\mathcal{A}$'s `LeakO` queries, but has the disadvantage that, if $\mathcal{A}$ outputs a forgery for a hardened wallet in game $\mathbf{wUfcma}_{\mathcal{A},\mathsf{ADWal}_R}$, then the reduction cannot use this forgery to win game $\mathsf{aSigForge{-}hrk}$ (since the keys of hardened wallets are independent of pk). For this reason, the reduction guesses the position of one single hardened wallet that the adversary will not corrupt and will potentially choose to generate a forgery for. This wallet is then generated as a non-hardened wallet, such that, should $\mathcal{A}$ provide a forgery for it, the reduction can use the forgery to win game $\mathsf{aSigForge{-}hrk}$. We note that this simulation allows the reduction to answer $\mathcal{A}$'s queries to oracles `PreSignO` and `SignO`, since (1) for hardened wallets the reduction knows the secret key, and (2) for non-hardened wallets the reduction can query the corresponding oracles of game $\mathsf{aSigForge{-}hrk}$ to receive (pre-)signatures. Proving indistinguishability of the above simulation works similarly as in the proof of Das et al.

During the challenge phase of game $\mathbf{wUfcma}_{\mathcal{A},\mathsf{ADWal}_R}$, adversary $\mathcal{A}$ first outputs an identifier ID^* and counter ctr^* and later a message m^*. The reduction then sends the public key pk^* of the wallet with identifier ID^* and the message m^* to game $\mathsf{aSigForge{-}hrk}$ and receives a pre-signature on m^* under public key pk^* for a randomly sampled statement Y. We have to show that the reduction can forward this pre-signature to $\mathcal{A}$ without $\mathcal{A}$ realizing that Y was sampled at random instead of deterministically derived from Y_m. Note that the hard relation is witness rerandomizable and that the randomness ρ for the witness rerandomization is derived as $\rho \leftarrow \mathsf{H}(y^{\mathsf{ID}^*}, \mathsf{ctr}^*, \mathsf{ID}^*)$, where H is modeled as a random oracle. Therefore, the adversary can distinguish statement Y from $Y^{\mathsf{ID}^*}_{\mathsf{ctr}^*}$ only if it queries H on input $(y^{\mathsf{ID}^*}, \mathsf{ctr}^*, \mathsf{ID}^*)$ or $(y_m, \cdot, \cdot)$. However, we can show that these queries happen at most with negligible probability by exhibiting reductions to the witness rerandomizable hard relation. These reductions work similarly as the one described in Claim. 3 of the wallet unlinkability proof with the only difference that we must provide two reductions, i.e., one for each critical random oracle query $(y^{\mathsf{ID}^*}, \mathsf{ctr}^*, \mathsf{ID}^*)$ and $(y_m, \cdot, \cdot)$.

A.2 Wallet Witness Extractability

Lemma 4. *The adaptor wallet construction from Fig. 10 satisfies wallet witness extractability as per Def. 15.*

The proof of this lemma follows in a similar way as the proof of Lemma 3 with the only difference that we do not require the final reductions to the witness rerandomizable hard relation.

A.3 Wallet Unlinkability

Lemma 5. *The adaptor wallet construction from Fig. 10 satisfies wallet unlinkability as per Def. 16.*

The proof of this lemma is similar to the wallet unlinkability proof for hierarchical deterministic wallets of Das et al. [6], however, we have to show additionally that derived statement/witness pairs of the witness rerandomizable hard relation do not help an adversary to win the game. For completeness, we present the entire proof here.

Proof. Consider a PPT adversary $\mathcal{A}$ that plays in the wallet unlinkability game **wUnl**. The game proceeds by first generating a master key pair, a state, and a master statement/witness pair via an execution of the Setup procedure, i.e., $(\mathsf{msk}, \mathsf{mpk}, \mathsf{St}, Y_m, y_m) \leftarrow \mathsf{Setup}(1^n)$. The game then executes $\mathcal{A}$ on input mpk and Y_m and grants access to oracles $\mathcal{O}$. Upon the adversary outputting the tuple $(\mathsf{ID}^*, c, \mathsf{st})$, the game first checks if the (hardened or non-hardened) wallet with identifier ID^* has already been generated, and if so aborts, i.e., the adversary loses the game.

In case $\mathcal{A}$ challenges a non-hardened wallet (i.e., $c = \mathsf{nh}$), the game either derives a public key $pk_{\mathsf{nh}}^{\mathsf{ID}^*} \leftarrow \mathsf{PKDer}_{\mathsf{NH}}(\mathsf{mpk}, \mathsf{St}, \mathsf{ID}^*)$ or it executes the setup procedure $(\mathsf{msk}', \mathsf{mpk}', \mathsf{St}', Y_m', y_m') \leftarrow \mathsf{Setup}(1^n)$ and derives a public key $pk_{\mathsf{nh}}^{\mathsf{ID}^*\prime} \leftarrow \mathsf{PKDer}_{\mathsf{NH}}(\mathsf{mpk}', \mathsf{St}', \mathsf{ID}^*)$. Recall that non-hardened public keys are computed as follows:

$$\begin{aligned} \rho \leftarrow & \mathsf{H}(\mathsf{mpk}, \mathsf{St}, \mathsf{ID}^*), \\ pk_{\mathsf{nh}}^{\mathsf{ID}^*} \leftarrow & \mathsf{RASig}_{R,\Sigma}.\mathsf{RandPK}(\mathsf{mpk}, \rho) \end{aligned}$$

Due to the (perfect) rerandomizability of keys property (cf. Def. 8) of the $\mathsf{RASig}_{R,\Sigma}$ scheme, the public keys $pk_{\mathsf{nh}}^{\mathsf{ID}^*}$ and $pk_{\mathsf{nh}}^{\mathsf{ID}^*\prime}$ are identically distributed from $\mathcal{A}$'s view as long as ρ is uniformly random. This is the case, if $\mathcal{A}$ has not previously queried the random oracle H on input $(\mathsf{mpk}, \mathsf{St}, \mathsf{ID}^*)$. In order to make such a query, $\mathcal{A}$ must correctly guess the state St, which is a uniformly random n-bit string. Since $\mathcal{A}$ makes a polynomial number (in n) of random oracle queries, the probability of guessing St correctly is negligible.

In case $\mathcal{A}$ challenges a hardened wallet (i.e., $c = \mathsf{h}$), the game either derives a public key $pk_{\mathsf{h}}^{\mathsf{ID}^*} \leftarrow \mathsf{PKDer}_{\mathsf{H}}(\mathsf{msk}, \mathsf{mpk}, \mathsf{St}, \mathsf{ID}^*)$ or it executes $(\mathsf{msk}', \mathsf{mpk}', \mathsf{St}', Y_m', y_m') \leftarrow \mathsf{Setup}(1^n)$ and derives a public key $pk_{\mathsf{h}}^{\mathsf{ID}^*\prime} \leftarrow \mathsf{PKDer}_{\mathsf{H}}(\mathsf{msk}', \mathsf{mpk}', \mathsf{St}', \mathsf{ID}^*)$. For hardened wallets, $\mathcal{A}$ is allowed to query the `StLeakO` oracle which returns the state St. Therefore, $\mathcal{A}$ does not need to guess St correctly in this case. Recall that hardened public keys are computed as follows:

$$\begin{aligned} \rho \leftarrow & \mathsf{H}(\mathsf{msk}, \mathsf{mpk}, \mathsf{St}, \mathsf{ID}^*), \\ pk_{\mathsf{h}}^{\mathsf{ID}^*} \leftarrow & \mathsf{RASig}_{R,\Sigma}.\mathsf{RandPK}(\mathsf{mpk}, \rho) \end{aligned}$$

Therefore, due to the (perfect) rerandomizability of keys property, $\mathcal{A}$ can distinguish $pk_{\mathsf{h}}^{\mathsf{ID}^*}$ and $pk_{\mathsf{h}}^{\mathsf{ID}^*\prime}$ only if it is able to compute the master secret key

msk. Let E be the event that $\mathcal{A}$ calls the random oracle on input $(\mathsf{msk}, \mathsf{mpk}, \mathsf{St}, \cdot)$. Then we can state the following claim:

Claim 2 *If* E *happens with more than negligible probability, then there exists a PPT algorithm* $\mathcal{C}$ *that wins game* $\mathsf{aSigForge{-}hrk}_{\mathcal{C},\mathsf{RASig}_{R,\Sigma}}$ *with more than negligible probability.*

Proof. We prove this claim via reduction to the $\mathsf{aSigForge{-}hrk}_{\mathcal{C},\mathsf{RASig}_{R,\Sigma}}$ game. In this reduction, $\mathcal{C}$ first receives a public key pk from game $\mathsf{aSigForge{-}hrk}$ and $\mathcal{C}$ additionally chooses a state $\mathsf{St} \leftarrow_\$ \{0,1\}^n$ and a master statement/witness pair $(Y_m, y_m) \leftarrow R.\mathsf{GenR}(1^n)$. $\mathcal{C}$ then forwards pk and Y_m to $\mathcal{A}$. The generated values St and (Y_m, y_m) allow $\mathcal{C}$ to honestly simulate oracles `NHKeyO`, `HardRelO` and `StLeakO`. The remaining oracles are simulated as follows:

– Random oracle H: $\mathcal{C}$ queries the oracle `Rand` from game $\mathsf{aSigForge{-}hrk}$ to receive randomness ρ and outputs ρ.
– (Pre-) Signing oracles `PreSignO` and `SignO` for non-hardened wallets: On input (m, ID, Y) and (m, ID) respectively, $\mathcal{C}$ calls the corresponding oracles in game $\mathsf{aSigForge{-}hrk}$ using the randomness that corresponds to ID.
– Hardened Child Derivation oracle `HKeyO` and `LeakO` oracle: On input ID, $\mathcal{C}$ generates a fresh key pair $(sk', pk') \leftarrow \mathsf{RASig}_{R,\Sigma}.\mathsf{Gen}(1^n)$ and assigns the tuple $(sk_{\mathsf{h}}^{\mathsf{ID}}, pk_{\mathsf{h}}^{\mathsf{ID}}) := (sk', pk')$. This allows also to simulate (pre-) signing queries for hardened wallets. The `LeakO` oracle on input ID can then be simulated by outputting $sk_{\mathsf{h}}^{\mathsf{ID}}$ and the corresponding witness y^{ID}. This simulation is indistinguishable for $\mathcal{A}$ due to the rerandomizability of keys property of RASig.

The only way to distinguish this simulation from the real game for $\mathcal{A}$ is if event E happens, i.e., if $\mathcal{A}$ correctly guesses the secret key sk corresponding to pk and makes a random oracle query on input sk. In this case, however, $\mathcal{C}$ learns sk and can use it to win its own game.

Finally, we have to show that $\mathcal{A}$ cannot break the wallet unlinkability property by distinguishing the master statement/witness pair (Y'_m, y'_m) from (Y_m, y_m) (in case of $b = 1$). In a nutshell, if $\mathcal{A}$ was able to correctly guess y_m, it can check, whether the challenge wallet with identifier ID^* has been initialized with a statement/witness pair derived from (Y_m, y_m) or from (Y'_m, y'_m). Let E' be the event that $\mathcal{A}$ calls the random oracle on input $(y_m, \cdot, \cdot)$. Then we can state the following claim:

Claim 3 *If* E' *happens with more than negligible probability, then there exists an algorithm* $\mathcal{C}$ *that breaks the hard relation* R *with more than negligible probability.*

Proof. We prove this claim via reduction to the hard relation R. In this reduction, $\mathcal{C}$ first receives a statement Y_m and additionally chooses a state $\mathsf{St} \leftarrow_\$ \{0,1\}^n$ and a master key pair $(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{RASig}_{R,\Sigma}.\mathsf{Gen}(1^n)$. $\mathcal{C}$ then forwards mpk to $\mathcal{A}$. The generated values St and $(\mathsf{msk}, \mathsf{mpk})$ allow $\mathcal{C}$ to honestly simulate oracles `PreSignO`, `SignO` and `StLeakO`. The remaining oracles are simulated as follows:

– (Non-) Hardened Child Derivation oracles $\texttt{NHKeyO}$ and $\texttt{HKeyO}$: On input ID, $\mathcal{C}$ computes the child key pair $(sk^{\mathsf{ID}}, pk^{\mathsf{ID}})$ correctly from $(\mathsf{msk}, \mathsf{mpk})$ and generates a fresh statement/witness pair $(Y', y') \leftarrow R.\mathsf{GenR}(1^n)$ and assigns $(Y^{\mathsf{ID}}, y^{\mathsf{ID}}) := (Y', y')$. This simulation is indistinguishable for $\mathcal{A}$ due to the witness rerandomizability property of R, as long as $\mathcal{A}$ does not correctly guess y_m and query H on input $(y_m, \cdot, \mathsf{ID})$.
– $\texttt{LeakO}$ and $\texttt{HardRelO}$ oracles: Using the freshly generated statement/witness pairs $(Y^{\mathsf{ID}}, y^{\mathsf{ID}})$ during the $\texttt{NHKeyO}$ and $\texttt{HKeyO}$ oracle executions, $\mathcal{C}$ can now correctly simulate the $\texttt{LeakO}$ and $\texttt{HardRelO}$ oracles.

The only way to distinguish this simulation from the real game for $\mathcal{A}$ is if event E' happens, i.e., if $\mathcal{A}$ correctly guesses y_m and makes a random oracle query on input y_m. In this case, however, $\mathcal{C}$ learns y_m s.t. $(Y_m, y_m) \in R$ and therefore breaks the hard relation R.

A.4 Wallet Signature Adaptability

Lemma 6. *The adaptor wallet construction from Fig. 10 satisfies wallet signature adaptability as per Def. 17.*

The proof of this lemma follows directly from the pre-signature adaptability property of $\mathsf{RASig}_{R,\Sigma}$.