Generic Adaptor Signature

Xianrui Qin, Handong Cui, and John Yuen

The University of Hong Kong, Hong Kong
{xrqin,hdcui,thyuen}@cs.hku.hk

Abstract. Adaptor signature is becoming an increasingly important tool in solving the scalability and interoperability issues of blockchain application. It has many useful properties. Firstly, it can reduce on-chain cost when it is used in the off-chain authenticated communication. Secondly, it can increase the fungibility of transactions using it as the signature. Thirdly, it can help circumvent the limitation of the blockchain's scripting language.
In this paper, we propose the first generic construction of adaptor signatures which is compatible with different signature schemes. It can be used as a general framework to combine with different privacy-preserving cryptosystems. Finally, we propose blind adaptor signature and linkable ring adaptor signature. We believe they are of independent interests.

1 Introduction

Blockchains are decentralized platforms run by miners, where each transaction on the blockchain can be seen as an application formed of some script(s). The scripting language of a blockchain defines potential functionalities that can be implemented on blockchain. Bitcoin, for example, consists of very few scripts, which restricts its use mainly in coin transactions. Ethereum, on the other hand, has a Turing-complete scripting language that enables users to run more advanced and complicated applications.

A user who wants to deploy and execute a transaction needs to pay a fee to the miners. The fee is determined by the storage and computational costs of running each script of the transaction. Thus, it is beneficial to handle some operations off-chain to reduce the on-chain fee paid to the miners. In this manner, Poelstra introduced the notion of scriptless scripts [16], which is later named as adaptor signatures [3, 8].

Adaptor signatures can be seen as an extension over a digital signature. The rough idea is that a "pre-signature" is firstly generated, but still an uncompleted one yet. After being completed a witness of the statement embedded in the pre-signature will be revealed. The verification of the completed adaptor signature is done in the same way as the normal signature, for which the adapting trace is hidden.

There are some papers that instantiate adaptor signatures based on Schnorr and ECDSA digital signatures [4] [13]. However, their constructions are limited to Schnorr and ECDSA signature schemes and therefore is not generic.

1.1 Our Contribution

In this paper, we give a generic construction of adaptor signature. The main contribution of our generic adaptor signature is that it can be applied to various signature schemes used in different cryptocurrencies, making atomic swap between different cryptocurrencies feasible. Our generic adaptor signature can be applied to discrete-logarithm(DL)-based, RSA-based and lattice-based signatures.

Technically, our generic adaptor signature is constructed from a new type of identification scheme called $\mathsf{Type\text{-}TA}$. It is derived from a three-move canonical identification with some special properties. Roughly speaking, it requires (1) the commitment algorithm to be (additive/multiplicative) homomorphic (in order to combine the commitment with the statement); (2) the response algorithm is homomorphic with respect to commitment randomness (in order to facilitate the conversion to normal signature); (3) the verification algorithm is composed of running the commitment algorithm on the response (in order to complete the security proof of witness extractability). The Schnorr identification, the RSA-based GQ identification and the lattice-based identification in [7] are some examples of $\mathsf{Type\text{-}TA}$ identification.

Finally, we propose blind adaptor signature and linkable ring adaptor signature. We believe they can be important tools to increase atomicity and privacy for cryptocurrency.

1.2 Related work

A recent work that is related to our work is lockable signature [19]. Similar to adaptor signature, lockable signature allows one to compute a lock, which is the analogue of the pre-signature. However, the difference between them is that in adaptor signature, the pre-signature can be computed without knowing the witness of the given relation, while in lockable signature, computing a lock requires the signer's secret key.

2 Preliminaries

In this paper, we use λ as the security parameter, $\mathsf{negl}(\lambda)$ to represent a negligible function with respect to λ and Δ to represent an appropriate space of randomness defined by the algorithm.

2.1 $\mathsf{Type\text{-}T}$ Signature and Canonical Identification

$\mathsf{Type\text{-}T}$ signature is defined in [2], as shown in Algorithm 1.

- The SIGN algorithm uses the algorithm A to generate a commitment R using a randomness r (chosen from a randomness domain Δ_r). Then, the message and R are inputted to a function H to obtain a challenge c (within the challenge space Δ_c). Finally, the algorithm uses the function Z to generate the signature using the secret key sk, r and c.

Algorithm 1: Type-T Signature

```
1  Procedure Setup($\lambda$):
2      return param;
3  Procedure Sign(sk, $M$):
4      $r \leftarrow \Delta_r$;
5      $R = A(r)$;
6      $c = H(M, R)$;
7      $z = Z(\mathsf{sk}, r, c)$;
8      return $\sigma = (z, c)$;
9  Procedure KeyGen():
10     return (pk, sk);
11 Procedure Verify(pk, $\sigma$, $M$):
12     parse $\sigma = (z, c)$;
13     $R' = V(\mathsf{pk}, z, c)$;
14     if $c \neq H(M, R')$ then
15         return 0;
16     return 1;
```

Algorithm 2: Type-T Canonical Identification

```
1  Procedure Setup($\lambda$):
2      return param;
3  Procedure KeyGen():
4      return (pk, sk);
5  Procedure Proof1(sk):
6      $r \leftarrow_s \Delta_r$;
7      $R = A(r)$;
8      return $(R, r)$;
9  Procedure Ch($R$):
10     return $c$;
11 Procedure Proof2(sk, $r$, $c$):
12     return $z = Z(\mathsf{sk}, r, c)$;
13 Procedure Verify(pk, $z$, $c$):
14     $R' = V(\mathsf{pk}, c, z)$;
15     if $c \neq \mathrm{Ch}(R')$ then
16         return 0;
17     auxiliary checking with $R', c, z$;
18     return 1;
```

– The Verify algorithm allows the reconstruction of R' from the public key pk, z and c using the function V. The signature is validated by using H on the message and R'.

Schnorr signature [18], Guillou-Quisquater signature [10], Katz-Wang signature [11] and EdDSA [6] are examples of Type-T signatures.

Type-T **Canonical Identification.** Canonical identification [1] is a three-move public-key authentication protocol of a specific form. We first define Type-T canonical identification in Algorithm 2, based on the definition of Type-T signature in [2]. We add the additional checking in line 17 of Algorithm 2, which is useful for lattice-based construction. It is straightforward that after applying the Fiat-Shamir transformation to Type-T canonical identification, we obtain a Type-T signature.

We define the security of *impersonation under key only attack* for Type-T canonical identification.

Definition 1. *A* Type-T *canonical identification is secure against impersonation under key only attack if there is no PPT adversary $\mathcal{A}$ such that* $\mathbf{Adv}^{\mathsf{imp}}_{\mathcal{A}}(\lambda)$ *is* *non-negligible, where:*

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{imp}}(\lambda) := \Pr[\textsc{Verify}(\mathsf{pk}, z^*, c_{i^*}) = 1 | \mathsf{param} \leftarrow \textsc{Setup}(\lambda), (\mathsf{pk}, \mathsf{sk}) \leftarrow \textsc{KeyGen}(), (c_{i^*}, z^*) \leftarrow \mathcal{A}^{\mathsf{CH}(\cdot)}(\mathsf{param}, \mathsf{pk})].$$

For the i-th query $\mathsf{CH}(R_i)$, the oracle returns c_i to $\mathcal{A}$, and $i^* \in [1, q_c]$, q_c is the number of query to CH*.*

2.2 Adaptor Signature

A relation R with a language $L_R := \{Y | \exists y : (Y, y) \in R\}$ is said to be hard if (i) a probabilistic polynomial time (PPT) generator $\mathsf{LockGen}(\lambda)$ that outputs $(Y, y) \in R$, (ii) for every PPT algorithm $\mathcal{A}$, given $Y \in L_R$, the probability of $\mathcal{A}$ outputting y is negligible.

According to [3], an adaptor signature $\prod_{R,\Sigma}$ is defined with respect to a hard relation R and a signature scheme $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$.

Definition 2 (Adaptor Signature Scheme). *An adaptor signature* AS *scheme* $\prod_{R,\Sigma}$ *consists of four algorithms* $(\mathsf{PreSign}, \mathsf{PreVerify}, \mathsf{Adapt}, \mathsf{Ext})$ *defined below.*

- $\mathsf{PreSign}((\mathsf{pk}, \mathsf{sk}), Y, M)$*: on input a key pair* $(\mathsf{pk}, \mathsf{sk})$*, a statement* $Y \in L_R$ *and a message* $M \in \{0,1\}^*$*, outputs a pre-signature* $\hat{\sigma}$*.*
- $\mathsf{PreVerify}(Y, \mathsf{pk}, \hat{\sigma}, M)$*: on input a statement* $Y \in L_R$*, a pre-signature* $\hat{\sigma}$*, a public key* pk *and a message* M *, outputs a bit* b*.*
- $\mathsf{Adapt}((Y, y), \mathsf{pk}, \hat{\sigma}, M)$*: on input a statement-witness pair* (Y, y)*, a public key* pk*, a pre-signature* $\hat{\sigma}$ *and a message* M*, outputs a signature* σ*.*
- $\mathsf{Ext}(Y, \sigma, \hat{\sigma})$*: on input a statement* $Y \in L_R$*, a signature* σ *and a pre-signature* $\hat{\sigma}$*, outputs a witness* y *such that* $(Y, y) \in R$*, or* $\perp$*.*

Definition 3 (Pre-signature Adaptability). *An adaptor signature scheme* $\prod_{R,\Sigma}$ *satisfies pre-signature adaptability if for every message* M *in the message space, every statement/witness pair* $(Y, y) \in R$ *and for all pre-signature* $\hat{\sigma}$*, the following holds:*

$$Pr\left[\mathsf{Verify}(\mathsf{pk}, \sigma, M) = 1 \middle| \begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda), \\ \mathsf{PreVerify}(Y, \mathsf{pk}, \hat{\sigma}, M) = 1, \\ \sigma \leftarrow \mathsf{Adapt}((Y, y), \mathsf{pk}, \hat{\sigma}, M). \end{array}\right] = 1.$$

Definition 4 (Unforgeability). *An adaptor signature scheme* $\prod_{R,\Sigma}$ *is* $\mathsf{aEUF{-}CMA}$ *secure if for every PPT adversary* $\mathcal{A}$ *runing the experiment* $\mathsf{aSignForge}_{\mathcal{A},\prod_{R,\Sigma}}$ *defined in Algorithm 3,* $\Pr[\mathsf{aSignForge}_{\mathcal{A},\prod_{R,\Sigma}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$*.*

Definition 5 (Witness Extractability). *An adaptor signature scheme* $\prod_{R,\Sigma}$ *is witness extractable if for every PPT adversary* $\mathcal{A}$ *running the experiment* $\mathsf{aWitExt}_{\mathcal{A},\prod_{R,\Sigma}}$ *defined in Algorithm 4,* $\Pr[\mathsf{aWitExt}_{\mathcal{A},\prod_{R,\Sigma}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$*.*

Algorithm 3: Experiment $\mathsf{aSignForge}_{\mathcal{A},\prod_{R,\Sigma}}$

```
Procedure aSignForge_{A,∏_{R,Σ}}(λ):
    $\mathcal{Q} := \emptyset$;
    $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$;
    $M^* \leftarrow \mathcal{A}^{O_\mathsf{S},O_\mathsf{pS}}(\mathsf{pk})$ ;
    $(Y,y) \leftarrow \mathsf{LockGen}(\lambda)$;
    $\hat{\sigma} \leftarrow \mathsf{PreSign}((\mathsf{pk},\mathsf{sk}),Y,M^*)$;
    $\sigma^* \leftarrow \mathcal{A}^{O_\mathsf{S},O_\mathsf{pS}}(\hat{\sigma},Y)$;
    return
      $((M^* \notin \mathcal{Q}) \wedge \mathsf{Verify}(\mathsf{pk},\sigma^*,M^*) = 1)$;
Procedure $O_\mathsf{S}(M)$:
    $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk},M)$;
    $\mathcal{Q} := \mathcal{Q} \cup \{M\}$;
    return $\sigma$;
Procedure $O_\mathsf{pS}(M,Y)$:
    $\hat{\sigma} \leftarrow \mathsf{PreSign}((\mathsf{pk},\mathsf{sk}),Y,M)$;
    $\mathcal{Q} := \mathcal{Q} \cup \{M\}$;
    return $\hat{\sigma}$;
```

Algorithm 4: Experiment $\mathsf{aWitExt}_{\mathcal{A},\prod_{R,\Sigma}}$

```
Procedure aWitExt_{A,∏_{R,Σ}}(λ):
    $\mathcal{Q} := \emptyset$;
    $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$;
    $(M^*,Y) \leftarrow \mathcal{A}^{O_\mathsf{S},O_\mathsf{pS}}(\mathsf{pk})$ /* $O_\mathsf{S},O_\mathsf{pS}$ are the same as Algorithm 3 */
    $\hat{\sigma} \leftarrow \mathsf{PreSign}((\mathsf{pk},\mathsf{sk}),Y)$;
    $\sigma^* \leftarrow \mathcal{A}^{O_\mathsf{S},O_\mathsf{pS}}(\hat{\sigma})$;
    $y^* \leftarrow \mathsf{Ext}(Y,\sigma^*,\hat{\sigma})$;
    return $((M^* \notin \mathcal{Q}) \wedge (Y,y^*) \notin R \wedge \mathsf{Verify}(\mathsf{pk},\sigma^*,M^*) = 1)$;
```

3 Generic Adaptor Signature

In this section, we give a generic construction of adaptor signature, which is built from a new security notion called Type-TA canonical identification.

3.1 Building Block: Type-TA Canonical Identification

We define a new security notion called Type-TA canonical identification, which is a special case of a Type-T canonical identification.

Definition 6. *A* Type-TA *canonical identification is a* Type-T *canonical identification with some additional properties:*

1. *For all $y \in \Delta_r$, $(A(y),y)$ belongs to some hard relation R.*
2. *For all $r_1, r_2 \in \Delta_r$,*

$$A(r_1) \oplus_R A(r_2) = A(r_1 \oplus r_2), \quad A(r_1^{-1}) = (A(r_1))^{-1}.$$

where $\oplus$ is a group operation in Δ_r, $\oplus_R$ is a group operation in the domain of R. The inverse functions are defined in the corresponding group operations ($\oplus$ and $\oplus_R$).

3. *For all $r_1, r_2 \in \Delta_r$, $c \in \Delta_c$ and secret key* sk*,*

$$Z(\mathsf{sk}, r_1, c) \oplus r_2 = Z(\mathsf{sk}, r_1 \oplus r_2, c).$$

It implies that $\oplus$ is also a group operation in the domain of z.

4. *For all z, c and public key* pk*, there is a PPT algorithm V' such that:*

$$V(\mathsf{pk}, z, c) = A(z) \oplus_R V'(\mathsf{pk}, c).$$

Looking ahead, the first property is to define the hard relation for an adaptor signature. The second property is to combine the commitment and the statement $A(y)$ and form a new commitment. The third property is for the correctness of the Adapt algorithm. The fourth property is for the proof of witness extractability.

Schnorr Identification [17]. In Schnorr identification, $A(r) = g^r$ is a hard relation if the DL assumption holds. Consider $\oplus_R$ as a multiplication in a cyclic group and $\oplus$ as a modular addition, we have $A(r_1) \cdot A(r_2) := g^{r_1} \cdot g^{r_2} = g^{r_1+r_2} = A(r_1+r_2)$ and $A(-r) = g^{-r} = A(r)^{-1}$. Also $Z(\mathsf{sk}, r_1, c) + r_2 := r_1 + c \cdot \mathsf{sk} + r_2 = (r_1+r_2)+c\cdot\mathsf{sk} = Z(\mathsf{sk}, r_1+r_2, c)$. Finally, $V(\mathsf{pk}, z, c) := g^z \cdot \mathsf{pk}^c = A(z) \cdot V'(\mathsf{pk}, c)$. Hence, Schnorr identification is a $\mathsf{Type\text{-}TA}$ identification.

GQ Identification [10]. In GQ identification, $A(r) = r^v$ is a hard relation if the standard RSA assumption holds. Consider both $\oplus_R$ and $\oplus$ as multiplication in a cyclic group. Then we have $A(r_1) \cdot A(r_2) := r_1^v \cdot r_2^v = (r_1 \cdot r_2)^v = A(r_1 \cdot r_2)$ and $A(r^{-1}) = r^{-v} = A(r)^{-1}$. Also $Z(\mathsf{sk}, r_1, c) \cdot r_2 := r_1 \mathsf{sk}^c \cdot r_2 = (r_1 \cdot r_2) \cdot \mathsf{sk}^c = Z(\mathsf{sk}, r_1 \cdot r_2, c)$. Finally, $V(\mathsf{pk}, z, c) := z^v \cdot \mathsf{pk}^c = A(z) \cdot V'(\mathsf{pk}, c)$. Hence, GQ identification is a $\mathsf{Type\text{-}TA}$ identification.

Lattice-based Identification [7]. In lattice-based identification [7], $A(\boldsymbol{y}) = \boldsymbol{A}\boldsymbol{y}$ is a hard relation where $|\boldsymbol{y}| \leq \beta_{\mathsf{SIS}}$, if the Module-SIS assumption holds. Consider both $\oplus_R$ and $\oplus$ as modular addition. Next we have $A(\boldsymbol{y}_1) + A(\boldsymbol{y}_2) := \boldsymbol{A}\boldsymbol{y}_1 + \boldsymbol{A}\boldsymbol{y}_2 = \boldsymbol{A}(\boldsymbol{y}_1 + \boldsymbol{y}_2) = A(\boldsymbol{y}_1 + \boldsymbol{y}_2)$ where $|\boldsymbol{y}_1 + \boldsymbol{y}_2| \leq \beta_{\mathsf{SIS}}$ and $A(-\boldsymbol{y}) = \boldsymbol{A}(-\boldsymbol{y}) = -\boldsymbol{A}\boldsymbol{y} = -A(\boldsymbol{y})$. Also $Z(\mathsf{sk}, \boldsymbol{y}_1, c) + \boldsymbol{y}_2 := \boldsymbol{y}_1 + c \cdot \mathsf{sk} + \boldsymbol{y}_2 = (\boldsymbol{y}_1 + \boldsymbol{y}_2) + c \cdot \mathsf{sk} = Z(\mathsf{sk}, \boldsymbol{y}_1 + \boldsymbol{y}_2, c)$ where $|\boldsymbol{y}_1 + \boldsymbol{y}_2| \leq \beta_{\mathsf{SIS}}$. Finally, $V(\mathsf{pk}, \boldsymbol{z}, c) := \boldsymbol{A}\boldsymbol{z} - c \cdot \mathsf{pk} = A(\boldsymbol{z}) + V'(\mathsf{pk}, c)$. Although we add an additional condition to bound the $|\boldsymbol{y}| \leq \beta_{\mathsf{SIS}}/2$, if we see it as a inherent requirement of lattice cryptography, lattice-based identification in [7] is also a $\mathsf{Type\text{-}TA}$ identification.

3.2 Our Construction

Then we give the generic adaptor signature in Algorithm 5.

Security Proof. The correctness of the generic adaptor signature is straightforward.

Theorem 1. *Our generic adaptor signature has pre-signature adaptability.*

Proof. Next we show that it has pre-signature adaptability. For a pre-signature $\hat{\sigma} = (\hat{z}, c)$ which passes the $\mathsf{PreVerify}$ algorithm, we have $R = V(\mathsf{pk}, \hat{z}, c) \oplus_R Y$

Algorithm 5: Generic adaptor signature (AS) from a Type-TA identification scheme ID for the language $L := \{Y : \exists y \in \Delta_r : Y = A(y)\}$.

```
Procedure Setup($\lambda$):
    define hash $H : \{0,1\}^* \to \Delta_c$;
    $\mathsf{param}_I \leftarrow$ ID.SETUP($\lambda$);
    return $(\mathsf{param}_I, H)$;
Procedure KeyGen():
    return ID.KEYGEN();
Procedure PreSign((pk, sk), $Y$, $M$):
    $r \leftarrow \Delta_r$;
    $R = A(r) \oplus_R Y$;
    $c = H(M, R)$;
    $\hat{z} = Z(\mathsf{sk}, r, c)$;
    return $\hat{\sigma} = (\hat{z}, c)$;
Procedure PreVerify($Y$, pk, $\hat{\sigma}$, $M$):
    parse $\hat{\sigma} = (\hat{z}, c)$;
    $R' = V(\mathsf{pk}, \hat{z}, c) \oplus_R Y$;
    if $c \neq H(M, R')$ then
        return 0;
    return 1;
Procedure Adapt(($Y$, $y$), pk, $\hat{\sigma}$, $M$):
    parse $\hat{\sigma} = (\hat{z}, c)$;
    $z = \hat{z} \oplus y$;
    return $\sigma = (z, c)$;
Procedure Ext($Y$, $\hat{\sigma}$, $\sigma$):
    parse $\hat{\sigma} = (\hat{z}, \hat{c})$ and $\sigma = (z, c)$;
    if $\hat{c} = c$ then
        return $y = (\hat{z})^{-1} \oplus z$;
    return $\perp$;
```

and $c = H(M, R)$. By the Adapt algorithm, we have $\sigma = (z = \hat{z} \oplus y, c)$. We further have

$$\begin{aligned} c &= H(M, R) \\ &= H(M, V(\mathsf{pk}, \hat{z}, c) \oplus_R Y) \\ &= H(M, A(\hat{z}) \oplus_R V'(\mathsf{pk}, c) \oplus_R Y) \\ &= H(M, A(z \ominus y) \oplus_R V'(\mathsf{pk}, c) \oplus_R Y) \\ &= H(M, A(z) \ominus_R A(y) \oplus_R V'(\mathsf{pk}, c) \oplus_R Y) \\ &= H(M, A(z) \ominus_R Y \oplus_R V'(\mathsf{pk}, c) \oplus_R Y) \\ &= H(M, A(z) \oplus_R V'(\mathsf{pk}, c)) \\ &= H(M, V(\mathsf{pk}, z, c)) \end{aligned} \tag{1}$$

It follows that σ is valid, i.e., $\mathsf{Verify}(\mathsf{pk}, \sigma, M) = 1$.

Theorem 2. *Our generic adaptor signature is aEUF-CMA secure in the random oracle model if the identification scheme* ID *is secure against impersonation under key only attack.*

Proof. Suppose that there is a PPT adversary $\mathcal{A}$ breaking the aEUF-CMA security of our generic adaptor signature. We build an algorithm $\mathcal{B}$ to break the security of ID. First, the challenger $\mathcal{C}$ of ID gives param and pk to $\mathcal{B}$. $\mathcal{B}$ forwards param and pk to $\mathcal{A}$.

For all signing oracle queries from $\mathcal{A}$ on a message M, $\mathcal{B}$ picks a random z and c from their corresponding domain and computes $R = V(\mathsf{pk}, z, c)$. $\mathcal{B}$ sets $c = H(M, R)$ in the random oracle H. $\mathcal{B}$ returns (z, c) to $\mathcal{A}$.

For the pre-signing oracle queries from $\mathcal{A}$ with input (M, Y), $\mathcal{B}$ picks a random $\hat{z}$ and c from their corresponding domain and computes $R = V(\mathsf{pk}, \hat{z}, c) \oplus_R Y$. $\mathcal{B}$ sets $c = H(M, R)$ in the random oracle H. $\mathcal{B}$ returns $(\hat{z}, c)$ to $\mathcal{A}$.

For all random oracle queries $H(M, R)$, $\mathcal{B}$ queries the oracle $\mathrm{CH}(R)$ from $\mathcal{C}$ and obtains c. $\mathcal{B}$ returns c to $\mathcal{A}$.

If $\mathcal{A}$ outputs a valid forgery (z^*, c^*) on a message M^*, we have $R^* = V(\mathsf{pk}, z^*, c^*), c^* = H(M^*, R^*)$. Then $\mathcal{B}$ returns (c^*, z^*) as the attack to $\mathcal{C}$. □

Theorem 3. *Our generic adaptor signature is witness extractable.*

Proof. Suppose that there is a PPT adversary $\mathcal{A}$ breaking the witness extractability of our generic adaptor signature. All oracle queries can be simulated by using the secret key.

In the challenge phase, $\mathcal{A}$ is given a pre-signature $\hat{\sigma} = (\hat{z}, c)$ for a message M^* and a statement Y, where $\hat{z} = Z(\mathsf{sk}, r, c)$ and $c = H(M^*, R)$. Then $\mathcal{A}$ outputs a full signature $\sigma^* = (z^*, c^*)$, where $R^* = V(\mathsf{pk}, z^*, c^*)$. If $\mathcal{A}$ wins, it implies that $\mathsf{Ext}(Y, \sigma^*, \hat{\sigma})$ did not output $\perp$. Hence $c = c^*$. By the collision resistant property of H, then $R^* = R$. It implies $V(\mathsf{pk}, z^*, c^*) = A(r) \oplus_R Y$. By the Ext algorithm, we can compute $y = (\hat{z})^{-1} \oplus z^*$. Hence we have:

$$V(\mathsf{pk}, \hat{z} \oplus y, c^*) = A(r) \oplus_R Y = V(\mathsf{pk}, \hat{z}, c^*) \oplus_R Y.$$

Also by the property 2 and 4 of the $\mathsf{Type\text{-}TA}$ identification, we have:

$$V(\mathsf{pk}, \hat{z} \oplus y, c^*) = A(\hat{z}) \oplus_R A(y) \oplus_R V'(\mathsf{pk}, c^*) = V(\mathsf{pk}, \hat{z}, c^*) \oplus_R A(y).$$

Hence we can extract y such that $A(y) = Y$. □

3.3 Discussion on ECDSA Adaptor Signature

ECDSA is the most commonly used signature scheme in cryptocurrency. ECDSA does not fall into the category of $\mathsf{Type\text{-}TA}$ signature and hence cannot be used with our generic construction. In particular, the inverse computation in ECDSA makes it difficult to compute an adaptor signature. The first provably secure ECDSA adaptor signature is given in [14]. Nevertheless, the language $L := \{Y : \exists y : Y = g^y\}$ used in the ECDSA adaptor signature is the same as the language used in our Schnorr-based instantiation.

4 Blind Adaptor Signature

In this section, we propose the notion of blind adaptor signature. In a blind signature scheme, a user can obtain a signature from a signer on a message M such that: (1) the signer cannot recognize the signature later (blindness, which implies that the message M is unknown to the signer) and (2) the user can compute a single signature per interaction with the signer (one-more unforgeability). Blind signature is used to provide private fiat-to-cryptocurrency swap in [20].

4.1 Security Notions

A blind signature scheme BS consists of the following algorithms:

- $\mathsf{Setup}(\lambda)$: It takes the security parameter 1^λ and returns public parameters param.
- $\mathsf{KeyGen}(\mathsf{param})$: It takes the public parameters param and returns a secret/public key pair $(\mathsf{sk}, \mathsf{pk})$.
- $\mathsf{Sign}(\mathsf{sk}), \mathsf{User}(\mathsf{pk}, M)$: an interactive protocol is run between the signer with private input a secret key sk and the user with private input a public key pk and a message M. The signer outputs $b = 1$ if the interaction completes successfully and $b = 0$ otherwise, while the user outputs a signature σ if it ends correctly, and $\perp$ otherwise.
- $\mathsf{Verify}(\mathsf{pk}, M, \sigma)$: it takes a public key pk, a message M, and a signature σ, and returns 1 if σ is valid on M under pk and returns 0 otherwise.

Definition 7 (Blind Adaptor Signature Scheme). *A blind adaptor signature (*BAS*) scheme* $\prod_{R,\mathsf{BS}}$ *with respect to a hard relation* R *with a language* $L_R := \{Y | \exists y : (Y, y) \in R\}$ *and a blind signature scheme* BS *consists of four algorithms* $(\mathsf{PreSign}, \mathsf{PreVerify}, \mathsf{Adapt}, \mathsf{Ext})$ *defined below.*

- $\mathsf{PreSign}(\mathsf{sk}, Y), \mathsf{User}(\mathsf{pk}, M, Y)$ *an interactive protocol is run between the signer with private input a secret key* sk *and the user with private input a public key* pk *and a message* M*. A statement* $Y \in L_R$ *is the public input. The signer outputs* $b = 1$ *if the interaction completes successfully and* $b = 0$ *otherwise, while the user outputs a pre-signature* $\hat{\sigma}$ *if it ends correctly, and* $\perp$ *otherwise.*
- $\mathsf{PreVerify}$*,* Adapt *and* Ext *are the same as the adaptor signature.*

4.2 Security Models

For the security models of BAS, we follow the security requirements of blind signature in [9] to define *one-more unforgeability* and *blindness*. We also define *pre-signature adaptability* and *witness extractability* as the adaptor signature.

In other to define the security model for BAS, suppose that there are N_s (resp. N_p) interactions by the signer in the $\mathsf{Sign}(\mathsf{sk})$ (resp. $\mathsf{PreSign}(\mathsf{sk}, Y)$) algorithm. We use $(m', st_1) \leftarrow \mathsf{Sign}_1(\mathsf{sk}, m)$ (resp. $(m', st_1) \leftarrow \mathsf{PreSign}_1(\mathsf{sk}, Y, m)$) to represent the first interaction, where m is the message received by the signer, m' is the message output and st_1 is the internal state. We use $(m', st_i) \leftarrow \mathsf{Sign}_i(st_{i-1}, m)$ (resp. $(m', st_1) \leftarrow \mathsf{PreSign}_i(st_{i-1}, m)$) to represent the i-th interaction, for $i \in [2, N_s - 1]$ (resp. $i \in [2, N_p - 1]$). We use $(m', b) \leftarrow \mathsf{Sign}_{N_s}(st_{N_s-1}, m)$ (resp. $(m', b) \leftarrow \mathsf{PreSign}_{N_p}(st_{N_p-1}, m)$) to represent the last interaction, where b is a bit.

One-more Unforgeability. The unforgeability model is defined to capture the attack that the adversary returns n distinct message-signature pairs when he is only given $k_2 < n$ pairs during the oracle queries. It is commonly known as the one-more unforgeability in blind signature [9].

Definition 8 (One-more Unforgeability). *A* BAS *scheme* $\Xi_{R,\Sigma}$ *is* omaEUF–CMA *secure if for every PPT adversary* $\mathcal{A}$ *running the experiment* $\mathsf{omaSignForge}_{\mathcal{A},\Xi_{R,\Sigma}}$ *defined in Algorithm 6,* $\Pr[\mathsf{omaSignForge}_{\mathcal{A},\Xi_{R,\Sigma}}(\lambda) = 1] \le \mathsf{negl}(\lambda)$.

Blindness. The blindness security model of BAS is the same as that of blind signature in [9], except that the algorithm User takes an extra input Y. It is because BAS mainly modifies the algorithms in the signer side.
Pre-signature Adaptability. The pre-signature adaptability of BAS is the same as that of an adaptor signature. It is because the PreSign algorithm is not involved in the model.
Witness extractability. The witness extractability of BAS is different from that of the adaptor signature. It is because the message signed by the oracles O_{S} and O_{pS} is unknown to the challenger. Hence, we have to change the definition of the oracles to avoid giving a full signature to the adversary.

Definition 9 (Witness Extractability). *A* BAS *scheme* $\prod_{R,\Sigma}$ *is witness extractable if for every PPT adversary* $\mathcal{A}$ *running the experiment* $\mathsf{baWitExt}_{\mathcal{A},\prod_{R,\Sigma}}$ *defined in Algorithm 7,* $\Pr[\mathsf{baWitExt}_{\mathcal{A},\prod_{R,\Sigma}}(\lambda) = 1] \le \mathsf{negl}(\lambda)$.

Algorithm 8: Clause Blind Schnorr adaptor signature for the language $L := \{Y | \exists y : Y = g^y\}$.

```
1  Procedure Setup($\lambda$):
2      return param = $(\mathbb{G}, g)$;
3  Procedure KeyGen(param):
4      sk $\xleftarrow{\$} \mathbb{Z}_p$, pk $= g^{\mathsf{sk}}$;
5      return (sk, pk);
6  Procedure PreSign(sk, $Y$) $\leftrightarrow$ User(pk, $M$, $Y$):
7      PreSign: For $i \in [0,1]$, $r_i \xleftarrow{\$} \mathbb{Z}_p$, $R_i = g^{r_i} \cdot Y$, send $R_0, R_1$ to user;
8      User: For $i \in [0,1]$, $\alpha_i, \beta_i \xleftarrow{\$} \Delta_r$, $R'_i = R_i \cdot g^{\alpha_i} \cdot \mathsf{pk}^{-\beta_i}$,
       $c_i = H(M, R'_i)$, $\hat{c}_i = c_i + \beta_i$, send $\hat{c}_0, \hat{c}_1$ to signer;
9      PreSign: Pick a bit $b \xleftarrow{\$} \{0,1\}$, send $z = r_b - \hat{c}_b \cdot \mathsf{sk}$ and $b$ to user;
10     User: If $R_b \ne g^z \cdot Y \cdot \mathsf{pk}^{\hat{c}_b}$, return $\perp$, else return $\hat{\sigma} = (\hat{z} = z + \alpha_b, c_b)$;
11 Procedure PreVerify($Y$, pk, $\hat{\sigma}$, $M$):
12     parse $\hat{\sigma} = (\hat{z}, c)$;
13     $R' = g^{\hat{z}} \cdot Y \cdot \mathsf{pk}^c$;
14     if $c \ne H(M, R')$ then
15         return 0;
16     return 1;
17 Procedure Adapt(($Y$, $y$), pk, $\hat{\sigma}$, $M$):
18     parse $\hat{\sigma} = (\hat{z}, c)$;
19     $z = \hat{z} + y$;
20     return $\sigma = (z, c)$;
21 Procedure Ext($Y$, $\hat{\sigma}$, $\sigma$):
22     parse $\hat{\sigma} = (\hat{z}, \hat{c})$ and $\sigma = (z, c)$;
23     if $\hat{c} = c$ then
24         return $y = z - \hat{z}$;
25     return $\perp$;
26 Procedure Verify(pk, $M$, $\sigma$):
27     parse $\sigma = (z, c)$;
28     $R' = g^z \cdot \mathsf{pk}^c$;
29     if $c \ne H(M, R')$ then
30         return 0;
31     return 1;
```

4.3 Clause Blind Schnorr Adaptor Signature

The clause blind Schnorr signature [9] can be viewed as the Schnorr identification with a crafted challenge, such that the message to be signed is hidden from the signer and the signer eventually signs one out of the two commitments. It is secure against the recent attack on the ROS problem [5]. Following our generic construction of adaptor signature in the last section, the clause blind Schnorr adaptor signature can be constructed as in Algorithm 8.

The clause blind Schnorr adaptor signature is omaEUF-CMA secure, has perfect blindness and pre-signature adaptability.

4.4 Security Proofs of Clause Blind Schnorr Adaptor Signature

Theorem 4. *The clause blind Schnorr adaptor signature is omaEUF-CMA secure if the clause blind Schnorr signature is omEUF-CMA secure.*

Proof. Suppose that there is a PPT adversary $\mathcal{A}$ breaking the aEUF-CMA security of the clause blind Schnorr adaptor signature. We build an algorithm $\mathcal{B}$ to break the security of the clause blind Schnorr signature. First, the challenger $\mathcal{C}$ of the clause blind Schnorr signature gives param and pk to $\mathcal{B}$. $\mathcal{B}$ forwards param and pk to $\mathcal{A}$.

For all signing oracle queries from $\mathcal{A}$ on a message M, $\mathcal{B}$ forwards them to the signing oracle of $\mathcal{C}$ to get the answer.

For the pre-signing oracle queries from $\mathcal{A}$ with input Y and session ID k, $\mathcal{B}$ asks the signing oracle of $\mathcal{C}$ and obtains (R_0, R_1). $\mathcal{B}$ returns $\hat{R}_0 = R_0 \cdot Y$ and $\hat{R}_1 = R_1 \cdot Y$ to $\mathcal{A}$. After that when $\mathcal{A}$ sends $\hat{c}_0, \hat{c}_1$ to $\mathcal{B}$ with the session ID k, $\mathcal{B}$ forwards the query to $\mathcal{C}$ and obtains (z, b). $\mathcal{B}$ answers $\mathcal{A}$ by (z, b).

After running $n-1$ queries to the signing oracle or the pre-signing oracle, $\mathcal{A}$ outputs n distinct message-signature pairs $(M_j, \sigma_j = (z_j, c_j))$ for $j \in [1, n]$ such that $R_j = g^{z_j} \cdot \mathsf{pk}^{c_j}, c_j = H(M_j, R_j)$. Then $\mathcal{B}$ forwards them as the attack to $\mathcal{C}$.

Theorem 5. *The clause blind Schnorr adaptor signature has prefect blindness.*

Proof. (Sketch) Observe that the blinding step of the user during the $\mathsf{PreSign}$ algorithm is the same as the Sign algorithm in the original blind signature scheme. Hence, the clause blind Schnorr adaptor signature has the same level of blindness as the clause blind Schnorr signature, i.e., prefect blindness.

Theorem 6. *The clause blind Schnorr adaptor signature has pre-signature adaptability.*

Proof. If a pre-signature $\hat{\sigma} = (\hat{z}, c)$ passes $\mathsf{PreVerify}$, we have:

$$R' = g^{\hat{z}} Y \mathsf{pk}^c = g^{\hat{z}+y} \mathsf{pk}^c, \quad c = H(M, R').$$

After Adapt, we have the full signature $\sigma = (z = \hat{z} + y, c)$. Then σ can pass Verify since $g^z \mathsf{pk}^c = g^{\hat{z}+y} \mathsf{pk}^c = R'$.

Theorem 7. *The clause blind Schnorr adaptor signature has witness extractability.*

Proof. Suppose that there is a PPT adversary $\mathcal{A}$ breaking the witness extractability of our clause blind Schnorr adaptor signature. All oracle queries can be simulated by using the secret key.

In the challenge phase, $\mathcal{A}$ is given a pre-signature $\hat{\sigma} = (\hat{z}, c)$ for a message M^* and a statement Y, where $\hat{z} = r - \hat{c} \cdot \mathsf{sk} + \alpha$ and $c = H(M^*, R') + \beta$. Then $\mathcal{A}$ outputs a full signature $\sigma^* = (z^*, c^*)$, where $R^* = g^{z^*}\mathsf{pk}^{c^*}$. If $\mathcal{A}$ wins, then $\mathsf{Ext}(Y, \sigma, \hat{\sigma})$ did not output $\perp$. It implies $c^* = c - \beta$. By the collision resistant property of H, then $R^* = R'$. It implies $g^{z^*}\mathsf{pk}^{c^*} = g^r \cdot Y \cdot g^\alpha \cdot \mathsf{pk}^{-\beta}$. By the Ext function, $\mathcal{B}$ computes $y = z^* - \hat{z}$. Hence we have:

$$g^{z^*}\mathsf{pk}^{c^*} = g^{y+\hat{z}}\mathsf{pk}^{c} = g^{y+r-\hat{c}\cdot\mathsf{sk}+\alpha}\mathsf{pk}^{\hat{c}-\beta} = g^r \cdot g^y \cdot g^\alpha \cdot \mathsf{pk}^{-\beta}.$$

Hence $\mathcal{B}$ can extract y such that $A(y) = Y$.

4.5 Discussion on Blind ECDSA Signature

ECDSA is commonly used in cryptocurrencies such as Bitcoin and Ethereum. Recently, blind ECDSA signature is proposed in [20] to provide recipient anonymity. However, there is no discussion about one-more unforgeability in [20]. It is not clear if any additional model is needed (e.g., the generic group model). Furthermore, the security proof for unforgeability in [20] is not rigorous: it is not clear how the signing oracle can be simulated without knowing the secret key. Therefore, we left the construction of blind ECDSA (adaptor) signature secure against one-more unforgeability as an interesting open problem.

5 Linkable Ring Adaptor Signature

In a linkable ring signature scheme [12], a signer has anonymity by hiding himself among a set of verification keys. However, if he signed twice, the two signatures are linked. It is useful in privacy-preserving cryptocurrency (e.g., Monero) to provide sender anonymity, while detecting double spending is feasible.

We will show that our generic adaptor signature can be applied to the linkable ring signature. In particular, we introduce the notion of *Linkable Ring Adaptor Signature* and define the corresponding security model. We will give a concrete construction based on the recent RingCT3.0 protocol [21].

5.1 Security Notions

A linkable ring adaptor signature (LRAS) scheme with respect to a hard relation R with a language $L_R := \{Y | \exists y : (Y, y) \in R\}$ and a linkable ring signature scheme $\Sigma = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Link})$ consists of four algorithms $\Xi_{R,\Sigma} =$ ($\mathsf{PreSign}$, $\mathsf{PreVerify}$, Adapt, Ext) defined as:

– $\mathsf{PreSign}(\mathsf{sk}, m, \vec{\mathsf{vk}}, Y)$: On input a secret key sk, a message m, a set of verification keys $\vec{\mathsf{vk}} = (\mathsf{vk}_1, \ldots, \mathsf{vk}_\mathsf{n})$ and a statement $Y \in L_R$, outputs a pre-signature $\hat{\sigma}$.
– $\mathsf{PreVerify}(m, \vec{\mathsf{vk}}, Y, \hat{\sigma})$: On input a message $m \in \{0,1\}^*$, a set of verification keys $\vec{\mathsf{vk}} = (\mathsf{vk}_1, \ldots, vk_n)$, a statement $Y \in L_R$ and a pre-signature $\hat{\sigma}$, outputs 1 for valid pre-signature or 0 otherwise.
– $\mathsf{Adapt}((Y, y), \vec{\mathsf{vk}}, \hat{\sigma}, m)$: On input $(Y, y) \in R$, a set of verification keys $\vec{\mathsf{vk}}$, a pre-signature $\hat{\sigma}$ and a message m, outputs a signature σ.
– $\mathsf{Ext}(Y, \hat{\sigma}, \sigma)$: on input a statement $Y \in L_R$, a signature σ and a pre-signature $\hat{\sigma}$, outputs a witness y such that $(Y, y) \in R$, or $\perp$.

Algorithm 9: Experiment $\mathsf{aSignForge}_{\mathcal{A},\Xi_{R,\Sigma}}$

```
Procedure $\mathit{aSignForge}_{\mathcal{A},\Xi_{R,\Sigma}}(\lambda)$:
    $\mathcal{Q} := \emptyset$ , $\mathcal{F} := \emptyset$, $\mathcal{T} := \emptyset$;
    $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(1^\lambda)$, for i $\in [1, N]$;
    $\vec{\mathsf{vk}} = \{\mathsf{vk}_1, \mathsf{vk}_2, ..., \mathsf{vk}_N\}$;
    $(m^*, \mathsf{vk}_{i^*}, \hat{\mathbf{vk}}) \leftarrow \mathcal{A}^{O_\mathsf{S}(\cdot), O_\mathsf{pS}(\cdot,\cdot), O_\mathsf{Corrupt}(\cdot)}(\vec{\mathsf{vk}})$, where $\mathsf{vk}_{i^*} \in \hat{\mathbf{vk}} \subseteq \vec{\mathsf{vk}}$ ;
    $(\mathsf{Y}, \mathsf{y}) \leftarrow \mathsf{LockGen}(\lambda)$;
    $\hat{\sigma} \leftarrow \mathsf{PreSign}(\mathsf{sk}_{i^*}, m^*, \hat{\mathbf{vk}}, \mathsf{Y})$;
    $(\tilde{\mathbf{vk}}, \sigma^*) \leftarrow \mathcal{A}^{O_\mathsf{S}(\cdot), O_\mathsf{pS(\cdot,\cdot)}, O_\mathsf{Corrupt}(\cdot)}(\hat{\sigma}, \mathsf{Y})$;
    return $((\tilde{\mathbf{vk}} \subseteq \vec{\mathsf{vk}} \backslash \mathcal{F}) \wedge ((\star, m^*, \tilde{\mathbf{vk}}) \notin \mathcal{Q}) \wedge \mathsf{Verify}(m^*, \tilde{\mathbf{vk}}, \sigma^*) = 1)$;
Procedure $O_\mathsf{Corrupt}(j)$:
    $\mathcal{F} := \mathcal{F} \cup \{\mathsf{vk}_j\}$;
    return $\mathsf{sk}_j$;
Procedure $O_\mathsf{S}(m, i, \overline{\vec{\mathsf{vk}}})$:
    $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_i, m, \overline{\vec{\mathsf{vk}}})$ /* Require that $\mathsf{vk}_i \in \overline{\vec{\mathsf{vk}}}$ */
    $\mathcal{Q} := \mathcal{Q} \cup \{i, m, \overline{\vec{\mathsf{vk}}}\}$ ;
    return $\sigma$;
Procedure $O_\mathsf{pS}(m, i, \overline{\vec{\mathsf{vk}}}, Y)$:
    $\hat{\sigma} \leftarrow \mathsf{PreSign}(\mathsf{sk}_i, m, \overline{\vec{\mathsf{vk}}}, Y)$ /* Require that $\mathsf{vk}_i \in \overline{\vec{\mathsf{vk}}}$ and $Y \in L_R$
        */
    $\mathcal{Q} := \mathcal{Q} \cup \{i, m, \overline{\vec{\mathsf{vk}}}\}$ ;
    return $\hat{\sigma}$;
```

5.2 Security Models

The security models for LRAScombines the security requirements from both linkable ring signatures (unforgeability, linkable anonymity, non-slanderability) and adaptor signatures (unforgeability, pre-signature adaptability, witness extractability). They are formally given in the following definitions.

Definition 10 (aEUF–CMA security). *A* LRAS *scheme* $\Xi_{R,\Sigma}$ *is* aEUF–CMA *secure if for every PPT adversary* $\mathcal{A}$ *runing the experiment* $\mathsf{aSignForge}_{\mathcal{A},\Xi_{R,\Sigma}}$ *defined in Algorithm 9,* $\Pr[\mathsf{aSignForge}_{\mathcal{A},\Xi_{R,\Sigma}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Definition 11 (Pre-signature anonymity). *A* LRAS *scheme* $\Xi_{R,\Sigma}$ *achieves linkable anonymity if for any PPT adversary* $\mathcal{A}$ *running the experiment* $\mathsf{aAnon}_{\mathcal{A},\Xi_{R,\Sigma}}$ *defined in Algorithm 10,* $\left|\Pr[\mathsf{aAnon}_{\mathcal{A},\Xi_{R,\Sigma}}(\lambda) = 1] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$.

Algorithm 10: Experiment $\mathsf{aAnon}_{\mathcal{A},\Xi_{R,\Sigma}}$

```
1  Procedure $\mathsf{aAnon}_{\mathcal{A},\Xi_{R,\Sigma}}(\lambda)$:
2      $\mathcal{Q} := \emptyset$;
3      $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(1^\lambda)$, for $i \in [1, N]$;
4      $\boldsymbol{vk} = \{\mathsf{vk}_1, \mathsf{vk}_2, ..., \mathsf{vk}_N\}$;
5      $(m^*, i_0, i_1, \tilde{\boldsymbol{vk}}, Y) \leftarrow \mathcal{A}^{O_\mathsf{S}(\cdot), O_\mathsf{pS}(\cdot,\cdot)}(\boldsymbol{vk})$,  where $\mathsf{vk}_{i_0}, \mathsf{vk}_{i_1} \in \tilde{\boldsymbol{vk}} \cup \boldsymbol{vk}$
          /* $O_\mathsf{S}, O_\mathsf{pS}$ are the same as Algorithm 9 */
6      $b \xleftarrow{\$} \{0, 1\}$;
7      $\hat{\sigma} \leftarrow \mathsf{PreSign}(\mathsf{sk}_{i_b}, m^*, \tilde{\boldsymbol{vk}}, Y)$;
8      $b' \leftarrow \mathcal{A}^{O_\mathsf{S}, O_\mathsf{pS}}(\hat{\sigma}, Y)$;
9      if $(b = b' \wedge \{i_0, \star, \star\} \notin \mathcal{Q} \wedge \{i_1, \star, \star\} \notin \mathcal{Q})$ then
10         return 1;
11     return 0;
```

Definition 12 (Linkability). *A* LRAS *scheme* $\Xi_{R,\Sigma}$ *satisfies pre-signature linkability w.r.t. insider corruption if for any PPT adversary* $\mathcal{A}$ *running the experiment* $\mathsf{aLink}_{\mathcal{A},\Xi_{R,\Sigma}}$ *defined in Algorithm 11,* $\left|\Pr[\mathsf{aLink}_{\mathcal{A},\Xi_{R,\Sigma}}(\lambda) = 1] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$.

Algorithm 11: Experiment $\mathsf{aLink}_{\mathcal{A},\Xi_{\mathsf{R},\Sigma}}$

```
1  Procedure $\mathsf{aLink}_{\mathcal{A},\Xi_{R,\Sigma}}(\lambda)$:
2      $\mathcal{Q} := \emptyset$, $\mathcal{F} := \emptyset$;
3      $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(1^\lambda)$, for $i \in [1, N]$;
4      $\boldsymbol{vk} = \{\mathsf{vk}_1, \mathsf{vk}_2, ..., \mathsf{vk}_N\}$;
5      $(m_i, \tilde{\boldsymbol{vk}}_i, \sigma_i)_{i=1,2} \leftarrow \mathcal{A}^{O_\mathsf{S}, O_\mathsf{pS}, O_\mathsf{Corrupt}}(\boldsymbol{vk})$ /* $O_\mathsf{S}, O_\mathsf{pS}, O_\mathsf{Corrupt}$ are the
          same as Algorithm 9 */
6      $b_1 := \mathsf{Verify}(m_i, \tilde{\boldsymbol{vk}}_i, \sigma_i)_{i=1,2}$;
7      $b_2 := \mathsf{Link}((m_1, \tilde{\boldsymbol{vk}}_1, \sigma_1), (m_2, \tilde{\boldsymbol{vk}}_2, \sigma_2))$;
8      $b_3 := \{\star, m_i, \tilde{\boldsymbol{vk}}_i\}_{i=1,2} \notin \mathcal{Q}$;
9      $b_4 := \left|((\tilde{\boldsymbol{vk}}_1 \cup \tilde{\boldsymbol{vk}}_2) \cap \mathcal{F}) \cup ((\tilde{\boldsymbol{vk}}_1 \cup \tilde{\boldsymbol{vk}}_2) \backslash \boldsymbol{vk})\right|$;
10     return$(b_1 = 1 \wedge b_2 = 0 \wedge b_3 = 1 \wedge b_4 \leq 1)$;
```

Definition 13 (Non-Slanderability). *A* LRAS *scheme* $\Xi_{R,\Sigma}$ *satisfies non-slanderability if for any PPT adversary* $\mathcal{A}$ *running the experiment* $\mathsf{aSlan}_{\mathcal{A},\Xi_{R,\Sigma}}$ *defined in Algorithm 12,* $\left|\Pr[\mathsf{aNSlan}_{\mathcal{A},\Xi_{R,\Sigma}}(\lambda) = 1] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$.

Algorithm 12: Experiment $\mathsf{aNSlan}_{\mathcal{A},\Xi_{\mathsf{R},\Sigma}}$

```
Procedure $\mathsf{aNSlan}_{\mathcal{A},\Xi_{R,\Sigma}}(\lambda)$:
    $\mathcal{Q} = \emptyset,\ \mathcal{F} = \emptyset;$
    $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(1^\lambda), for\ i \in [1, N]\ ;$
    $\boldsymbol{vk} = \{\mathsf{vk}_1, \mathsf{vk}_2, ..., \mathsf{vk}_N\};$
    $(m, \tilde{\boldsymbol{vk}}, \sigma) \leftarrow \mathcal{A}^{O_{\mathsf{S}}, O_{\mathsf{pS}}, O_{\mathsf{Corrupt}}}(\boldsymbol{vk})$ /* $O_{\mathsf{S}}, O_{\mathsf{pS}}, O_{\mathsf{Corrupt}}$ are the same
        as Algorithm 9                                                  */
    $b_1 := \mathsf{Verify}(m, \tilde{\boldsymbol{vk}}, \sigma);$
    $b_2 := \{\star, \star, \sigma\} \notin \mathcal{Q};$
    $b_3 := \mathsf{Link}((m, \tilde{\boldsymbol{vk}}, \sigma), (\hat{m}, \hat{\boldsymbol{vk}}, \hat{\sigma})) = 1, \exists \{\hat{m}, \hat{\boldsymbol{vk}}, \hat{\sigma}\} \in \mathcal{Q};$
    $b_4 := (\boldsymbol{vk} \cap \hat{\boldsymbol{vk}} \cap (\tilde{\boldsymbol{vk}} \backslash \mathcal{F})) \neq \emptyset;$
    $return(b_1 = 1 \wedge b_2 = 1 \wedge b_3 = 1 \wedge b_4 = 1);$
```

Definition 14 (Pre-signature adaptability). *A* LRAS *scheme $\Xi_{R,\Sigma}$ satisfies pre-signature adaptability if for any $n \in \mathbb{N}$, any message $m \in \{0,1\}^*$, any statement/witness pair $(T, t) \in R$, any key pair $(\mathsf{vk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$, any set of verification keys $\boldsymbol{vk} = \{\mathsf{vk}_1, ..., \mathsf{vk}_N\}$ and any pre-signature $\hat{\sigma} \leftarrow \{0,1\}^*$ with* $\mathsf{PreVerify}(m, \boldsymbol{vk}, T, \hat{\sigma}) = 1$*, we have* $\Pr[\mathsf{Verify}(m, \boldsymbol{vk}, \mathsf{Adapt}((T, t), \boldsymbol{vk}, \hat{\sigma}, M)) = 1] = 1$.

Definition 15 (Witness extractability). *A linkable ring adaptor signature scheme $\Xi_{R,\Sigma}$ is witness extractable if for every PPT adversary $\mathcal{A}$ running the experiment* $\mathsf{aWitExt}_{\mathcal{A},\Xi_{R,\Sigma}}$ *defined in Algorithm 13,* $\Pr[\mathsf{aWitExt}_{\mathcal{A},\Xi_{R,\Sigma}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Algorithm 13: Experiment $\mathsf{aWitExt}_{\mathcal{A},\Xi_{R,\Sigma}}$

```
Procedure $\mathit{aWitExt}_{\mathcal{A},\Xi_{R,\Sigma}}(\lambda)$:
    $\mathcal{Q} = \emptyset;$
    $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(1^\lambda), for\ i \in [1, N]\ ;$
    $\boldsymbol{vk} = \{\mathsf{vk}_1, \mathsf{vk}_2, ..., \mathsf{vk}_N\};$
    $(m^*, i^*, Y^*, \tilde{\boldsymbol{vk}}) \leftarrow \mathcal{A}^{O_{\mathsf{S}}, O_{\mathsf{pS}}}(\boldsymbol{vk})$ /* $O_{\mathsf{S}}, O_{\mathsf{pS}}$ are the same as
        Algorithm 9                                                     */
    $\hat{\sigma} \leftarrow \mathit{PreSign}(\mathsf{sk}_{i^*}, m^*, \tilde{\boldsymbol{vk}}, Y^*);$
    $\sigma \leftarrow \mathcal{A}^{O_{\mathsf{S}}, O_{\mathsf{pS}}}(\hat{\sigma});$
    $t' := \mathsf{Ext}(Y^*, \sigma, \hat{\sigma});$
    $return\ (m^* \notin \mathcal{Q} \wedge (Y^*, y') \notin R \wedge \mathsf{Verify}(m^*, \tilde{\boldsymbol{vk}}, \sigma) = 1);$
```

5.3 Generic Construction of LRAS

We first give a definition of Type-T Linkable Ring Canonical Identification in Algorithm 14. Then Type-TA Linkable Ring Canonical Identification can be defined in the same way as the previous section.

Algorithm 14: Type-T Linkable Ring Canonical Identification

```
1  Procedure SETUP($\lambda$):
2      return $\mathsf{param}$;
3  Procedure KEYGEN():
4      return $(\mathsf{pk}, \mathsf{sk})$;
5  Procedure PROOF1($\mathsf{sk}, \{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$):
6      $r \leftarrow_s \Delta_r$;
7      $R = A(r, \{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\})$;
8      return $(R, r)$;
9  Procedure CH($R$):
10     return $c$;
11 Procedure PROOF2($\mathsf{sk}, \{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}, r, c$):
12     return $z = Z(\mathsf{sk}, \{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}, r, c)$;
13 Procedure VERIFY($\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}, z, c$):
14     $R' = V(\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}, c, z)$;
15     if $c \neq \mathrm{CH}(R')$ then
16         return 0;
17     auxiliary checking with $R', \{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}, c, z$;
18     return 1;
19 Procedure LINK($\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}, (R_1, c_1, z_1), (R_2, c_2, z_2)$):
20     return 1 for linked, 0 for unlinked, or $\perp$ for any invalid transcript;
```

Then, we can build a generic construction of LRAS in Algorithm 15. The security of the generic construction of LRAS can be reduced to the underlying Type-TA linkable ring identification scheme or the generic adaptor signature. The security proofs are almost the same as the proofs of the generic adaptor signature, and hence they are omitted.

Observe that the linkable ring signature schemes in RingCT [15] and RingCT3.0 [21] are both Type-TA linkable ring canonical identification. Hence we can construct a concrete linkable ring adaptor signature similarly. Details are given in the Appendix A.1.

6 Conclusion

Adaptor signatures are a novel cryptographic primitive with important applications for cryptocurrencies. In this paper, we propose the first generic construction of adaptor signature. It can be combined with a number of different cryptographic protocols, such as blind adaptor signature and linkable ring adaptor signature. An interesting open question is that whether we can give a more generalized version of adaptor signature that can include non-Type-T signatures such as ECDSA. We leave it as the future work.

References

1. Abdalla, M., An, J.H., Bellare, M., Namprempre, C.: From identification to signatures via the fiat-shamir transform: Necessary and sufficient conditions for security and forward-security. IEEE Trans. Information Theory **54**(8), 3631–3646 (2008)
2. Abe, M., Ohkubo, M., Suzuki, K.: 1-out-of-n signatures from a variety of keys. In: Zheng, Y. (ed.) ASIACRYPT 2002. Lecture Notes in Computer Science, vol. 2501, pp. 415–432. Springer (2002)

3. Aumayr, L., Ersoy, O., Erwig, A., Faust, S., Hostakova, K., Maffei, M., Moreno-Sanchez, P., Riahi, S.: Generalized bitcoin-compatible channels. Cryptology ePrint Archive, Report 2020/476 (2020), `https://eprint.iacr.org/2020/476`
4. Aumayr, L., Ersoy, O., Erwig, A., Faust, S., Hostakova, K., Maffei, M., Moreno-Sanchez, P., Riahi, S.: Generalized bitcoin-compatible channels. IACR Cryptol. ePrint Arch. **2020**, 476 (2020)
5. Benhamouda, F., Lepoint, T., Loss, J., Orrù, M., Raykova, M.: On the (in)security of ros. Cryptology ePrint Archive, Report 2020/945 (2020), `https://eprint.iacr.org/2020/945`
6. Bernstein, D.J., Duif, N., Lange, T., Schwabe, P., Yang, B.: High-speed high-security signatures. In: Preneel, B., Takagi, T. (eds.) CHES 2011. Lecture Notes in Computer Science, vol. 6917, pp. 124–142. Springer (2011)
7. Esgin, M.F., Ersoy, O., Erkin, Z.: Post-quantum adaptor signatures and payment channel networks. In: Chen, L., Li, N., Liang, K., Schneider, S.A. (eds.) ESORICS 2020. Lecture Notes in Computer Science, vol. 12309, pp. 378–397. Springer (2020)
8. Fournier, L.: One-time verifiably encrypted signatures a.k.a. adaptor signatures. https://github.com/LLFourn/one-time-VES/blob/master/main.pdf (October 2019) (2019)
9. Fuchsbauer, G., Plouviez, A., Seurin, Y.: Blind schnorr signatures and signed elgamal encryption in the algebraic group model. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. Lecture Notes in Computer Science, vol. 12106, pp. 63–95. Springer (2020)
10. Guillou, L.C., Quisquater, J.: A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In: Goldwasser, S. (ed.) CRYPTO '88. Lecture Notes in Computer Science, vol. 403, pp. 216–231. Springer (1988)
11. Katz, J., Wang, N.: Efficiency improvements for signature schemes with tight security reductions. In: Jajodia, S., Atluri, V., Jaeger, T. (eds.) CCS 2003. pp. 155–164. ACM (2003)
12. Liu, J.K., Wei, V.K., Wong, D.S.: Linkable spontaneous anonymous group signature for ad hoc groups (extended abstract). In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. Lecture Notes in Computer Science, vol. 3108, pp. 325–335. Springer (2004)
13. Malavolta, G., Moreno-Sanchez, P., Schneidewind, C., Kate, A., Maffei, M.: Anonymous multi-hop locks for blockchain scalability and interoperability. In: 26th Annual Network and Distributed System Security Symposium, NDSS 2019 (2019)
14. Malavolta, G., Moreno-Sanchez, P., Schneidewind, C., Kate, A., Maffei, M.: Anonymous multi-hop locks for blockchain scalability and interoperability. In: NDSS 2019. The Internet Society (2019)
15. Noether, S.: Ring signature confidential transactions for monero. Cryptology ePrint Archive, Report 2015/1098 (2015), `https://eprint.iacr.org/2015/1098`
16. Poelstra, A.: Scriptless scripts (2017)
17. Schnorr, C.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO '89. Lecture Notes in Computer Science, vol. 435, pp. 239–252. Springer (1989)
18. Schnorr, C.P.: Efficient signature generation by smart cards. J. Cryptology **4**, 161–174 (1991)
19. Thyagarajan, S.A.K., Malavolta, G.: Lockable signatures for blockchains: Scriptless scripts for all signatures
20. Yi, X., Lam, K.: A new blind ECDSA scheme for bitcoin transaction anonymity. In: Galbraith, S.D., Russello, G., Susilo, W., Gollmann, D., Kirda, E., Liang, Z. (eds.) AsiaCCS 2019. pp. 613–620. ACM (2019)

21. Yuen, T.H., Sun, S., Liu, J.K., Au, M.H., Esgin, M.F., Zhang, Q., Gu, D.: Ringct 3.0 for blockchain confidential transaction: Shorter size and stronger security. In: Bonneau, J., Heninger, N. (eds.) FC 2020. Lecture Notes in Computer Science, vol. 12059, pp. 464–483. Springer (2020)

A Details of Linkable Ring Adaptor Signatures

A.1 Concrete Instantiation

We give the linkable ring adaptor signature in Algorithm 16 and 17. It is based on the ring signature in [21]. The **Verify** protocol is the same as that in [21], which is very similar to the **PreVerify** and hence it is omitted. Note that $\vec{y}^n = (1, y, y^2, \ldots, y^{n-1})$ for some integer y.

Security.

Lemma 1 (aEUF-CMA security). *Assuming that DL assumption holds and L_R is a hard relation, the linkable ring adaptor signature scheme $\Xi_{R,\Sigma}$ is* aEUF-CMA *secure in the random oracle model.*

Lemma 2 (Pre-signature anonymity). *If DDH assumption is hard, then the linkable ring adaptor signature scheme $\Xi_{R,\Sigma}$ is anonymous in the random oracle model.*

Lemma 3 (Linkability). *Assuming that DL assumption is hard, then the linkable ring adaptor signature $\Xi_{R,\Sigma}$ is linkable w.r.t. insider corruption in the random oracle model.*

Lemma 4 (Non-slanderability). *Assuming that DL assumption is hard, then the linkable ring adaptor signature $\Xi_{R,\Sigma}$ is non-slanderable w.r.t. insider corruption in the random oracle model.*

Lemma 5 (Pre-signature adaptability). *The linkable ring adaptor signature $\Xi_{R,\Sigma}$ satisfies pre-signature adaptability w.r.t. the relation L_R.*

Lemma 6 (Witness extractability). *Assuming that DL assumption is hard, then the linkable ring adaptor signature $\Xi_{R,\Sigma}$ is witness extractable in the random oracle model.*

Algorithm 6: Experiment $\mathsf{omaSignForge}_{\mathcal{A},\prod_{R,\Sigma}}$

```
Procedure $\mathit{omaSignForge}_{\mathcal{A},\Xi_{R,\Sigma}}(\lambda)$:
    $\mathcal{S}_s := \emptyset$, $\mathcal{S}_p := \emptyset$, $k_{s1} := 0$, $k_{p1} := 0$, $k_2 := 0$;
    $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$;
    $(M_1^*, \ldots, M_n^*) \leftarrow \mathcal{A}^{O_\mathsf{S}, O_\mathsf{pS}}(\mathsf{pk})$ ;
    $(Y, y) \leftarrow \mathsf{LockGen}(\lambda)$;
    $\hat{\sigma}_i \leftarrow \mathsf{PreSign}((\mathsf{pk}, \mathsf{sk}), Y, M_i^*)$ $\forall i \in [1, n]$ ;
    $(\sigma_1^*, \ldots, \sigma_n^*) \leftarrow \mathcal{A}^{O_\mathsf{S}, O_\mathsf{pS}}(\hat{\sigma}_1, \ldots, \hat{\sigma}_n, Y)$;
    return $(k_2 < n \wedge (M_i^*, \sigma_i^*) \neq (M_j^*, \sigma_j^*)$ $\forall i \neq j \in [1, n]$
        $\wedge \mathsf{Verify}(\mathsf{pk}, M_i^*, \sigma_i^*) = 1$ $\forall i \in [1, n]$ );
Procedure $O_\mathsf{S}(M, i, j)$:
    if $i = 1$ then
        $k_{s1} = k_{s1} + 1$;
        $(M', st_{k_{s1},1}) \leftarrow \mathsf{Sign}_1(\mathsf{sk}, M)$;
        $\mathcal{S}_s = \mathcal{S}_s \cup \{k_{s1}\}$;
        return $(k_{s1}, M')$;
    if $i = N_s$ then
        if $j \notin \mathcal{S}_s$ then
            return $\perp$;
        $(M', b) \leftarrow \mathsf{Sign}_{N_s}(st_{j,N_s}, M)$;
        if $b = 1$ then
            $\mathcal{S}_s = \mathcal{S}_s \setminus \{j\}$;
            $k_2 = k_2 + 1$;
        return $M'$;
    if $i \in [2, N_s - 1]$ then
        if $j \notin \mathcal{S}_s$ then
            return $\perp$;
        $(M', st_{j,i}) \leftarrow \mathsf{Sign}_i(st_{j,i-1}, M)$;
        return $M'$;
    return $\perp$;
Procedure $O_\mathsf{pS}(M, Y, i, j)$:
    if $i = 1$ then
        $k_{p1} = k_{p1} + 1$;
        $(M', st_{k_{p1},1}) \leftarrow \mathsf{PreSign}_1(\mathsf{sk}, Y, M)$;
        $\mathcal{S}_p = \mathcal{S}_p \cup \{k_{p1}\}$;
        return $(k_{p1}, M')$;
    if $i = N_p$ then
        if $j \notin \mathcal{S}_p$ then
            return $\perp$;
        $(M', b) \leftarrow \mathsf{PreSign}_{N_p}(st_{j,N_s}, M)$;
        if $b = 1$ then
            $\mathcal{S}_p = \mathcal{S}_p \setminus \{j\}$;
            $k_2 = k_2 + 1$;
        return $M'$;
    if $i \in [2, N_s - 1]$ then
        if $j \notin \mathcal{S}_p$ then
            return $\perp$;
        $(M', st_{j,i}) \leftarrow \mathsf{PreSign}_i(st_{j,i-1}, M)$;
        return $M'$;
    return $\perp$;
```

Algorithm 7: Experiment $\mathsf{baWitExt}_{\mathcal{A},\prod_{R,\Sigma}}$

```
Procedure $\mathit{baWitExt}_{\mathcal{A},\prod_{R,\Sigma}}(\lambda)$:
    $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$;
    $(M^*,Y) \leftarrow \mathcal{A}^{O'_\mathsf{S},O'_\mathsf{pS}}(\mathsf{pk})$ /* $O'_\mathsf{S},O'_\mathsf{pS}$ are the same as $O_\mathsf{S},O_\mathsf{pS}$ in
        Algorithm 6 except that $O_\mathsf{S}(\cdot,N_s,\cdot),O_\mathsf{pS}(\cdot,\cdot,N_p,\cdot)$ are not
        allowed.                                                        */
    $\hat{\sigma} \leftarrow \mathsf{PreSign}(\mathsf{sk},Y) \leftrightarrow \mathsf{User}(\mathsf{pk},M^*,Y)$;
    $\sigma^* \leftarrow \mathcal{A}^{O_\mathsf{S},O_\mathsf{pS}}(\hat{\sigma},Y)$;
    $y^* \leftarrow \mathsf{Ext}(Y,\sigma^*,\hat{\sigma})$;
    return $((Y,y^*) \notin R \wedge \mathsf{Verify}(\mathsf{pk},\sigma^*,M^*) = 1)$;
```

Algorithm 15: Generic linkable ring adaptor signature from a Type-TA linkable ring identification scheme LRID for the language $L := \{Y : \exists y \in \Delta_r : Y = A(y)\}$.

```
Procedure $\mathsf{Setup}(\lambda)$:
    define hash $H : \{0,1\}^* \to \Delta_c$;
    $\mathsf{param}_I \leftarrow \text{LRID.Setup}(\lambda)$;
    return $(\mathsf{param}_I, H)$;
Procedure $\mathsf{KeyGen}()$:
    return LRID.KeyGen();
Procedure $\mathsf{PreSign}(\mathsf{sk},m,\vec{\mathsf{vk}},Y)$:
    $r \leftarrow \Delta_r$;
    $R = A(r,\vec{\mathsf{vk}}) \oplus_R Y$;
    $c = H(m,R,\vec{\mathsf{vk}})$;
    $\hat{z} = Z(\mathsf{sk},\vec{\mathsf{vk}},r,c)$;
    return $\hat{\sigma} = (\hat{z},c)$;
Procedure $\mathsf{PreVerify}(m,\vec{\mathsf{vk}},Y,\hat{\sigma})$:
    parse $\hat{\sigma} = (\hat{z},c)$;
    $R' = V(\vec{\mathsf{vk}},c,\hat{z}) \oplus_R Y$;
    if $c \neq H(M,R',\vec{\mathsf{vk}})$ then
        return 0;
    return 1;
Procedure $\mathsf{Adapt}((Y,y),\vec{\mathsf{vk}},\hat{\sigma},m)$ and
  $\mathsf{Ext}(Y,\hat{\sigma},\sigma)$:
    Same as Algorithm 5
```

Algorithm 16: Linkable Ring Adaptor Signature

```
Procedure Setup($\lambda$):
    pick random generators $g, u, G, H_1, \ldots, H_n, G' \in \mathbb{G}$ of prime order $p$,
      denote $\vec{H} = [H_1, \ldots, H_n]$;
    $\hat{H}$ is a hash function $\{0,1\}^* \to \mathbb{Z}_p$, $\hat{H}_G$ is a hash function $\{0,1\}^* \to \mathbb{G}$;
    return $\mathsf{param} = (\mathbb{G}, p, g, G, \vec{H}, \hat{H}, \hat{H}_G)$;
Procedure KGen(param):
    pick $\mathsf{sk} \in \mathbb{Z}_p$;
    $\mathsf{vk} = g^{\mathsf{sk}}$;
    return $(\mathsf{vk}, \mathsf{sk})$;
Procedure PreSign($\mathsf{sk}_j, m, \mathbf{vk} = \{\mathsf{vk}_1, \ldots, \mathsf{vk}_n\}, Y$):
    set $\vec{b}_L = [0, 0, \ldots, 1, \ldots, 0]$ /* "1" is at the $j$-th pos */
    set $\vec{b}_R = [-1, -1, \ldots, 0, \ldots, -1]$ /* "0" is at the $j$-th pos */
    $U = u^{\mathsf{sk}}$, $h = \hat{H}_G(\mathbf{vk})$;
    pick a random $\alpha, \beta, \rho, r_\alpha, r_{\mathsf{sk}}, \in \mathbb{Z}_p$ and random vectors $\vec{s}_L, \vec{s}_R \in \mathbb{Z}_p^n$;
    $B = h^\alpha \mathbf{vk}^{\vec{b}_L}$, $A = h^\beta \vec{H}^{\vec{b}_R}$, $S_1 = h^{r_\alpha} g^{r_{\mathsf{sk}}} Y$, $S_2 = h^\rho \mathbf{vk}^{\vec{s}_L} \vec{H}^{\vec{s}_R}$, $S_3 = u^{r_{\mathsf{sk}}}$;
    $\mathsf{str} = h||B||A||S_1||S_2||S_3$, $y = \hat{H}(1||\mathsf{str})$, $z = \hat{H}(2||\mathsf{str})$, $w = \hat{H}(3||\mathsf{str})$;
    $\vec{c}_L = \vec{b}_L - z \cdot \vec{1}^n$, $\vec{c}_R = \vec{y}^n \circ (w \cdot \vec{b}_R + wz \cdot \vec{1}^n) + z^2 \cdot \vec{1}^n$;
    $\vec{s}'_R = \vec{s}_R \circ \vec{y}^n$, $t_1 = \langle \vec{s}_L, \vec{c}_R \rangle + \langle \vec{c}_L, \vec{s}'_R \rangle \mod p$, $t_2 = \langle \vec{s}_L, \vec{s}'_R \rangle \mod p$;
    pick a random $\tau_1, \tau_2 \in \mathbb{Z}_p$;
    $T_1 = g^{t_1} h^{\tau_1}$, $T_2 = g^{t_2} h^{\tau_2}$;
    $x = \hat{H}(4||y||z||w||T_1||T_2||m)$;
    $\tau = \tau_1 x + \tau_2 x^2 \mod p$, $\mu = \alpha + \beta w + \rho x \mod p$;
    $z_\alpha = r_\alpha + \alpha x \mod p$, $z'_{\mathsf{sk}} = r_{\mathsf{sk}} + \mathsf{sk}_j x \mod p$;
    $\vec{l} = \vec{c}'_L + \vec{s}_L \cdot x$, $\vec{r} = \vec{c}'_R + \vec{s}'_R \cdot x$, $\zeta = \langle \vec{l}, \vec{r} \rangle$;
    Set $\vec{H'} = [H_1, H_2^{y^{-1}}, \ldots, H_n^{y^{-n+1}}]$;
    $P = \mathbf{vk}^{\vec{l}} \vec{H'}^{\vec{r}}$;
    $\pi \leftarrow \mathsf{NIPA.Proof}(\mathbf{vk}, \vec{H'}, P, \zeta; \vec{l}, \vec{r})$;
    return $\hat{\sigma} = (B, A, S_1, S_2, S_3, T_1, T_2, \tau, \mu, z_\alpha, z'_{\mathsf{sk}}, \zeta, \pi, U)$;
Procedure PreVerify($m, \mathbf{vk} = \{\mathsf{vk}_1, \ldots, \mathsf{vk}_n\}, Y, \hat{\sigma}$):
    parse $\hat{\sigma} = (B, A, S_1, S_2, S_3, T_1, T_2, \tau, \mu, z_\alpha, z'_{\mathsf{sk}}, \zeta, \pi, U)$;
    $h = \hat{H}_G(\mathbf{vk})$;
    $\mathsf{str} = h||B||A||S_1||S_2||S_3$, $y = \hat{H}(1||\mathsf{str})$, $z = \hat{H}(2||\mathsf{str})$, $w = \hat{H}(3||\mathsf{str})$;
    $x = \hat{H}(4||y||z||w||T_1||T_2||m)$;
    $\vec{H'} = [H_1, H_2^{y^{-1}}, \ldots, H_n^{y^{-n+1}}]$;
    $P = BA^w S_2^x \mathbf{vk}^{-z \cdot \vec{1}^n} \vec{H}^{wz} \vec{H'}^{z^2 \cdot \vec{1}^n} h^{-\mu}$;
    if $\mathsf{NIPA.Verify}(\mathbf{vk}, \vec{H'}, P, \zeta, \pi) = 1$ and $u^{z'_{\mathsf{sk}}} = S_3 U^x$ and
      $g^\zeta h^\tau = g^{z^2 + wz(1-z)\sum_{i=1}^n y^{i-1} - nz^3} T_1^x T_2^{x^2}$ and $h^{z_\alpha} g^{z'_{\mathsf{sk}}} Y = S_1 B^x$ then
        return 1;
    return 0;
Procedure Verify($m, \mathbf{vk} = \{\mathsf{vk}_1, \ldots, \mathsf{vk}_n\}, \sigma$):
    parse $\sigma = (B, A, S_1, S_2, S_3, T_1, T_2, \tau, \mu, z_\alpha, z_{\mathsf{sk}}, \zeta, \pi, U)$;
    Compute $h, y, z, w, c, \vec{H'}, P$ as PreVerify;
    if $\mathsf{NIPA.Verify}(\mathbf{vk}, \vec{H'}, P, \zeta, \pi) = 1$ and $u^{z_{\mathsf{sk}}} = S_3 U^x$ and
      $g^\zeta h^\tau = g^{z^2 + wz(1-z)\sum_{i=1}^n y^{i-1} - nz^3} T_1^x T_2^{x^2}$ and $h^{z_\alpha} g^{z_{\mathsf{sk}}} = S_1 B^x$ then
        return 1;
    return 0;
```

Algorithm 17: Linkable Ring Adaptor Signature (cont.)

```
1  Procedure $\mathsf{Link}((m_1, \tilde{\boldsymbol{vk}}_1, \sigma_1), (m_2, \tilde{\boldsymbol{vk}}_2, \sigma_2))$:
2      parse $\sigma_1 = (\ldots, U_1)$ and $\sigma_2 = (\ldots, U_2)$;
3      return 1 if $U_1 = U_2$ or 0 otherwise;
4  Procedure $\mathsf{Adapt}((Y, y), \mathbf{vk}, \hat{\sigma}, m)$:
5      parse $\hat{\sigma} = (B, A, S_1, S_2, S_3, T_1, T_2, \tau, \mu, z_\alpha, z'_{\mathsf{sk}}, \zeta, \pi, U)$;
6      $z_{\mathsf{sk}} = z'_{\mathsf{sk}} + t \mod p$;
7      return $\sigma = (B, A, S_1, S_2, S_3, T_1, T_2, \tau, \mu, z_\alpha, z_{\mathsf{sk}}, \zeta, \pi, U)$;
8  Procedure $\mathsf{Ext}(T, \sigma, \hat{\sigma})$:
9      retrieve $z_{\mathsf{sk}}$ from $\sigma$ and $z'_{\mathsf{sk}}$ from $\hat{\sigma}$;
10     $t = z_{\mathsf{sk}} - z'_{\mathsf{sk}} \mod p$;
11     if $T = g^t$ then
12         return $t$;
13     return $\perp$;
```