An extended abstract of this paper appears in ESORICS 2016. In the proceeding version, we have mistakenly removed the tag-based trapdoor functions from the re-randomizable tagging construction which was in the submission version. Some other minor mistakes have also been corrected.

Efficient Sanitizable Signatures without Random Oracles

(Full Version)

Russell W. F. Lai[1], Tao Zhang[1], Sherman S. M. Chow[1]*, and Dominique Schröder[2]

[1] Department of Information Engineering
The Chinese University of Hong Kong
Sha Tin, N.T., Hong Kong
{wflai, zt112, sherman}@ie.cuhk.edu.hk
[2] Chair for Applied Cryptography
Friedrich-Alexander University Erlangen-Nürnberg
Erlangen and Nuremberg, Bavaria, Germany
dosc@cs.fau.de

March 23, 2017

Abstract. Sanitizable signatures, introduced by Ateniese et al. (ESORICS '05), allow the signer to delegate the sanitization right of signed messages. The sanitizer can modify the message and update the signature accordingly, so that the sanitized part of the message is kept private. For a stronger protection of sensitive information, it is desirable that no one can link sanitized message-signature pairs of the same document. This idea was formalized by Brzuska et al. (PKC '10) as unlinkability, which was followed up recently by Fleischhacker et al. (PKC '16). Unfortunately, the existing generic constructions of sanitizable signatures, unlinkable or not, are based on building blocks with specially crafted features of which efficient (standard model) instantiations are absent. Basing on existing primitives or a conceptually simple primitive is more desirable.

In this work, we present two such generic constructions, leading to efficient instantiations in the standard model. The first one is based on rerandomizable tagging, a new primitive which may find independent interests. It captures the core accountability mechanism of sanitizable signatures. The second one is based on accountable ring signatures (CARDIS '04, ESORICS '15). As an intermediate result, we propose the first accountable ring signature scheme in the standard model.

1 Introduction

Regular signatures are non-malleable. It is infeasible to maul a valid message-signature pair (m, σ) into a modified pair (m', σ') that passes the verification. However, a controlled form of malleability can be desirable in many settings, such as research study on sanitized Internet traffic or anonymized medical data, commercial usages that replace advertisements in authenticated media streams, or updates of reliable routing information [ACdT05]. Sanitizable signatures, introduced by Ateniese *et al.* [ACdT05], support controlled malleability. The signer can specify parts of a (signed) message which a designated third party, called the sanitizer, can change and then adapt the signature accordingly. Brzuska *et al.* [BFF+09] formalized five security properties, including privacy which states that the sanitized part of the message cannot be recovered from a sanitized signature. A strictly stronger property, called unlinkability, was suggested one year later [BFLS10]. Unlinkability ensures that one cannot link sanitized message-signature pairs of the same document. It is particularly important in the motivating applications which sanitize data for privacy [ACdT05] as it prevents the attacker from combining information of several sanitized versions of a document for reconstructing (parts of) the original document. Such linkage is useful for de-anonymization.

Unlinkable sanitizable signatures was then constructed [BFLS10] from group signatures having the specific property that the keys of the signers can be computed independently even before seeing the keys of the group manager. Although this property was defined in the seminal work of Bellare, Micciancio, and Warinshi [BMW03], in a typical application of group signature, the group is formed first and the signers join the group later. This order is even exploited for gaining efficiency in building group signature scheme

* Corresponding Author

via the notion of certified signatures [Gro07]. In a very recent study of Fleischhacker *et al.* [FKM$^+$16], to instantiate the generic construction of Brzuska *et al.* [BFLS10], they need to use an *inefficient* scheme based on the random oracle model (ROM) and generic group model (GGM) [FY05], or look into the details of the scheme [Gro07] and perform the adaption accordingly to fit with the special requirement. This diminishes the benefits of a generic construction. Although the scheme [Gro07] is proven in the standard model without random oracles, the proof requires the adversary to only perform group operations on the given elements (generic group model or GGM). No existing simple assumption supports the proof. Their study suggested that, to this date, no efficient group signature scheme *that has the required properties* is known, which also means that no efficient unlinkable sanitizable signature scheme is known. In response, they gave another generic construction from signatures with re-randomizable keys, which is very efficient when instantiated with Schnorr signature, yet with security argued with the ROM heuristics. Unfortunately, the re-randomizable keys property is also an unusual property, as showcased by the original authors [FKM$^+$16] that two pairing-based short signature schemes cannot serve as a building block. This leaves limited and unsatisfactory choices of schemes:

- having a subset of the security properties [ACdT05, BFF$^+$09],
- relying on the ROM [FKM$^+$16],
- secure without ROM, but building upon inefficient construction [BFLS10].

1.1 Our Contribution

Our main result is closing the research gap, presenting the first efficient (unlinkable) sanitizable signature schemes which are secure in the standard model. In fact, we propose two very different generic constructions which are both simple. Our study also gives several new results that are of independent interests.

Generic Constructions. We propose two fundamentally different generic constructions. The first generic construction is based on *rerandomizable tagging*, a new notion which may find independent applications. Indeed, it can be considered as a dual notion of double-trapdoor anonymous tag [ACHO13], a primitive proven to be useful for privacy-oriented authorship management mechanism. In particular, using it in a generic construction of traceable signature schemes allows the signer (or the group manager on behalf) to deny the authorship of a signature [ACHO13].

While both our tags and the public-keys expected by the signature scheme required in the previous generic construction [FKM$^+$16] are “re-randomizable”, we believe that our formulation captures the essential functionality to achieve accountability, for both creation and sanitization. This leads to our conceptually simple generic construction, in which the rerandomizable tagging scheme takes care of the accountability, and regular signature schemes for the signing functionality. Using only basic primitives and our new rerandomizable tagging *without any zero-knowledge proofs*, this construction is very efficient and achieves privacy, in the standard model and under only the relatively simpler static assumptions.

Our second generic construction, which achieves unlinkability, is based on *accountable ring signatures* [XY04]. In contrast to the existing generic construction from group signatures [BFLS10], where the latter is required to satisfy some special property, our construction relies on an existing notion which can be used as-is. One can immediately instantiate our construction by a recent scheme [BCC$^+$15], which yields an efficient unlinkable sanitizable signature scheme in the ROM. As an extra feature, this generic construction naturally supports *multiple sanitizers* [CJL12].

Accountable Ring Signatures in the Standard Model. Aiming at constructing unlinkable sanitizable signatures in the standard model, we also construct the first accountable ring signature scheme in the standard model. The assumption required by this scheme is a q-type assumption due to the membership proof [BDR15]. Our scheme inherits the constant signature size (with respect to number of members in the ring) from non-accountable schemes in the literature [BDR15]. The existing scheme [BCC$^+$15] only relies on the (static) decisional Diffie-Hellman assumption yet requires a logarithmic signature size. Due to the existing results [BCC$^+$15, BCC$^+$16], it also leads to a constant-size instantiation of a strong variant of fully dynamic group signatures, in which group manager not only can enroll, but also revoke group members.

1.2 Related Work

Ateniese *et al.* [ACdT05] informally describe the following properties of sanitizable signatures. *Unforgeability* says that signatures can only be created by honest signers and sanitizers. *Immutability* demands only designated parts of the message can be modified by the (malicious) sanitizer. *Transparency* ensures the indistinguishability of signatures computed by the signer and the sanitizer (or more precisely, they are indistinguishable to *public verifiers*, *i.e.*, anyone other than the signer and the sanitizer themselves). *Accountability* means that neither the malicious signer nor the malicious sanitizer can deny authorship of the message. When the need arises, the signer can generate a *proof of authorship*.

These requirements were formalized by Brzuska *et al.* [BFF$^+$09]. Since then, many works formalize various other properties. Note that transparency ensures that any public verifier cannot even notice if the message has been sanitized. *Unlinkability*, introduced by Brzuska *et al.* [BFLS10], takes a step further in which a sanitized signature cannot be linked to its original version. This is crucial for privacy.

It is tricky to get a right balance of accountability and transparency. Canard *et al.* [CLM08] addressed the lack of accountability in the seminal work [ACdT05], yet at the cost of transparency. On the other hand, unconditional transparency is often undesirable, which motivates the need for accountability. The original accountability notion [ACdT05, BFF$^+$09] is interactive since it needs the participation of the signer. A non-interactive version was later proposed [BPS12], which allows a third party to determine if a message originates from the signer or the sanitizer, without any help from the signer. Nevertheless, non-interactive accountability and transparency cannot be achieved simultaneously [FKM$^+$16], so we focus on schemes that have (interactive) accountability and transparency.

Holding the sanitizer accountable is a measure after the fact. Another idea is to limit the allowable sanitization [KL06, CJ10]. However, unlinkability in this setting is even more complicated. For instance, one may want to also conceal the sets of allowed modifications [BPS13]. Yet, it appears to be difficult to construct such a scheme efficiently. Recently, Derler and Slamanig [DS15] suggested an intermediate notion (weaker than unlinkability but stronger than privacy) as a compromise for achieving efficient construction. We remark that Canard *et al.* [CJL12] considered multiple signers and sanitizers, with construction based on group signatures.

Malleable signatures were considered in many variations, such as homomorphic signatures [JMSW02, Cat14], which allows public evaluation of functions on more than one signed messages, or redactable signatures [JMSW02, BBD$^+$10], which allows parts of the message to be removable. They aim to solve related but different problems, and are not directly applicable in our motivating scenarios as discussed [ACdT05, BFF$^+$09, BFLS10, FKM$^+$16].

Delegation of signing right is considered in proxy signatures [BPW12]. Yet, the signatures produced by the proxy are often publicly distinguishable from signatures created by the designator, which violates the transparency property of sanitizable signatures. Recent advances such as (delegatable) functional signatures [BMS16] associate the signing right with a policy specifying which messages can be signed, or even arbitrary functions to be applied on the key and the messages, such that the policy or the function remain hidden. These works show theoretical solutions, but are too slow for practical use.

2 Rerandomizable Tagging Schemes

At a high level, the core of a sanitizable signature is a cryptographic object which is computed by the signer with some secret information embedded. This object can can be rerandomized by the sanitizer many times in an indistinguishable way. In addition, when the sanitizer changes the object, it will no longer match with the embedded secret, indicating that the signature is sanitized.

To capture the above functionality, we introduce a new primitive called rerandomizable tagging. In a rerandomizable tagging scheme, the tag issuer $\mathtt{I}$ generates a tag τ using its private key $\mathsf{sk}_\mathtt{I}$ with respect to a user public key $\mathsf{pk}_\mathtt{U}$. The user $\mathtt{U}$ can then use its own private key $\mathsf{sk}_\mathtt{U}$ to rerandomize the tag which looks indistinguishable from the one issued by the issuer. When necessary, however, the tag issuer can generate a proof π to claim or deny the authorship of a (rerandomized) tag.

2.1 Definition of Rerandomizable Tagging Schemes

In this section, we formalize the notion of rerandomizable tagging schemes.

Definition 1 (Rerandomizable Tagging Schemes). *A* rerandomizable tagging *scheme* $\mathcal{RT} = (\mathsf{TGen}_{\mathtt{I}}, \mathsf{TGen}_{\mathtt{U}}, \mathsf{Tag}, \mathsf{ReTag}, \mathsf{TVer}, \mathsf{TProv}, \mathsf{TJud})$ *consists of seven efficient algorithms:*

Key Generation. *The key generation algorithms for the issuer and the user respectively both create a public/private key pair:* $(\mathsf{pk}_{\mathtt{I}}, \mathsf{sk}_{\mathtt{I}}) \leftarrow \mathsf{TGen}_{\mathtt{I}}(1^\lambda)$*,* $(\mathsf{pk}_{\mathtt{U}}, \mathsf{sk}_{\mathtt{U}}) \leftarrow \mathsf{TGen}_{\mathtt{U}}(1^\lambda)$*.*

Tagging. *The tagging algorithm takes as input an issuer private key* $\mathsf{sk}_{\mathtt{I}}$*, a user public key* $\mathsf{pk}_{\mathtt{U}}$*, and a message* $m \in \{0,1\}^*$*. It outputs a tag* $\tau \leftarrow \mathsf{Tag}(\mathsf{sk}_{\mathtt{I}}, \mathsf{pk}_{\mathtt{U}}, m)$*.*

Re-Tagging. *The re-tagging algorithm takes as input the issuer public key* $\mathsf{pk}_{\mathtt{I}}$*, a user private key* $\mathsf{sk}_{\mathtt{U}}$*, two messages* $m, m' \in \{0,1\}^*$*, and a tag* τ*. It outputs a new tag* $\tau' \leftarrow \mathsf{ReTag}(\mathsf{pk}_{\mathtt{I}}, \mathsf{sk}_{\mathtt{U}}, m, m', \tau)$*.*

Verification. *The verification algorithm takes as input the issuer public key* $\mathsf{pk}_{\mathtt{I}}$*, a user public key* $\mathsf{pk}_{\mathtt{U}}$*, a message* $m \in \{0,1\}^*$*, and a tag* τ*. It outputs a bit* $b \leftarrow \mathsf{TVer}(\mathsf{pk}_{\mathtt{I}}, \mathsf{pk}_{\mathtt{U}}, m, \tau)$*.*

Proof. *The proof algorithm takes as input the issuer private key* $\mathsf{sk}_{\mathtt{I}}$*, a user public key* $\mathsf{pk}_{\mathtt{U}}$*, a message* $m \in \{0,1\}^*$*, and a tag* τ*. It outputs a proof* $\pi \leftarrow \mathsf{TProv}(\mathsf{sk}_{\mathtt{I}}, \mathsf{pk}_{\mathtt{U}}, m, \tau)$*.*

Judge. *The judge algorithm takes as input the issuer and user public keys* $\mathsf{pk}_{\mathtt{I}}, \mathsf{pk}_{\mathtt{U}}$*, a message* $m \in \{0,1\}^*$*, a tag* τ*, and a proof* π*. It outputs a decision* $d \in \{\mathtt{I}, \mathtt{U}\}$ *indicating whether the tag was created by the issuer or the user:* $d \leftarrow \mathsf{TJud}(\mathsf{pk}_{\mathtt{I}}, \mathsf{pk}_{\mathtt{U}}, m, \tau, \pi)$*.*

We define correctness of rerandomizable tagging as follows:

Definition 2 (Correctness). *A rerandomizable tagging scheme* $\mathcal{RT}$ *is* correct *if, for all parameters* $\lambda \in \mathbb{N}$*, for all messages* $m, m' \in \{0,1\}^*$*, for all keys generated from* $(\mathsf{pk}_{\mathtt{I}}, \mathsf{sk}_{\mathtt{I}}) \leftarrow \mathsf{TGen}_{\mathtt{I}}(1^\lambda)$ *and* $(\mathsf{pk}_{\mathtt{U}}, \mathsf{sk}_{\mathtt{U}}) \leftarrow \mathsf{TGen}_{\mathtt{U}}(1^\lambda)$*, for all tags generated from* $\tau \leftarrow \mathsf{Tag}(\mathsf{sk}_{\mathtt{I}}, \mathsf{pk}_{\mathtt{U}}, m)$ *and* $\tau' \leftarrow \mathsf{ReTag}(\mathsf{pk}_{\mathtt{I}}, \mathsf{sk}_{\mathtt{U}}, m, m', \tau)$*, it holds that* $\mathsf{TVer}(\mathsf{pk}_{\mathtt{I}}, \mathsf{pk}_{\mathtt{U}}, m, \tau) = 1$ *and* $\mathsf{TVer}(\mathsf{pk}_{\mathtt{I}}, \mathsf{pk}_{\mathtt{U}}, m', \tau') = 1$*. Furthermore, for all proofs generated from* $\pi \leftarrow \mathsf{TProv}(\mathsf{sk}_{\mathtt{I}}, \mathsf{pk}_{\mathtt{U}}, m, \tau)$ *and* $\pi' \leftarrow \mathsf{TProv}(\mathsf{sk}_{\mathtt{I}}, \mathsf{pk}_{\mathtt{U}}, m', \tau')$*, it holds that* $\mathsf{TJud}(\mathsf{pk}_{\mathtt{I}}, \mathsf{pk}_{\mathtt{U}}, m, \tau, \pi) = \mathtt{I}$ *and* $\mathsf{TJud}(\mathsf{pk}_{\mathtt{I}}, \mathsf{pk}_{\mathtt{U}}, m', \tau', \pi') = \mathtt{U}$*.*

2.2 Security of Rerandomizable Tagging Schemes

Rerandomizable tagging schemes abstract the core properties of sanitizable signatures. Therefore, their security properties, namely, (proof-restricted) privacy, accountability, and (proof-restricted) transparency, follow from the corresponding ones of sanitizable signatures [BFF+09]. For sanitizable signatures, (proof-restricted) transparency implies (proof-restricted) privacy. We therefore omit the definition of the latter.

Accountability. This property demands that the origin of a (possibly rerandomized) tag should be undeniable. We distinguish between *issuer-accountability* and *user-accountability*. The former says that, if a tag has not been rerandomized, then a malicious issuer cannot make the judge accuse the user. In the issuer-accountability game, a malicious issuer $\mathcal{A}_{\mathsf{Tag}}$ gets a user public key $\mathsf{pk}_{\mathtt{U}}$ as input and has access to a re-tagging oracle, which takes as input tuples $(\mathsf{pk}_{\mathtt{I},i}, m_i, m'_i, \tau_i)$ and returns τ'_i. Eventually, $\mathcal{A}_{\mathsf{Tag}}$ outputs a tuple $(\mathsf{pk}^*_{\mathtt{I}}, m^*, \tau^*, \pi^*)$ and wins the game if TJud accuses the user of the new key $\mathsf{pk}^*_{\mathtt{I}}$ with a valid tag τ^* on the message m^*.

Definition 3 (Issuer-Accountability). *A rerandomizable tagging scheme* $\mathcal{RT}$ *is* issuer-accountable *if, for all* PPT *adversaries* $\mathcal{A}_{\mathsf{Tag}}$*, the probability that the experiment* $\mathsf{Iss\text{-}Acc}^{\mathcal{RT}}_{\mathcal{A}_{\mathsf{Tag}}}(\lambda)$ *outputs* 1 *is negligible (in* λ*), where*

Experiment $\mathsf{Iss\text{-}Acc}^{\mathcal{RT}}_{\mathcal{A}_{\mathsf{Tag}}}(\lambda)$
  $(\mathsf{pk}_{\mathtt{U}}, \mathsf{sk}_{\mathtt{U}}) \leftarrow \mathsf{TGen}_{\mathtt{U}}(1^\lambda)$*;* $(\mathsf{pk}^*_{\mathtt{I}}, m^*, \tau^*, \pi^*) \leftarrow \mathcal{A}_{\mathsf{Tag}}^{\mathsf{ReTag}(\cdot, \mathsf{sk}_{\mathtt{U}}, \cdot, \cdot, \cdot)}(\mathsf{pk}_{\mathtt{U}})$
    where $(\mathsf{pk}_{\mathtt{I},i}, m_i, m'_i, \tau_i)$ *and* τ'_i *denote the queries and answers to and from oracle* ReTag*.*
  Output 1 *if for all* i *the following holds:*
    $(\mathsf{pk}^*_{\mathtt{I}}, m^*) \neq (\mathsf{pk}_{\mathtt{I},i}, m'_i) \;\wedge\; \mathsf{TVer}(\mathsf{pk}^*_{\mathtt{I}}, \mathsf{pk}_{\mathtt{U}}, m^*, \tau^*) = 1 \;\wedge\; \mathsf{TJud}(\mathsf{pk}^*_{\mathtt{I}}, \mathsf{pk}_{\mathtt{U}}, m^*, \tau^*, \pi^*) \neq \mathtt{I}$
  else output 0*.*

In the user-accountability game, $\mathcal{A}_{\mathsf{ReTag}}$ models a malicious user with access to Tag and TProv oracles. It succeeds if it outputs a key $\mathsf{pk}^*_{\mathtt{U}}$, a message m^*, and a tag τ^*, such that $(\mathsf{pk}^*_{\mathtt{U}}, m^*)$ is different from $(\mathsf{pk}_{\mathtt{U},i}, m_i)$ previously queried to the Tag oracle. Moreover, the proof π^* produced by the issuer via TProv is required to lead the judge to decide "I", *i.e.*, the tag was created by the issuer.

Definition 4 (User-Accountability). *A rerandomizable tagging scheme* $\mathcal{RT}$ *is* user-accountable *if, for all* PPT *adversaries* $\mathcal{A}_{\mathsf{ReTag}}$*, the probability that the experiment* $\mathsf{Usr\text{-}Acc}^{\mathcal{RT}}_{\mathcal{A}_{\mathsf{ReTag}}}(\lambda)$ *evaluates to* 1 *is negligible (in* λ*), where*

```
***Experiment*** $\mathsf{Usr\text{-}Acc}^{\mathcal{RT}}_{\mathcal{A}_{\mathsf{ReTag}}}(\lambda)$
  $(\mathsf{pk}_{\mathtt{I}}, \mathsf{sk}_{\mathtt{I}}) \leftarrow \mathsf{TGen}_{\mathtt{I}}(1^\lambda)$; $(\mathsf{pk}^*_{\mathtt{U}}, m^*, \tau^*) \leftarrow \mathcal{A}_{\mathsf{ReTag}}^{\mathsf{Tag}(\mathsf{sk}_{\mathtt{I}},\cdot,\cdot),\mathsf{TProv}(\mathsf{sk}_{\mathtt{I}},\cdot,\cdot,\cdot)}(\mathsf{pk}_{\mathtt{I}})$
      *where* $(\mathsf{pk}_{\mathtt{U},i}, m_i)$ *and* $\tau_i$ *denote the queries and answers of oracle* $\mathsf{Tag}$.
  $\pi \leftarrow \mathsf{TProv}(\mathsf{sk}_{\mathtt{I}}, \mathsf{pk}^*_{\mathtt{U}}, m^*, \tau^*)$
  *Output* 1 *if for all* $i$ *the following holds:*
      $(\mathsf{pk}^*_{\mathtt{U}}, m^*) \neq (\mathsf{pk}_{\mathtt{U},i}, m_i) \ \wedge\ \mathsf{TVer}(\mathsf{pk}_{\mathtt{I}}, \mathsf{pk}^*_{\mathtt{U}}, m^*, \tau^*) = 1 \ \wedge\ \mathsf{TJud}(\mathsf{pk}_{\mathtt{I}}, \mathsf{pk}^*_{\mathtt{U}}, m^*, \tau^*, \pi) \neq \mathtt{U}$
  *else output* 0.
```

Transparency. This property says that one cannot decide if a tag has been rerandomized or not. Formally, this is defined in a game where an adversary $\mathcal{A}$ has access to Tag, ReTag, and TProv oracles to create (rerandomized) tags and learn the proofs. In addition, $\mathcal{A}$ gets access to a $\mathsf{Tag}/\mathsf{ReTag}_b(\cdot,\cdot)$ oracle with a secret random bit $b \in \{0,1\}$ embedded which, on input a messages m and m', behaves as follows:

- for $b = 0$ runs the tagging algorithm to create $\tau \leftarrow \mathsf{Tag}(\mathsf{sk}_{\mathtt{I}}, \mathsf{pk}_{\mathtt{U}}, m)$, then runs the re-tagging algorithm $\tau' \leftarrow \mathsf{ReTag}(m, m', \mathsf{pk}_{\mathtt{I}}, \mathsf{sk}_{\mathtt{U}}, \tau)$ and returns the rerandomized tag τ';
- for $b = 1$ runs the tagging algorithm to create $\tau' \leftarrow \mathsf{Tag}(\mathsf{sk}_{\mathtt{I}}, \mathsf{pk}_{\mathtt{U}}, m')$, then returns the tag τ'.

Eventually, the adversary $\mathcal{A}$ eventually produces an output a as a guess for b. A rerandomizable tagging is *transparent* if for all efficient algorithms $\mathcal{A}$ the probability for a right guess $a = b$ in the above game is negligibly close to $\frac{1}{2}$. Following [BFLS10], we define a relaxed version called *proof-restricted transparency*, where the attacker is not allowed to query the challenge messages received from the challenge oracle $\mathsf{Tag}/\mathsf{ReTag}_b(\cdot,\cdot)$.

Definition 5 ((Proof-Restricted) Transparency). *A rerandomizable tagging scheme* $\mathcal{RT}$ *is* proof-restrictedly transparent *if, for all* PPT *adversaries* $\mathcal{A}$*, the probability that the experiment* $\mathsf{Trans}^{\mathcal{RT}}_{\mathcal{A}}(\lambda)$ *returns* 1 *is negligibly close to* $\frac{1}{2}$ *(in* λ*).*

```
***Experiment*** $\mathsf{Trans}^{\mathcal{RT}}_{\mathcal{A}}(\lambda)$
  $(\mathsf{pk}_{\mathtt{I}}, \mathsf{sk}_{\mathtt{I}}) \leftarrow \mathsf{TGen}_{\mathtt{I}}(1^\lambda)$; $(\mathsf{pk}_{\mathtt{U}}, \mathsf{sk}_{\mathtt{U}}) \leftarrow \mathsf{TGen}_{\mathtt{U}}(1^\lambda)$; $b \leftarrow \{0,1\}$
  $a \leftarrow \mathcal{A}^{\mathsf{Tag}(\mathsf{sk}_{\mathtt{I}},\cdot,\cdot),\mathsf{ReTag}(\cdot,\mathsf{sk}_{\mathtt{U}},\cdot,\cdot,\cdot),\mathsf{TProv}(\mathsf{sk}_{\mathtt{I}},\cdot,\cdot,\cdot),\mathsf{Tag}/\mathsf{ReTag}_b(\cdot,\cdot)}(\mathsf{pk}_{\mathtt{I}}, \mathsf{pk}_{\mathtt{U}})$
  *Output* 1 *if* $\left(a = b \wedge M_{\mathsf{Tag}/\mathsf{ReTag}} \cap M_{\mathsf{TProv}} = \emptyset\right)$ *else output* 0
    *where* $M_{\mathsf{Tag}/\mathsf{ReTag}}$ *and* $M_{\mathsf{TProv}}$ *denote the sets of messages output from and*
    *queried to oracles* $\mathsf{Tag}/\mathsf{ReTag}_b$ *and* $\mathsf{TProv}$ *respectively.*
```

3 Construction of Rerandomizable Tagging Schemes

We describe a rerandomizable tagging construction based on double-trapdoor chameleon hashing [CDFG08], a variant of tag-based trapdoor functions to be defined below, and an extractable public key encryption scheme (Appendix A). A double-trapdoor chameleon hash function is a chameleon hash function [KR00] with an efficient algorithm which takes as input a pair of collisions and outputs one of the trapdoors. Its formal definition can be found in Appendix A.

3.1 Tag-based Trapdoor Functions

We define our required variant of tag-based trapdoor functions.

Definition 6 (Tag-based Trapdoor Functions). *A* tag-based trapdoor function *is a tuple of* PPT *algorithms* $\mathsf{TD} = (\mathsf{TDGen}, \mathsf{TDEval}, \mathsf{TDInv})$ *that are defined as follows:*

$\mathsf{TDGen}(1^\lambda)$**:** *The key generation algorithm returns a key-pair* $(\mathsf{pk}, \mathsf{sk})$.

$\mathsf{TDEval}(\mathsf{pk}, \mu, \rho)$**:** *The evaluation algorithm takes as input the public key* pk*, a tag* μ*, and a pre-image* ρ *in some domain* D*. It outputs an image* y.

$\mathsf{TDInv}(\mathsf{sk}, \mu, y)$**:** *The inversion algorithm takes as input the secret key* sk*, a tag* μ*, and an image* y*. It outputs a pre-image* ρ *such that* $y = \mathsf{TDEval}(\mathsf{pk}, \mu, \rho)$.

For our purpose, we need tag-based trapdoor functions where it is efficiently possible to sample elements from the domain and pre-image.

Definition 7 (Domain and Pre-image Sampling). *A tag-based trapdoor function* $\mathsf{TD} = (\mathsf{TDGen}, \mathsf{TDEval}, \mathsf{TDInv})$ *has an efficiently* samplable domain and pre-image space*, if there exists an efficiently samplable distribution* χ *over the domain* D*, such that* $\mathsf{TDInv}(\mathsf{sk}, \mu, y)$ *samples* ρ *from the conditional distribution of* χ *given* $y = \mathsf{TDEval}(\mathsf{pk}, \mu, \rho)$*.*

Security of Tag-based Trapdoor Functions In the following, we define a security property called *collision-resistance under selective-tag adaptive-image attacks*, which is inspired by the existential unforgeability under selective chosen message attack of digital signatures. Indeed, if the images y_i are all identical to some y and are determined by the challenger instead of the adversary, one can interpret (pk, y) as the public key of a signature scheme, and ρ_i as a signature of the message μ_i.

Definition 8 (Collision-Resistant under Selective-Tag Adaptive-Image Attacks). *A tag-based trapdoor function* $\mathsf{TD} = (\mathsf{TDGen}, \mathsf{TDEval}, \mathsf{TDInv})$ *has is* collision-resistant under selective-tag adaptive-image attacks *if for all* PPT *adversary* $\mathcal{A}$ *which makes any* Q *number of queries, the probability of it winning the following security game is negligible: The adversary* $\mathcal{A}$ *chooses* Q *distinct tags* $(\mu_1, \ldots, \mu_Q)$ *that it wishes to invert. The challenger* $\mathcal{C}$ *receives the tags, generates the key pair* $(\mathsf{pk}, \mathsf{sk})$*, and sends the public key* pk *to* $\mathcal{A}$*.* $\mathcal{A}$ *can adaptively choose images* y_i *from* $i = 1$ *to* Q*. Upon receiving* y_i*,* $\mathcal{C}$ *runs* $\rho_i \leftarrow \mathsf{TDInv}(\mathsf{sk}, \mu_i, y_i)$ *and sends* ρ_i *to* $\mathcal{A}$*. After answering all* Q *queries,* $\mathcal{A}$ *outputs* $((\mu_1^*, \rho_1^*), (\mu_2^*, \rho_2^*))$*. It wins the game if* $\mathsf{TDEval}(\mathsf{pk}, \mu_1^*, \rho_1^*) = \mathsf{TDEval}(\mathsf{pk}, \mu_2^*, \rho_2^*)$*,* $(\mu_1^*, \rho_1^*) \neq (\mu_2^*, \rho_2^*)$*, and* $\mu_1^*, \mu_2^* \notin (\mu_1, \ldots, \mu_Q)$*.*

Our variant of tag-based trapdoor functions can be constructed similar to the construction of a selectively secure signature scheme from lattice-based trapdoor functions [MP12, Section 6.2]. For completeness, we describe the construction in Appendix B.

3.2 Informal Description of the Construction of Rerandomizable Tagging Schemes

In our construction, the user public key mainly consists that of a tag-based trapdoor function and a tag τ of a message m of the randomness ρ_1 and ρ_2. The first randomness ρ_1 is for deriving a (random) hash value μ of the message m. The hash value μ is used as a tag and evaluated with ρ_2 in the trapdoor function to give an image y. The values μ and y are absent from the rerandomizable tag, but are implicitly fixed by the tuple (m, ρ_1, ρ_2).

To allow proving of the tag authorship later, the issuer prepares the randomness ρ_1 and ρ_2, and other auxiliary information, using the following procedures: It first obtains random seeds r_i for $i = 1, 2, 3$ by evaluating a pseudorandom function on random inputs q_i, then applies a pseudorandom generator on r_1 and r_2, and uses them as the randomness ρ_1 for the random hash and pre-image ρ_2 for the trapdoor function respectively. The last randomness r_3 is used for generating a ciphertext c of the message m. It then generates a signature σ for the tuple $(\mathsf{pk}_\mathsf{U}, y, q_1, q_2, q_3, c)$, and outputs the tag $\tau := (\rho_1, \rho_2, q_1, q_2, q_3, c, \sigma)$. Observe that only the values ρ_1 and ρ_2 can be changed by the user. In the case of dispute, the issuer can recover the q_i's and hence the r_i's from the tag, and use the r_i's as the proof of (non-)authorship. The pseudorandomness r_1, r_2, and r_3 allow the judge to compute the original pseudorandomness ρ_1 and ρ_2 using the pseudorandom generator, and extract the original message m from the ciphertext. It can thus verify the authorship of the tag. The extractability of the encryption scheme makes the proof algorithm "history-free".

To rerandomize a tag for a message m into a tag for another message m', the user first recovers the values μ and y from the tuple (m, ρ_1, ρ_2). It then computes a random digest μ' of m'. Finally, it uses the trapdoor to sample a pre-image ρ_2' of y with tag μ'.

The above approach almost works except an annoying problem: In the proof of issuer-accountability, the simulator must first commit to a set of μ_i's, and hope that the adversary forges a tag with respect to μ^* outside of this set, which it is not obliged to do so. To resolve this obstacle, we require that the string μ to be computed by a double-trapdoor chameleon hash function specified in the user public key. We note that the trapdoors for the chameleon hash function are not used except in the security proof. In this way, if the adversary comes up with a μ^* inside the set chosen by the simulator, the latter can recover the second trapdoor of the hash function and thus break the collision-resistance of the hash function.

3.3 Formal Description of the Construction of Rerandomizable Tagging Schemes

Let $F : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^\lambda$ be a pseudorandom function, $g_1 : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ be a pseudorandom generator, $g_2 : \{0,1\}^\lambda \to D$ be a sampler of the domain D of TD, $\mathsf{C} = (\mathsf{CGen}, \mathsf{TCGen}, \mathsf{CEval}, \mathsf{CInv})$

```
$\mathsf{TGen}_{\mathtt{I}}(1^\lambda)$
  $K \leftarrow \{0,1\}^\lambda$
  $(\mathsf{pk}_\Sigma, \mathsf{sk}_\Sigma) \leftarrow \mathsf{SGen}(1^\lambda)$
  $\mathsf{pk}_{\mathtt{I}} := \mathsf{pk}_\Sigma$
  $\mathsf{sk}_{\mathtt{I}} := (K, \mathsf{sk}_\Sigma)$
  return $(\mathsf{pk}_{\mathtt{I}}, \mathsf{sk}_{\mathtt{I}})$
```

```
$\mathsf{TGen}_{\mathtt{U}}(1^\lambda)$
  $(\mathsf{pk}_{\mathsf{C}}, \mathsf{sk}_{\mathsf{C},0}, \mathsf{sk}_{\mathsf{C},1}) \leftarrow \mathsf{CGen}(1^\lambda)$
  $(\mathsf{pk}_{\mathsf{TD}}, \mathsf{sk}_{\mathsf{TD}}) \leftarrow \mathsf{TDGen}(1^\lambda)$
  $(\mathsf{pk}_e, \mathsf{sk}_e) \leftarrow \mathsf{EGen}(1^\lambda)$
  $\mathsf{pk}_{\mathtt{U}} := (\mathsf{pk}_{\mathsf{C}}, \mathsf{pk}_{\mathsf{TD}}, \mathsf{pk}_e)$
  $\mathsf{sk}_{\mathtt{U}} := (\mathsf{sk}_{\mathsf{C},0}, \mathsf{sk}_{\mathsf{TD}}, \mathsf{sk}_e)$
  return $(\mathsf{pk}_{\mathtt{U}}, \mathsf{sk}_{\mathtt{U}})$
```

```
$\mathsf{ReTag}(\mathsf{pk}_{\mathtt{I}}, \mathsf{sk}_{\mathtt{U}}, m, m', \tau)$
  $\mu \leftarrow \mathsf{CEval}(\mathsf{pk}_{\mathsf{C}}, m; \rho_1)$
  $y \leftarrow \mathsf{TDEval}(\mathsf{pk}_{\mathsf{TD}}, \mu, \rho_2)$
  $\rho_1' \leftarrow \{0,1\}^{2\lambda}$
  $\mu' \leftarrow \mathsf{CEval}(\mathsf{pk}_{\mathsf{C}}, m'; \rho_1')$
  $\rho_2' \leftarrow \mathsf{TDInv}(\mathsf{sk}_{\mathsf{TD}}, \mu', y)$
  $\tau' := (\rho_1', \rho_2', q_1, q_2, q_3, c, \sigma)$
  return $\tau'$
```

```
$\mathsf{Tag}(\mathsf{sk}_{\mathtt{I}}, \mathsf{pk}_{\mathtt{U}}, m)$
  $q_i \leftarrow \{0,1\}^\lambda, i \in \{1,2,3\}$
  $r_i \leftarrow F(K, q_i), i \in \{1,2,3\}$
  $\rho_i \leftarrow g_i(r_i), i \in \{1,2\}$
  $\mu \leftarrow \mathsf{CEval}(\mathsf{pk}_{\mathsf{C}}, m; \rho_1)$
  $y \leftarrow \mathsf{TDEval}(\mathsf{pk}_{\mathsf{TD}}, \mu, \rho_2)$
  $c \leftarrow \mathsf{Enc}(\mathsf{pk}_e, m; r_3)$
  $\eta := (\mathsf{pk}_{\mathtt{U}}, y, q_1, q_2, q_3, c)$
  $\sigma \leftarrow \mathsf{SSig}(\mathsf{sk}_\Sigma, \eta)$
  $\tau := (\rho_1, \rho_2, q_1, q_2, q_3, c, \sigma)$
  return $\tau$
```

```
$\mathsf{Ver}(\mathsf{pk}_{\mathtt{I}}, \mathsf{pk}_{\mathtt{U}}, m, \tau)$
  $\mu \leftarrow \mathsf{CEval}(\mathsf{pk}_{\mathsf{C}}, m; \rho_1)$
  $y \leftarrow \mathsf{TDEval}(\mathsf{pk}_{\mathsf{TD}}, \mu, \rho_2)$
  $\eta := (\mathsf{pk}_{\mathtt{U}}, y, q_1, q_2, q_3, c)$
  $b \leftarrow \mathsf{SVer}(\eta, \sigma, \mathsf{pk}_\Sigma)$
  return $b$
```

```
$\mathsf{TProv}(\mathsf{sk}_{\mathtt{I}}, \mathsf{pk}_{\mathtt{U}}, m, \tau)$
  $r_i \leftarrow F(K, q_i), i \in \{1,2,3\}$
  $\pi := (r_1, r_2, r_3)$
  return $\pi$
```

```
$\mathsf{TJud}(\mathsf{pk}_{\mathtt{I}}, \mathsf{pk}_{\mathtt{U}}, m, \tau, \pi)$
  parse $\pi = (r_1', r_2', r_3')$
  $\rho_i' \leftarrow g(r_i'), i \in \{1,2\}$
  $m' \leftarrow \mathsf{Ext}(\mathsf{pk}_e, c, r_3)$
  $\mu \leftarrow \mathsf{CEval}(\mathsf{pk}_{\mathsf{C}}, m; \rho_1)$
  $y \leftarrow \mathsf{TDEval}(\mathsf{pk}_{\mathsf{TD}}, \mu, \rho_2)$
  $\mu' \leftarrow \mathsf{CEval}(\mathsf{pk}_{\mathsf{C}}, m'; \rho_1')$
  $y' \leftarrow \mathsf{TDEval}(\mathsf{pk}_{\mathsf{TD}}, \mu, \rho_2')$
  if $y = y' \wedge$
    $(\mu, \rho_2) \neq (\mu', \rho_2')$ then
    return $d = \mathtt{U}$
  else
    return $d = \mathtt{I}$
  endif
```

Fig. 1. Our rerandomizable tagging scheme

be a double-trapdoor chameleon hash which hashes messages $m \in \{0,1\}^*$ with randomness $\rho \in \{0,1\}^{2\lambda}$, $\mathsf{TD} = (\mathsf{TDGen}, \mathsf{TDEval}, \mathsf{TDInv})$ be a tag-based trapdoor function, $\mathcal{E} = (\mathsf{EGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Ext})$ be an extractable public key encryption scheme, and $\Sigma = (\mathsf{SGen}, \mathsf{SSig}, \mathsf{SVer})$ be a signature scheme. We construct a rerandomizable tagging scheme $\mathcal{RT}$ as shown in Figure 1. The correctness of $\mathcal{RT}$ follows those of the building blocks.

Theorem 1. *If one-way function exists, then $\mathcal{RT}$ is user-accountable. If C is collision-resistant, and TD is collision-resistant under selective-tag adaptive-image attacks, then $\mathcal{RT}$ is issuer-accountable. If one-way function exists, $\mathcal{E}$ is CPA-secure, and TD supports domain and pre-image sampling, then $\mathcal{RT}$ is proof-restrictedly transparent.*

We refer the readers to Appendix E for the detailed proof.

4 Accountable Ring Signatures

Accountable ring signatures allow both spontaneous group formulation as ring signatures and designated opening of signer identity as group signatures. Xu and Yung [XY04] introduced this primitive. Bootle *et al.* [BCC+15] recently formalized it, and gave both a generic construction and an efficient instantiation in the random oracle model. We follow the definitions of Bootle *et al.* [BCC+15], which can be found in Appendix C.

We adopt the ring signature scheme of Bose *et al.* [BDR15] (referred to as BDR hereinafter), which in turn uses the full Boneh-Boyen (FBB) signature scheme [BB04] for signing hash values output by a collision-resistant hash function $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_n$. We transform BDR into an accountable ring signature scheme $\mathcal{RS}$, described in Figure 2 and 3, by using a structure-preserving encryp-

```
$\mathsf{RSetup}(1^\lambda)$
$(crs, \mathcal{G}, \mathsf{xk}) \leftarrow \mathsf{GSSetup}(1^\lambda)$ where
$\mathcal{G} = (n, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \hat{e}, g_1, g_2)$
$\beta \leftarrow \mathbb{Z}_n^*$
$qSDH := (g_1, g_1^\beta, g_1^{\beta^2}, \ldots, g_1^{\beta^q})$
$\mathsf{pp} := (1^\lambda, \mathcal{G}, crs, qSDH, g_2^\beta)$
return $\mathsf{pp}$
```

```
$\mathsf{ROKGen}(\mathsf{pp})$
$(\mathsf{opk}, \mathsf{osk}) \leftarrow \mathsf{EGen}(1^\lambda)$
return $(\mathsf{opk}, \mathsf{osk})$
```

```
$\mathsf{RUKGen}(\mathsf{pp})$
$\mathsf{sk} := (a, b) \leftarrow \mathbb{Z}_n^2$
$A := g_2^a$
$B := g_2^b$
$q_a := a^2 \bmod n$
$q_b := b^2 \bmod n$
$\mathsf{pk} := (A, B, q_a, q_b)$
return $(\mathsf{pk}, \mathsf{sk})$
```

```
$\mathsf{ROpen}(\mathsf{osk}, m, \mathcal{R}, \sigma)$ where $\mathcal{R} = \{\mathsf{pk}_i = (q_{i,a}, q_{i,b})\}_{i=1}^k$
$A^* \leftarrow \mathsf{Dec}(\mathsf{osk}, e_a)$
$B^* \leftarrow \mathsf{Dec}(\mathsf{osk}, e_b)$
if $\exists\ i, q_a, q_b$ s.t. $\mathsf{pk}_i = (A^*, B^*, q_a, q_b)$ then
    $\phi_{d_a} \leftarrow \mathsf{GSProv}(\{A^* = \mathsf{Dec}(\underline{\mathsf{osk}}, e_a)\}, \mathsf{osk})$
    $\phi_{d_b} \leftarrow \mathsf{GSProv}(\{B^* = \mathsf{Dec}(\underline{\mathsf{osk}}, e_b)\}, \mathsf{osk})$
    $\mathsf{pk}^* := \mathsf{pk}_i$
    $\psi := (\phi_{d_a}, \phi_{d_b})$
    return $(\mathsf{pk}^*, \psi)$
else
    return $\perp$
endif
```

```
$\mathsf{RJud}(\mathsf{opk}, m, \mathcal{R}, \sigma, \mathsf{pk}^*, \psi)$ where $\mathcal{R} = \{\mathsf{pk}_i = (q_{i,a}, q_{i,b})\}_{i=1}^k$
$c_{d_a} \leftarrow \mathsf{GSVer}(\{A^* = \mathsf{Dec}(\underline{\mathsf{osk}}, e_a)\}, \phi_{d_a})$
$c_{d_b} \leftarrow \mathsf{GSVer}(\{B^* = \mathsf{Dec}(\underline{\mathsf{osk}}, e_b)\}, \phi_{d_b})$
return $c_{d_a} \wedge c_{d_b}$
```

Fig. 2. Our accountable ring signature scheme - Part I

tion scheme $\mathcal{SPE} = (\mathsf{EGen}, \mathsf{Enc}, \mathsf{Dec})$ of Camenisch *et al.* [CHK+11] which is secure against chosen-ciphertext attack (CCA). We use a collision-resistant hash function $H_2 : \mathbb{Z}_n \to \mathbb{G}_1$ to create the labels for $\mathcal{SPE}$. Roughly, we encrypt the public key and prove using the Groth-Sahai proof system [GS08] $\mathcal{GS} = (\mathsf{GSSetup}, \mathsf{GSProv}, \mathsf{GSVer})$ that the encrypted key matches with the one for verifying the BDR signature. A tracing authority holding the decryption key can identify the real signer. BDR ring signature requires composite order group, so our accountable ring signature is also constructed in a composite order group setting. Our scheme inherits the nice features of BDR, including constant signature size and security without random oracles. We underline the witness components in the statement to be proven by Groth-Sahai proof system. The details of $\mathcal{SPE}$ and Groth-Sahai proof system can be found in Appendix A.

Analysis. The correctness of $\mathcal{RS}$ follows those of BDR ring signatures and the proof on $\mathcal{SPE}$ ciphertexts. The efficiency of $\mathcal{RS}$ depends on the instantiation of the Groth-Sahai proof. Instantiating $\mathcal{SPE}$ and our accountable ring signature scheme with a composite order group, and the Groth-Sahai proof system with the symmetric external Diffie-Hellman (SXDH) assumption [GS08], the signing algorithm $\mathsf{RSig}()$ requires 121 multiplications, 102 exponentiations (including the commitments for the proofs), and 10 pairings.

Theorem 2. *If $\mathcal{GS}$ is sound and the underlying scheme [BDR15] is unforgeable, then $\mathcal{RS}$ is unforgeable. If $\mathcal{SPE}$ is CCA-secure, and $\mathcal{GS}$ is hiding, then $\mathcal{RS}$ is CCA-anonymous under full key exposure. If $\mathcal{GS}$ is complete, and $\mathcal{SPE}$ is perfectly correct, then $\mathcal{RS}$ is traceable. If $\mathcal{GS}$ is sound, and $\mathcal{SPE}$ is perfectly correct, then $\mathcal{RS}$ has tracing soundness.*

We refer the readers to Appendix F for the detailed proof.

5 Constructions of Sanitizable Signatures

Syntax. Sanitizable signature schemes allow the delegation of signing capabilities to a sanitizer. These capabilities are realized by letting the signer "attach" a description of the admissible modifications for

```
$\mathsf{RSig}(\mathsf{opk}, m, \mathcal{R}, \mathsf{sk})$ where $\mathcal{R} = \{\mathsf{pk}_i = (q_{i,a}, q_{i,b})\}_{i=1}^{k}$ and $\mathsf{pk} \in \mathcal{R}$
    $m' \leftarrow H_1(m||\{\mathsf{pk}_i\})$;   $\rho \leftarrow \mathbb{Z}_n \setminus \{\frac{-a+m'}{b}\}$;   $\varDelta \leftarrow g_1^{\frac{1}{a+\rho b+m'}}$
    $\mathcal{R}_a := (q_{1,a}, \ldots, q_{k,a})$;   $\mathcal{R}_b := (q_{1,b}, \ldots, q_{k,b})$
    $W_a \leftarrow \mathsf{MemWit}(\mathsf{pp}, q_a, \mathcal{R}_a)$;   $W_b \leftarrow \mathsf{MemWit}(\mathsf{pp}, q_b, \mathcal{R}_b)$
    $\phi_{mem_a} \leftarrow \mathsf{MemProv}(\mathsf{pp}, \mathcal{R}_a, W_a)$;   $\phi_{mem_b} \leftarrow \mathsf{MemProv}(\mathsf{pp}, \mathcal{R}_b, W_b)$
    $\phi_{q_a} \leftarrow \mathsf{GSProv}(\{\underline{q_a} = \underline{a}^2\}, (q_a, a))$;   $\phi_{q_b} \leftarrow \mathsf{GSProv}(\{\underline{q_b} = \underline{b}^2\}, (q_b, b))$
    $\phi_{\mathsf{pk}_a} \leftarrow \mathsf{GSProv}(\{\underline{A} = g_2^{\underline{a}}\}, (A, a))$;   $\phi_{\mathsf{pk}_b} \leftarrow \mathsf{GSProv}(\{\underline{B} = g_2^{\underline{b}}\}, (B, b))$
    $e_a \leftarrow \mathsf{Enc}(\mathsf{opk}, H_2(m'), A; r_a)$;   $e_b \leftarrow \mathsf{Enc}(\mathsf{opk}, H_2(m'), B; r_b)$
    $\phi_{e_a} \leftarrow \mathsf{GSProv}(\{e_a = \mathsf{Enc}(\mathsf{opk}, H_2(m'), \underline{A}; \underline{r_a})\}, (A, r_a))$
    $\phi_{e_b} \leftarrow \mathsf{GSProv}(\{e_b = \mathsf{Enc}(\mathsf{opk}, H_2(m'), \underline{B}; \underline{r_b})\}, (B, r_b))$
    $\phi_{sig} \leftarrow \mathsf{GSProv}(\{\underline{B}^{\rho} = \underline{B'} \wedge e(\underline{\varDelta}, \underline{A})e(\underline{\varDelta}, \underline{B'})e(\underline{\varDelta}, g_2^{m'}) = e(g_1, g_2)\}, (\varDelta, A, B, B'))$
    return $\sigma := (\rho, e_a, e_b, \phi_{mem_a}, \phi_{mem_b}, \phi_{sig}, \phi_{q_a}, \phi_{q_b}, \phi_{\mathsf{pk}_a}, \phi_{\mathsf{pk}_b}, \phi_{e_a}, \phi_{e_b})$

$\mathsf{RVer}(\mathsf{opk}, m, \mathcal{R}, \sigma)$ where $\mathcal{R} = \{\mathsf{pk}_i = (q_{i,a}, q_{i,b})\}_{i=1}^{k}$
    $m' \leftarrow H_1(m||\{\mathsf{pk}_i\})$;   $\mathcal{R}_a := (q_{1,a}, \ldots, q_{k,a})$;   $\mathcal{R}_b := (q_{1,b}, \ldots, q_{k,b})$
    $c_{mem_a} \leftarrow \mathsf{MemVer}(\mathsf{pp}, \mathcal{R}_a, \phi_{mem_a})$;   $c_{mem_b} \leftarrow \mathsf{MemVer}(\mathsf{pp}, \mathcal{R}_b, \phi_{mem_b})$
    $c_{q_a} \leftarrow \mathsf{GSVer}(\{\underline{q_a} = \underline{a}^2\}, \phi_{q_a})$;   $c_{q_b} \leftarrow \mathsf{GSVer}(\{\underline{q_b} = \underline{b}^2\}, \phi_{q_b})$
    $c_{\mathsf{pk}_A} \leftarrow \mathsf{GSVer}(\{\underline{A} = g_2^{\underline{a}}\}, \phi_{\mathsf{pk}_A})$;   $c_{\mathsf{pk}_B} \leftarrow \mathsf{GSVer}(\{\underline{B} = g_2^{\underline{b}}\}, \phi_{\mathsf{pk}_B})$
    $c_{e_a} \leftarrow \mathsf{GSVer}(\{e_a = \mathsf{Enc}(\mathsf{opk}, H_2(m'), \underline{A}; \underline{r_a})\}, \phi_{e_a})$
    $c_{e_b} \leftarrow \mathsf{GSVer}(\{e_b = \mathsf{Enc}(\mathsf{opk}, H_2(m'), \underline{B}; \underline{r_b})\}, \phi_{e_b})$
    $c_{sig} \leftarrow \mathsf{GSVer}(\{\underline{B}^{\rho} = \underline{B'} \wedge e(\underline{\varDelta}, \underline{A})e(\underline{\varDelta}, \underline{B'})e(\underline{\varDelta}, g_2^{m'}) = e(g_1, g_2)\}, \phi_{sig})$
    return $(c_{mem_a} \wedge c_{mem_b} \wedge c_{sig} \wedge c_{q_a} \wedge c_{q_b} \wedge c_{\mathsf{pk}_A} \wedge c_{\mathsf{pk}_B} \wedge c_{e_a} \wedge c_{e_b})$
```

Fig. 3. Our accountable ring signature scheme - Part II

a particular message and sanitizer. The sanitizers may then change the message according to some modification and update the signature. More formally, the signer uses its private key $\mathsf{sk_S}$ to sign a message m and the description of the admissible modifications α for some sanitizer $\mathsf{pk_Z}$. The sanitizer, having a matching private key $\mathsf{sk_Z}$, can update the message according to some modification δ and compute a new signature using $\mathsf{sk_Z}$. If there is a dispute about the origin of a message-signature pair, the signer can compute a proof π (using an algorithm Prov) from previously signed messages which (dis)proves that a signature has been created by the sanitizer. The verification of this proof is done by an algorithm Jud (that only decides the origin of a valid message-signature pair in question; for invalid pairs such decisions are in general impossible). We mostly follow the existing syntax [BFF+09, BFLS10] except that our key generation algorithms take as input a public parameter generated by a setup algorithm. For the formal syntax and security definitions of sanitizable signatures, readers can refer to Appendix D.

Sanitizable signatures should satisfy (proof-restricted) privacy, immutability, sanitizer- and signer-accountability, and (proof-restricted) transparency. Some schemes also satisfy the even stronger unlinkability. It is known that full transparency or unlinkability both imply privacy separately [BFF+09, BFLS10], while proof-restricted transparency implies a proof-restricted privacy [BFLS10].

To the best of our knowledge, there is no efficient instantiation of sanitizable signatures satisfying either proof-restricted privacy or unlinkability, and all other security properties simultaneously, without using random oracles. We thus fill this gap by describing two constructions. The first is more efficient while satisfying privacy based on the rerandomizable tagging. The second one uses the accountable ring signature scheme and can achieve unlinkability.

```
$\mathsf{Setup}(1^\lambda)$
    $\mathsf{pp} = 1^\lambda$
    return $\mathsf{pp}$

$\mathsf{Prov}(\mathsf{sk_S}, \mathsf{pk_Z}, m, \sigma)$
    $\pi \leftarrow \mathsf{TProv}(\mathsf{sk}_\mathtt{I}, \mathsf{pk}_\mathtt{U}, m, \tau)$
    return $\pi$

$\mathsf{KGen_S}(\mathsf{pp})$
    $(\mathsf{pk_f}, \mathsf{sk_f}) \leftarrow \mathsf{SGen}(1^\lambda)$
    $(\mathsf{pk}_\mathtt{I}, \mathsf{sk}_\mathtt{I}) \leftarrow \mathsf{TGen}_\mathtt{I}(1^\lambda)$
    $\mathsf{pk_S} = (\mathsf{pk_f}, \mathsf{pk}_\mathtt{I})$
    $\mathsf{sk_S} = (\mathsf{sk_f}, \mathsf{sk}_\mathtt{I})$
    return $(\mathsf{pk_S}, \mathsf{sk_S})$

$\mathsf{KGen_Z}(\mathsf{pp})$
    $(\mathsf{pk}_\mathtt{U}, \mathsf{sk}_\mathtt{U}) \leftarrow \mathsf{TGen}_\mathtt{U}(1^\lambda)$
    $\mathsf{pk_Z} = \mathsf{pk}_\mathtt{U}$
    $\mathsf{sk_Z} = \mathsf{sk}_\mathtt{U}$
    return $(\mathsf{pk_Z}, \mathsf{sk_Z})$
```

Fig. 4. Our first sanitizable signature scheme - Part I

```
$\mathsf{Sig}(\mathsf{sk_S}, \mathsf{pk_Z}, m, \alpha)$
    $m_f := (f_\alpha(m), \mathsf{pk_Z}, \alpha)$
    $\sigma_f \leftarrow \mathsf{SSig}(\mathsf{sk_f}, m_f)$
    $\tau \leftarrow \mathsf{Tag}(\mathsf{sk}_\mathtt{I}, \mathsf{pk}_\mathtt{U}, m)$
    $\sigma := (\sigma_f, \tau, \alpha)$
    return $\sigma$

$\mathsf{San}(\mathsf{pk_S}, \mathsf{sk_Z}, m, \delta, \sigma)$
    $m_f := (f_\alpha(m), \mathsf{pk_Z}, \alpha)$
    $m' \leftarrow \delta(m)$
    $\tau' \leftarrow \mathsf{ReTag}(\mathsf{pk}_\mathtt{I}, \mathsf{sk}_\mathtt{U}, m, m', \tau)$
    $\sigma' := (\sigma_f, \tau', \alpha)$
    return $(m', \sigma')$

$\mathsf{Ver}(\mathsf{pk_S}, \mathsf{pk_Z}, m, \sigma)$
    $m_f := (f_\alpha(m), \mathsf{pk_Z}, \alpha)$
    if $\mathsf{SVer}(m_f, \sigma_f, \mathsf{pk_f}) = 1 \ \wedge$
        $\mathsf{TVer}(\mathsf{pk}_\mathtt{I}, \mathsf{pk}_\mathtt{U}, m, \tau) = 1$ then
        return $1$
    else
        return $0$
    endif

$\mathsf{Jud}(\mathsf{pk_S}, \mathsf{pk_Z}, m, \sigma, \pi)$
    if $\mathsf{TJud}(\mathsf{pk}_\mathtt{I}, \mathsf{pk}_\mathtt{U}, m, \tau, \pi) = \mathtt{U}$ then
        return $d = \mathtt{Z}$
    else
        return $d = \mathtt{S}$
    endif
```

Fig. 5. Our first sanitizable signature scheme - Part II

5.1 Basic Construction from Rerandomizable Tagging Scheme

Informal Description. Our first construction relies heavily on the rerandomizable tagging scheme (Section 2) which captures the accountability properties of sanitizable signatures. We complement it with signature schemes to restrict the malleability delegated to the sanitizers. The details of signature schemes can be found in Appendix A.

To sign, the signer computes a tag τ of the message m using the rerandomizable tagging scheme. Then, the signer uses its long-term private key $\mathsf{sk_S}$ to sign the fixed part of the message $f_\alpha(m)$, the sanitizer public key $\mathsf{pk_Z}$, and the admissible modifications α. The signature thus consists of a signature σ_f of the fixed part, the tag τ, and the admissible modifications α. To sanitize, the sanitizer rerandomizes the tag with respect to the new message $m' = \delta(m)$ using the rerandomizable tagging scheme, and replaces the tag in the signature with the rerandomized one.

Formal Description. Let $\Sigma = (\mathsf{SGen}, \mathsf{SSig}, \mathsf{SVer})$ be a digital signature scheme, and $\mathcal{RT} = (\mathsf{TGen}_\mathtt{I}, \mathsf{TGen}_\mathtt{U}, \mathsf{Tag}, \mathsf{ReTag}, \mathsf{TProv}, \mathsf{TJud})$ be a rerandomizable tagging scheme (Section 2). Figure 4 and 5 describe our first sanitizable signature scheme $\mathcal{SS}_1$. Its correctness follows directly from those of Σ and $\mathcal{RT}$.

Theorem 3. *If Σ is EUF-CMA secure, then $\mathcal{SS}_1$ is immutable. If Σ is EUF-CMA secure, and $\mathcal{RT}$ is user-accountable, then $\mathcal{SS}_1$ is sanitizer-accountable. If $\mathcal{RT}$ is issuer-accountable, then $\mathcal{SS}_1$ is signer-accountable. If $\mathcal{RT}$ is proof-restrictedly transparent, then $\mathcal{SS}_1$ is proof-restrictedly transparent.*

We refer the readers to Appendix G for the detailed proof.

```
$\mathsf{Setup}(1^\lambda)$
    $\mathsf{pp}_{\mathcal{RS}} \leftarrow \mathsf{RSetup}(1^\lambda)$
    $\mathsf{pp} := (1^\lambda, \mathsf{pp}_{\mathcal{RS}})$
    return $\mathsf{pp}$

$\mathsf{KGen}_{\mathsf{Z}}(\mathsf{pp})$
    $(\mathsf{pk}_{\mathcal{RS}}, \mathsf{sk}_{\mathcal{RS}}) \leftarrow \mathsf{RUKGen}(\mathsf{pp}_{\mathcal{RS}})$
    $\mathsf{pk}_{\mathsf{Z}} := \mathsf{pk}_{\mathcal{RS}}$
    $\mathsf{sk}_{\mathsf{Z}} := \mathsf{sk}_{\mathcal{RS}}$
    return $(\mathsf{pk}_{\mathsf{Z}}, \mathsf{sk}_{\mathsf{Z}})$

$\mathsf{KGen}_{\mathsf{S}}(\mathsf{pp})$
    $(\mathsf{pk}_{\mathsf{f}}, \mathsf{sk}_{\mathsf{f}}) \leftarrow \mathsf{SGen}(1^\lambda)$
    $(\mathsf{opk}_{\mathcal{RS}}, \mathsf{osk}_{\mathcal{RS}}) \leftarrow \mathsf{ROKGen}(\mathsf{pp}_{\mathcal{RS}})$
    $(\mathsf{pk}_{\mathcal{RS}}, \mathsf{sk}_{\mathcal{RS}}) \leftarrow \mathsf{RUKGen}(\mathsf{pp}_{\mathcal{RS}})$
    $\mathsf{pk}_{\mathsf{S}} := (\mathsf{pk}_{\mathsf{f}}, \mathsf{opk}_{\mathcal{RS}}, \mathsf{pk}_{\mathcal{RS}})$
    $\mathsf{sk}_{\mathsf{S}} := (\mathsf{sk}_{\mathsf{f}}, \mathsf{osk}_{\mathcal{RS}}, \mathsf{sk}_{\mathcal{RS}})$
    return $(\mathsf{pk}_{\mathsf{S}}, \mathsf{sk}_{\mathsf{S}})$

$\mathsf{Sig}(\mathsf{sk}_{\mathsf{S}}, \mathsf{pk}_{\mathsf{Z}}, m, \alpha)$
    $\mathcal{R} := \{\mathsf{pk}_{\mathcal{RS}}, \mathsf{pk}'_{\mathcal{RS}}\}$
    $m_f := (f_\alpha(m), \alpha, \mathcal{R})$
    $\sigma_f \leftarrow \mathsf{SSig}(\mathsf{sk}_{\mathsf{f}}, m_f)$
    $\hat{\sigma} \leftarrow \mathsf{RSig}(\mathsf{opk}_{\mathcal{RS}}, m, \mathcal{R}, \mathsf{sk}_{\mathcal{RS}})$
    $\sigma := (\sigma_f, \hat{\sigma}, \alpha)$
    return $\sigma$

$\mathsf{San}(\mathsf{pk}_{\mathsf{S}}, \mathsf{sk}_{\mathsf{Z}}, m, \delta, \sigma)$
    $\mathcal{R} := \{\mathsf{pk}_{\mathcal{RS}}, \mathsf{pk}'_{\mathcal{RS}}\}$
    $m' \leftarrow \delta(m)$
    $\hat{\sigma}' \leftarrow \mathsf{RSig}(\mathsf{opk}_{\mathcal{RS}}, m', \mathcal{R}, \mathsf{sk}'_{\mathcal{RS}})$
    $\sigma' := (\sigma_f, \hat{\sigma}', \alpha)$
    return $(m', \sigma')$

$\mathsf{Ver}(\mathsf{pk}_{\mathsf{S}}, \mathsf{pk}_{\mathsf{Z}}, m, \sigma)$
    $\mathcal{R} := \{\mathsf{pk}_{\mathcal{RS}}, \mathsf{pk}'_{\mathcal{RS}}\}$
    $m_f := (f_\alpha(m), \alpha, \mathcal{R})$
    $b_1 \leftarrow \mathsf{RVer}(\mathsf{opk}_{\mathcal{RS}}, m, \mathcal{R}, \hat{\sigma})$
    $b_2 \leftarrow \mathsf{SVer}(m_f, \sigma_f, \mathsf{pk}_{\mathsf{f}})$
    return $(b_1 \wedge b_2)$
```

Fig. 6. Our second sanitizable signature scheme - Part I

5.2 Unlinkability from Accountable Ring Signatures

Our second construction is similar to the construction by Brzuska *et al.* [BFLS10] based on group signatures, except that we replace the special group signatures with accountable ring signatures reviewed in Section 4. This change has two interesting effects. First, the construction of sanitizable signatures becomes simpler: The signer does not need to create a new group for each sanitizable signature, which also eliminates the use of pseudorandom functions to generate the group [BFLS10]. Second, in contrast to the special group signatures, of which the instantiations (with or without random oracle heuristics) are not efficient [FKM$^+$16], our accountable ring signatures scheme in Section 4 is efficient and is secure without random oracles, though it requires composite order group.

Another route leading to our discovery is the observation that the fully dynamic group signatures constructed from accountable ring signatures [BCC$^+$15, BCC$^+$16] features the property that the user key generation does not depend on the group key pair, which is the property required in the sanitizable signatures construction by Brzuska *et al.* [BFLS10].

Informal Description. We proceed directly to the signing and sanitizing procedures. To issue a signature, the signer forms a ring consisting of itself and the sanitizer, and ring-signs the message. It binds the sanitizer to this sanitizing chain by signing the fixed part of the message together with the sanitizer public key using its private key. Sanitizing becomes computing a new accountable ring signature on the modified message.

Formal Description. Let $\mathcal{RS} = (\mathsf{RSetup}, \mathsf{ROKGen}, \mathsf{RUKGen}, \mathsf{RSig}, \mathsf{RVer}, \mathsf{ROpen}, \mathsf{RJud})$ be an accountable ring signature scheme (Section 4), and $\Sigma = (\mathsf{SGen}, \mathsf{SSig}, \mathsf{SVer})$ be a deterministic signature scheme. Figures 6 and 7 describe the construction of our unlinkable sanitizable signature scheme $\mathcal{SS}_2$. The correctness of $\mathcal{SS}_2$ follows those of $\mathcal{RS}$ and Σ.

```
$\mathsf{Prov}(\mathsf{sk}_\mathtt{S}, \mathsf{pk}_\mathtt{Z}, m, \sigma)$
$\mathcal{R} := \{\mathsf{pk}_{\mathcal{RS}}, \mathsf{pk}'_{\mathcal{RS}}\}$
$(\mathsf{pk}^*_{\mathcal{RS}}, \psi) \leftarrow \mathsf{ROpen}(\mathsf{osk}_{\mathcal{RS}}, m, \mathcal{R}, \hat{\sigma})$
$\pi := (\mathsf{pk}^*_{\mathcal{RS}}, \psi)$
return $\pi$
```

```
$\mathsf{Jud}(\mathsf{pk}_\mathtt{S}, \mathsf{pk}_\mathtt{Z}, m, \sigma, \pi)$
$\mathcal{R} := \{\mathsf{pk}_{\mathcal{RS}}, \mathsf{pk}'_{\mathcal{RS}}\}$
if $\mathsf{RJud}(\mathsf{opk}_{\mathcal{RS}}, m, \mathcal{R}, \hat{\sigma}, \mathsf{pk}^*_{\mathcal{RS}}, \psi) = 1$
  $\wedge\ \mathsf{pk}^*_{\mathcal{RS}} = \mathsf{pk}'_{\mathcal{RS}}$ then
  return $d := \mathtt{Z}$
else
  return $d := \mathtt{S}$
endif
```

Fig. 7. Our second sanitizable signature scheme - Part II

Multiple Sanitizers. Ring signatures support rings containing more than two members, so we can extend $\mathcal{SS}_2$ easily to support more sanitizers: The signer can sign the public keys of a ring of multiple sanitizers when issuing a sanitizable signature. This grants partial signing power to each the sanitizers (possibly corresponding to different admissible modifications). Furthermore, since our accountable ring signatures have constant signature size with respect to the number of users in the ring, the scheme supporting multiple sanitizers also features constant signature size with respect to the number of sanitizers.

Theorem 4. *If Σ is sEUF-CMA-secure, then $\mathcal{SS}_2$ is immutable and unlinkable. If $\mathcal{RS}$ is traceable and unforgeable, then $\mathcal{SS}_2$ is sanitizer-accountable. If $\mathcal{RS}$ is unforgeable, then $\mathcal{SS}_2$ is signer-accountable. If $\mathcal{RS}$ is anonymous, $\mathcal{SS}_2$ is proof-restrictedly transparent.*

We refer the readers to Appendix H for the detailed proof.

6 Concluding Remarks

We compare our two constructions with some existing schemes in Table 1, taking both their security and efficiency into consideration. To compare with $\mathcal{SS}_1$ the signature scheme Σ is instantiated with Waters signature [Wat05], the extractable public-key encryption scheme $\mathcal{E}$ is instantiated with Cramer-Shoup encryption [CS98], and the double-trapdoor chameleon hash is instantiated with the scheme by Chen *et al.* [CZT$^+$08]. 'E' denotes group exponentiation, 'P' denotes pairing, and 'M' denotes matrix multiplication. For $\mathcal{SS}_2$, Σ is instantiated with full Boneh-Boyen signature [BB04]. For simplicity, we do not differentiate between elements from different groups in this comparison. A more detailed comparison can be found in the full version. We remark that $\mathcal{SS}_2$ is instantiated with a composite order group. It shows that instantiating our generic construction leads to the first efficient unlinkable sanitizable signature schemes in the standard model.

Acknowledgments

We thank the anonymous reviewers for their comments on our manuscript.

Sherman Chow is supported by the Early Career Award and the Early Career Scheme (CUHK 439713), and General Research Funds (CUHK 14201914) of the Research Grants Council, University Grant Committee of Hong Kong.

Dominique Schröder is supported by the German Federal Ministry of Education and Research (BMBF) through funding for the project PROMISE and by the German research foundation (DFG) through funding for the collaborative research center 1223.

References

ACdT05. Giuseppe Ateniese, Daniel H. Chou, Breno de Medeiros, and Gene Tsudik. Sanitizable signatures. In Sabrina De Capitani di Vimercati, Paul F. Syverson, and Dieter Gollmann, editors, *ESORICS 2005: 10th European Symposium on Research in Computer Security*, volume 3679 of *Lecture Notes in Computer Science*, pages 159–177, Milan, Italy, September 12–14, 2005. Springer, Heidelberg, Germany.

	$\mathcal{SS}_1$	$\mathcal{SS}_2$	[FKM+16]	[BFLS10] using [Gro07]	[BFLS10] using [FY05]	[BFF+09]
Security	Privacy	Unlinkability	Unlinkability	Unlinkability	Unlinkability	Privacy
Model	Standard	Standard	ROM	Standard	ROM	ROM
Group	Prime	Composite	Prime	Prime	Composite	Prime
Assumption	Static	q-type	Static	GGM	GGM	Static
KGen_S	6E+2P	32E+1P	7E	1E	1E	3E +1P
KGen_Z	7E+1M	2E	1E	1E	4E	2E
Sig	11E+2M	103E+10P	15E	194E+2P	2813E	$(2 \cdot \lvert m\rvert + 2)$E
San	4E+10M	102E+10P	14E	186E+1P	2814E	2P
Ver	2E+4P+2M	2E+148P	17E	207E+62P	2011E	$2 \cdot \lvert m\rvert$E +2P
Prov	$\perp$	126E+152P	23E	14E+1P	18E	$4 \cdot \lvert m\rvert^2$E
Jud	5E+4M	152P	6E	1E+2P	2E	4E
pk_S	$(8+2\lvert m\rvert)\mathbb{G}_1 + 2\mathbb{G}_T$	$16\mathbb{G}_1 + 5\mathbb{G}_2 + 1\mathbb{G}_T + 2\mathbb{Z}n$	$5\mathbb{G} + 2\mathbb{Z}p$	$1\mathbb{G}$	$1\mathbb{G}$	$(4+\lvert m\rvert)\mathbb{G}_1 + 1\mathbb{G}_T$
sk_S	$2\mathbb{G}_1 + 1\mathbb{Z}p$	$25\mathbb{Z}n$	$7\mathbb{Z}p$	$1\mathbb{Z}p$	$1\mathbb{Z}p$	$1\mathbb{G}_1 + 1\mathbb{Z}p$
pk_Z	$5\mathbb{G}_1 + (\ell + 1)\mathbb{Z}_p^{n\times nk}$	$2\mathbb{G}_2 + 2\mathbb{Z}n$	$1\mathbb{G}$	$3\mathbb{G} + 1\mathbb{G}_T$	$2\mathbb{G}_p + 3\mathbb{G}_q$	$2\mathbb{G}_1$
sk_Z	$7\mathbb{Z}p + 1\mathbb{Z}^{\bar{m}\times nk}$	$2\mathbb{Z}n$	$1\mathbb{Z}p$	$2\mathbb{G} + 1\mathbb{Z}p$	$1\mathbb{Z}q$	$2\mathbb{Z}p$
σ	$8\mathbb{G}_1 + 6\mathbb{Z}p$	$23\mathbb{G}_1 + 12\mathbb{G}_2 + 82\mathbb{G}_T + 3\mathbb{Z}n$	$5\mathbb{G} + 9\mathbb{Z}p$	$65\mathbb{G} + 2\mathbb{G}_T + 2\mathbb{Z}p$	$3\mathbb{G}_p + 6\mathbb{G}_q + 1406\mathbb{Z}p + 205\mathbb{Z}q$	$2\mathbb{G}_1 + (3 + \lvert m\rvert)\mathbb{Z}p$
π	$3\mathbb{Z}p$	$86\mathbb{G}_2 + 16\mathbb{G}_T + 2\mathbb{Z}n$	$1\mathbb{G} + 3\mathbb{Z}p$	$1\mathbb{G}$	$2\mathbb{G}_q + 2\mathbb{Z}p$	$(4+\lvert m\rvert)\mathbb{G}_1 + 1\mathbb{G}_T + 5\mathbb{Z}p$

Table 1. Comparison of Different Sanitizable Signature Schemes

ACHO13. Masayuki Abe, Sherman S. M. Chow, Kristiyan Haralambiev, and Miyako Ohkubo. Double-trapdoor anonymous tags for traceable signatures. *Int. J. Inf. Sec.*, 12(1):19–31, 2013.

BB04. Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73, Interlaken, Switzerland, May 2–6, 2004. Springer, Heidelberg, Germany.

BBD+10. Christina Brzuska, Heike Busch, Özgür Dagdelen, Marc Fischlin, Martin Franz, Stefan Katzenbeisser, Mark Manulis, Cristina Onete, Andreas Peter, Bertram Poettering, and Dominique Schröder. Redactable signatures for tree-structured data: Definitions and constructions. In Jianying Zhou and Moti Yung, editors, *ACNS 10: 8th International Conference on Applied Cryptography and Network Security*, volume 6123 of *Lecture Notes in Computer Science*, pages 87–104, Beijing, China, June 22–25, 2010. Springer, Heidelberg, Germany.

BCC+15. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, Jens Groth, and Christophe Petit. Short accountable ring signatures based on DDH. In Günther Pernul, Peter Y. A. Ryan, and Edgar R. Weippl, editors, *ESORICS 2015: 20th European Symposium on Research in Computer Security, Part I*, volume 9326 of *Lecture Notes in Computer Science*, pages 243–265, Vienna, Austria, September 21–25, 2015. Springer, Heidelberg, Germany.

BCC+16. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, and Jens Groth. Foundations of fully dynamic group signatures. In Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider, editors, *ACNS 2016*, volume 9696 of *LNCS*, pages 117–136. Springer, 2016.

BDR15. Priyanka Bose, Dipanjan Das, and Chandrasekaran Pandu Rangan. Constant size ring signature without random oracle. In Ernest Foo and Douglas Stebila, editors, *ACISP 15: 20th Australasian Conference on Information Security and Privacy*, volume 9144 of *Lecture Notes in Computer Science*, pages 230–247, Wollongong, NSW, Australia, June 29 – July 1, 2015. Springer, Heidelberg, Germany.

BFF+09. Christina Brzuska, Marc Fischlin, Tobias Freudenreich, Anja Lehmann, Marcus Page, Jakob Schelbert, Dominique Schröder, and Florian Volk. Security of sanitizable signatures revisited. In Stanislaw Jarecki and Gene Tsudik, editors, *PKC 2009: 12th International Conference on Theory and Practice of Public Key Cryptography*, volume 5443 of *Lecture Notes in Computer Science*, pages 317–336, Irvine, CA, USA, March 18–20, 2009. Springer, Heidelberg, Germany.

BFLS10. Christina Brzuska, Marc Fischlin, Anja Lehmann, and Dominique Schröder. Unlinkability of sanitizable signatures. In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010: 13th International Conference on Theory and Practice of Public Key Cryptography*, volume 6056 of *Lecture Notes in Computer Science*, pages 444–461, Paris, France, May 26–28, 2010. Springer, Heidelberg, Germany.

BFM88. Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 103–112, Chicago, Illinois, USA, May 2–4, 1988. ACM Press.

BMS16. Michael Backes, Sebastian Meiser, and Dominique Schröder. Delegatable functional signatures. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016: 19th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 9614 of *Lecture Notes in Computer Science*, pages 357–386, Taipei, Taiwan, March 6–9, 2016. Springer, Heidelberg, Germany.

BMW03. Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 614–629, Warsaw, Poland, May 4–8, 2003. Springer, Heidelberg, Germany.

BPS12. Christina Brzuska, Henrich Christopher Pöhls, and Kai Samelin. Non-interactive public accountability for sanitizable signatures. In Sabrina De Capitani di Vimercati and Chris Mitchell, editors, *EuroPKI*, volume 7868 of *LNCS*, pages 178–193. Springer, 2012.

BPS13. Christina Brzuska, Henrich Christopher Pöhls, and Kai Samelin. Efficient and perfectly unlinkable sanitizable signatures without group signatures. In Sokratis K. Katsikas and Isaac Agudo, editors, *EuroPKI*, volume 8341 of *LNCS*, pages 12–30. Springer, 2013.

BPW12. Alexandra Boldyreva, Adriana Palacio, and Bogdan Warinschi. Secure proxy signature schemes for delegation of signing rights. *Journal of Cryptology*, 25(1):57–115, January 2012.

Cat14. Dario Catalano. Homomorphic signatures and message authentication codes. In Michel Abdalla and Roberto De Prisco, editors, *SCN 14: 9th International Conference on Security in Communication Networks*, volume 8642 of *Lecture Notes in Computer Science*, pages 514–519, Amalfi, Italy, September 3–5, 2014. Springer, Heidelberg, Germany.

CDFG08. Dario Catalano, Mario Di Raimondo, Dario Fiore, and Rosario Gennaro. Off-line/on-line signatures: Theoretical aspects and experimental results. In Ronald Cramer, editor, *PKC 2008: 11th International Workshop on Theory and Practice in Public Key Cryptography*, volume 4939 of *Lecture Notes in Computer Science*, pages 101–120, Barcelona, Spain, March 9–12, 2008. Springer, Heidelberg, Germany.

CHK$^+$11. Jan Camenisch, Kristiyan Haralambiev, Markulf Kohlweiss, Jorn Lapon, and Vincent Naessens. Structure preserving CCA secure encryption and applications. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 89–106, Seoul, South Korea, December 4–8, 2011. Springer, Heidelberg, Germany.

CHKP10. David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 523–552, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany.

CJ10. Sébastien Canard and Amandine Jambert. On extended sanitizable signature schemes. In Josef Pieprzyk, editor, *Topics in Cryptology – CT-RSA 2010*, volume 5985 of *Lecture Notes in Computer Science*, pages 179–194, San Francisco, CA, USA, March 1–5, 2010. Springer, Heidelberg, Germany.

CJL12. Sébastien Canard, Amandine Jambert, and Roch Lescuyer. Sanitizable signatures with several signers and sanitizers. In Aikaterini Mitrokotsa and Serge Vaudenay, editors, *AFRICACRYPT 12: 5th International Conference on Cryptology in Africa*, volume 7374 of *Lecture Notes in Computer Science*, pages 35–52, Ifrance, Morocco, July 10–12, 2012. Springer, Heidelberg, Germany.

CLM08. Sébastien Canard, Fabien Laguillaumie, and Michel Milhau. Trapdoorsanitizable signatures and their application to content protection. In Steven M. Bellovin, Rosario Gennaro, Angelos D. Keromytis, and Moti Yung, editors, *ACNS 08: 6th International Conference on Applied Cryptography and Network Security*, volume 5037 of *Lecture Notes in Computer Science*, pages 258–276, New York, NY, USA, June 3–6, 2008. Springer, Heidelberg, Germany.

CS98. Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25, Santa Barbara, CA, USA, August 23–27, 1998. Springer, Heidelberg, Germany.

CZT$^+$08. Xiaofeng Chen, Fangguo Zhang, Haibo Tian, Baodian Wei, Willy Susilo, Yi Mu, Hyunrok Lee, and Kwangjo Kim. Efficient generic on-line/off-line (threshold) signatures without key exposure. *Inf. Sci.*, 178(21):4192–4203, 2008.

DS15. David Derler and Daniel Slamanig. Rethinking privacy for extended sanitizable signatures and a black-box construction of strongly private schemes. In Man Ho Au and Atsuko Miyaji, editors, *ProvSec 2015: 9th International Conference on Provable Security*, volume 9451 of *Lecture Notes in Computer Science*, pages 455–474, Kanazawa, Japan, November 24–26, 2015. Springer, Heidelberg, Germany.

FKM$^+$16. Nils Fleischhacker, Johannes Krupp, Giulio Malavolta, Jonas Schneider, Dominique Schröder, and Mark Simkin. Efficient unlinkable sanitizable signatures from signatures with re-randomizable keys.

In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016: 19th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 9614 of *Lecture Notes in Computer Science*, pages 301–330, Taipei, Taiwan, March 6–9, 2016. Springer, Heidelberg, Germany.

FY05. Jun Furukawa and Shoko Yonezawa. Group signatures with separate and distributed authorities. In Carlo Blundo and Stelvio Cimato, editors, *SCN 04: 4th International Conference on Security in Communication Networks*, volume 3352 of *Lecture Notes in Computer Science*, pages 77–90, Amalfi, Italy, September 8–10, 2005. Springer, Heidelberg, Germany.

Gro07. Jens Groth. Fully anonymous group signatures without random oracles. In Kaoru Kurosawa, editor, *Advances in Cryptology – ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 164–180, Kuching, Malaysia, December 2–6, 2007. Springer, Heidelberg, Germany.

GS08. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg, Germany.

JMSW02. Robert Johnson, David Molnar, Dawn Xiaodong Song, and David Wagner. Homomorphic signature schemes. In Bart Preneel, editor, *Topics in Cryptology – CT-RSA 2002*, volume 2271 of *Lecture Notes in Computer Science*, pages 244–262, San Jose, CA, USA, February 18–22, 2002. Springer, Heidelberg, Germany.

KL06. Marek Klonowski and Anna Lauks. Extended sanitizable signatures. In Min Surp Rhee and Byoungcheon Lee, editors, *ICISC 06: 9th International Conference on Information Security and Cryptology*, volume 4296 of *Lecture Notes in Computer Science*, pages 343–355, Busan, Korea, November 30 – December 1, 2006. Springer, Heidelberg, Germany.

KR00. Hugo Krawczyk and Tal Rabin. Chameleon signatures. In *ISOC Network and Distributed System Security Symposium – NDSS 2000*, San Diego, California, USA, February 2–4, 2000. The Internet Society.

MP12. Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

Wat05. Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg, Germany.

XY04. Shouhuai Xu and Moti Yung. Accountable ring signatures: A smart card approach. In Jean-Jacques Quisquater, Pierre Paradinas, Yves Deswarte, and Anas Abou El Kalam, editors, *CARDIS*, volume 153 of *IFIP*, pages 271–286. Kluwer/Springer, 2004.

A Preliminaries

A.1 Double Trapdoor Chameleon Hash Functions

A double trapdoor chameleon hash function is a chameleon hash function with two trapdoors. This means that given one of the trapdoors sk_i, a message m, some randomness r, and another message m', it is possible to find a randomness r' s.t. $\mathsf{CEval}(\mathsf{pk}, m; r) = \mathsf{CEval}(\mathsf{pk}, m'; r')$.

Definition 9 (Chameleon Hash Function). *A* double trapdoor chameleon hash function *is a tuple of PPT algorithms* $\mathcal{CH} = (\mathsf{CGen}, \mathsf{TCGen}, \mathsf{CEval}, \mathsf{CInv})$*:*

$\mathsf{CGen}(1^\lambda)$**:** *The key generation algorithm returns a key-pair* $(\mathsf{pk}, \mathsf{sk}_0, \mathsf{sk}_1)$*.*

$\mathsf{TCGen}(1^\lambda, i)$**:** *Upon input a bit i, the algorithm returns a key-pair* $(\mathsf{pk}, \mathsf{sk}_i)$*.*

$\mathsf{CEval}(\mathsf{pk}, m; r)$**:** *The hash input is a message m and some randomness $r \in \{0,1\}^\lambda$. It outputs a hash value.*

$\mathsf{CInv}(\mathsf{sk}_i, m, r, m')$**:** *Upon input of one of the trapdoors* sk_i*, a message m, some randomness r, and another message m', the collision finding algorithm returns some randomness r' s.t.* $\mathsf{CEval}(\mathsf{pk}, m; r) = \mathsf{CEval}(\mathsf{pk}, m'; r')$*.*

Distribution of Keys: *Let* $\overline{\mathsf{CGen}(1^\lambda, i)}$ *be the algorithm that first executes* $\mathsf{CGen}(1^\lambda)$ *and restricts the output to* $(\mathsf{pk}, \mathsf{sk}_i)$*. The distributions of* $\overline{\mathsf{CGen}(1^\lambda, i)}$ *and* $\mathsf{TCGen}(1^\lambda, i)$ *are identical.*

Uniform Distribution: *The output of* $\mathsf{CEval}(\mathsf{pk}, m; r)$ *is uniformly distributed, thus is independent of m. Furthermore, the distribution of* $\mathsf{CInv}(\mathsf{sk}_i, m, r, m')$ *is identical to the distribution of r for $i = 0, 1$.*

A double trapdoor chameleon hash is required to be collision-resistant, *i.e.*, no PPT adversary should be able to find $\mathsf{sk}_{i\oplus 1}$ given pk and sk_i, and there exists an efficient algorithm which, on input pk and a collision (m, r) and (m', r') s.t. $(m, r) \neq (m', r')$ and $\mathsf{CEval}(\mathsf{pk}, m; r) = \mathsf{CEval}(\mathsf{pk}, m'; r')$, outputs at least one of the trapdoors sk_i. As a consequence, it is infeasible to find such collision without one of the trapdoors.

A.2 Digital Signatures

We recall the definition of a digital signature scheme and the standard notion of existential unforgeability.

Definition 10 (Signatures). *A signature scheme* $\Sigma = (\mathsf{SGen}, \mathsf{SSig}, \mathsf{SVer})$ *is defined by:*
$\mathsf{SGen}(1^\lambda)$*: The key generation algorithm takes the security parameter* 1^λ *and generates a key pair* $(\mathsf{pk}, \mathsf{sk})$*.*
$\mathsf{SSig}(\mathsf{sk}, m)$*: The signing algorithm takes a private key* sk *and a message* m*, and outputs a signature* σ*.*
$\mathsf{SVer}(\mathsf{pk}, m, \sigma)$*: It takes a public key* pk*, a message* m*, and a candidate signature* σ*, and outputs a bit* b*.*

Correctness *The scheme is* correct *if and only if, for all* $\lambda \in \mathbb{N}$*, all key-pairs* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{SGen}(1^\lambda)$*, all messages* $m \in \{0, 1\}^*$*, and all signatures* $\sigma \leftarrow \mathsf{SSig}(\mathsf{sk}, m)$*, it holds that* $\mathsf{SVer}(\mathsf{pk}, m, \sigma) = 1$*.*

Definition 11 (Existential Unforgeability). *A signature scheme* $\Sigma = (\mathsf{SGen}, \mathsf{SSig}, \mathsf{SVer})$ *is said to be* existentially unforgeable under chosen message attacks (EUF) *if and only if for all probabilistic polynomial-time adversaries* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that* $\Pr\left[\mathsf{EUF}^{\Sigma}_{\mathcal{A}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$ *where* $\mathsf{EUF}^{\Sigma}_{\mathcal{A}}$ *is the* existential unforgeability experiment *defined as follows:*

```
Experiment $\mathsf{EUF}^{\Sigma}_{\mathcal{A}}(\lambda)$ :
    $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{SGen}(1^\lambda)$; $Q := \emptyset$
    $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}(\mathsf{sk}, \cdot)}(\mathsf{pk})$
    If $\mathsf{SVer}(\mathsf{pk}, \sigma^*) = 1$ and $m^* \notin Q$
    Output 1; else output 0
```

```
$\mathcal{O}(\mathsf{sk}, m)$ :
    $Q := Q \cup \{m\}$
    $\sigma \leftarrow \mathsf{SSig}(\mathsf{sk}, m)$
    output $\sigma$
```

A.3 Extractable Public Key Encryption

We shortly recall the definitions of an extractable public key encryption scheme as well as the standard notion of CCA security. An extractable public key encryption scheme is a public key encryption scheme with an extra extraction algorithm which, on input a ciphertext and the randomness used for encryption, outputs the underlying message. Most, if not all, public key encryption schemes are extractable.

Definition 12 (Public Key Encryption Scheme). *An extractable public key encryption scheme* $\mathcal{E} = (\mathsf{EGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Ext})$ *consists of four efficient algorithms:*
$\mathsf{EGen}(1^\lambda)$*: The key generation algorithm takes the security parameter* 1^λ *and generates a key pair* $(\mathsf{dk}, \mathsf{ek})$*.*
$\mathsf{Enc}(\mathsf{ek}, m)$*: It takes an encryption key* ek *and a message* $m \in \{0, 1\}^*$*, and outputs a ciphertext* c*.*
$\mathsf{Dec}(\mathsf{dk}, c)$*: The decryption algorithm takes a decryption key* dk *and a ciphertext* c*, and outputs a message* m*.*
$\mathsf{Ext}(\rho, c)$*: It takes an encryption randomness* $\rho \in \chi$ *and a ciphertext* c*, and outputs a message* m*.*

Correctness *The scheme is* correct *if and only if for all* $\lambda \in \mathbb{N}$*, all* $(\mathsf{dk}, \mathsf{ek}) \leftarrow \mathsf{EGen}(1^\lambda)$*, all* $m \in \{0, 1\}^*$*, all* $\rho \in \{0, 1\}^\lambda$*, and all* $c \leftarrow \mathsf{Enc}(\mathsf{ek}, m; r)$*, it holds that* $m = \mathsf{Dec}(\mathsf{dk}, c) = \mathsf{Ext}(r, c)$*.*

Definition 13 (Indistinguishability under Chosen Ciphertext Attacks). *An extractable public key encryption scheme* $\mathcal{E} = (\mathsf{EGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Ext})$ *has* indistinguishable encryptions under chosen ciphertext attacks ($\mathsf{IND\text{-}CCA}$) *if for all (possibly stateful) PPT adversaries* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ *the probability that the experiment* $\mathsf{IND\text{-}CCA}^{\mathcal{E}}_{\mathcal{A}}(\lambda)$ *evaluates to* 1 *is negligibly bigger than* $\frac{1}{2}$ *(in* λ*), where*

```
Experiment $\mathsf{IND\text{-}CCA}^{\mathcal{E}}_{\mathcal{A}}(\lambda)$ :
    $(\mathsf{dk}, \mathsf{ek}) \leftarrow \mathsf{EGen}(1^\lambda)$; $b \leftarrow \{0, 1\}$
    $m_0, m_1 \leftarrow \mathcal{A}_0^{\mathsf{Dec}(\mathsf{dk}, \cdot)}(\mathsf{ek})$
    $c_b \leftarrow \mathsf{Enc}(\mathsf{ek}, m_b)$
    $a \leftarrow \mathcal{A}_1^{\mathsf{Dec}'(\mathsf{dk}, c_b, \cdot)}(c_b)$
    If $a = b$, output 1; else output 0
```

```
$\mathsf{Dec}'(\mathsf{dk}, c_b, c)$ :
    If $c \neq c_b$
    then output $\mathsf{Dec}(\mathsf{dk}, c)$
    else output $\perp$
```

A.4 Structure-Preserving CCA-Secure Encryption

We adopt the structure-preserving CCA-secure encryption scheme by Camenisch *et al.* [CHK$^+$11] in composite order group. Let $\mathbb{G}_2$ be a group with order $n = p \cdot q$ generated by g_2 where the DLIN assumption holds (for the sake of CCA security). Let $\hat{e} : \mathbb{G} \times \mathbb{G}_2 \to \mathbb{G}_{\hat{T}}$ be a non-degenerate efficiently computable bilinear map.

$\mathsf{EGen}(1^\lambda)$: The private key is a tuple of exponents $(\alpha_1, \alpha_2, \alpha_3, \{\beta_{i,1}, \beta_{i,2}, \beta_{i,3}\}_{i=0}^{5}) \in \mathbb{Z}_n^{21}$. The public key is a tuple of group elements $(\hat{g}_1, \hat{g}_2, \hat{g}_3, h_2, h_2, \{f_{i,1}, f_{i,2}\}_{i=0}^{5}) \in \mathbb{G}^{17}$ where $\hat{g}_1, \hat{g}_2, \hat{g}_3 \leftarrow \mathbb{G}_2$, $h_1 = \hat{g}_1^{\alpha_1} \hat{g}_3^{\alpha_3}$, $h_2 = \hat{g}_2^{\alpha_2} \hat{g}_3^{\alpha_3}$, $f_{i,1} = \hat{g}_1^{\beta_{i,1}} \hat{g}_3^{\beta_{i,3}}$, and $f_{i,2} = \hat{g}_1^{\beta_{i,2}} \hat{g}_3^{\beta_{i,3}}$.

$\mathsf{Enc}(\mathsf{pk}, L, m)$: To encrypt a message $m \in \mathbb{G}_2$ with label $L \in \mathbb{G}_2$, sample $r, s \leftarrow \mathbb{Z}_n$ and compute $c = (u_1, u_2, u_3, d, v) \in \mathbb{G}_2^4 \times \mathbb{G}_{\hat{T}}$, where $u_0 = g_2$, $u_1 = \hat{g}_1^r$, $u_2 = \hat{g}_2^s$, $u_3 = \hat{g}_3^{r+s}$, $d = m \cdot h_1^r h_2^s$, and $v = \prod_{i=0}^{3} e(f_{i,1}^r f_{i,2}^s, u_i) \cdot e(f_{4,1}^r f_{4,2}^s, d) \cdot e(f_{5,1}^r f_{5,2}^s, L)$.

$\mathsf{Dec}(\mathsf{sk}, c)$: To decrypt, check whether $v \overset{?}{=} \prod_{i=0}^{3} e(u_1^{\beta_{i,1}} u_2^{\beta_{i,2}} u_3^{\beta_{i,3}}, u_i) \cdot e(u_1^{\beta_{4,1}} u_2^{\beta_{4,2}} u_3^{\beta_{4,3}}, d) \cdot e(u_1^{\beta_{5,1}} u_2^{\beta_{5,2}} u_3^{\beta_{5,3}}, L)$ where $u_0 = g_2$. If so, output $m = d \cdot (u_1^{\alpha_1} u_2^{\alpha_2} u_3^{\alpha_3})^{-1}$.

A.5 Groth-Sahai Proof

Groth and Sahai [GS08] have proposed several instantiations for efficient non-interactive zero-knowledge (NIZK) proof of knowledge. The proof is about group elements satisfying a pairing product equation.

Definition 14. *An NIZK proof system $\mathcal{GS} = (\mathsf{GSSetup}, \mathsf{GSProv}, \mathsf{GSVer}, \mathsf{GSExt})$ is defined by:*

- $\mathsf{GSSetup}(1^\lambda)$*: The setup algorithm takes in the security parameter 1^λ and generates the common reference string crs and the extraction key xk of the proof system. All other algorithms take as input a bilinear group $\mathcal{G}$ specified externally and the common reference string crs. They are omitted for conciseness.*
- $\mathsf{GSProv}(stmt, wit)$*: The proving algorithm takes in a statement $stmt$ with one witness wit, and generates a proof of the validity of $stmt$ with respect to wit.*
- $\mathsf{GSVer}(stmt, \pi)$*: The verification algorithm takes in a statement $stmt$, and a proof π, and outputs* 1 *if π is a proof of $stmt$ with a valid witness,* 0 *otherwise.*
- $\mathsf{GSExt}(\mathsf{xk}, \pi)$*: It takes in the extraction key xk, and a proof π, and outputs the witness in π.*

Blum, Feldman and Micali [BFM88] introduced NIZK proofs, and defined its three properties below.

- Completeness: The probability of succeeding in proving a true statement is overwhelming.
- Soundness: The probability of succeeding in proving a false statement is negligible.
- Zero-knowledge: The proof gives no information but the validity of the theorem.

B Our Variant of Tag-based Trapdoor Functions

We construct our required variant of tag-based trapdoor functions using lattice-based trapdoor functions [MP12]. The construction is similar to the construction of a selectively secure signature scheme from lattice-based trapdoor functions [MP12, Section 6.2].

Informal Description. The user generates its public key consisting of a sequence of matrices $\mathbf{A}, \mathbf{A}_0, \ldots, \mathbf{A}_\ell$ for which it knows the lattice trapdoor of $\mathbf{A}$. Recall from [MP12] that, using the trapdoor of $\mathbf{A}$, the user can sample a short vector $\mathbf{v}$ such that $\mathbf{Av} = \mathbf{u}$ for any target vector $\mathbf{u}$, which is otherwise infeasible. Furthermore, the user can compute the trapdoor of $\mathbf{A}_\mu := [\mathbf{A} | \mathbf{A}_0 + \sum_{i \in [\ell]} \mu_i \mathbf{A}_i]$ for any $\mu \in \{0,1\}^\ell$. On the other hand, for any $\mu \in \{0,1\}^\ell$, any party can publicly sample a short vector $\mathbf{v}$ and compute a target vector $\mathbf{u} = \mathbf{Av}$.

Formal Description. Let $\mathbf{G} \in \mathbb{Z}_q^{n \times nk}$ be a gadget matrix [MP12] where $n = \mathsf{poly}(\lambda)\,(1^\lambda)$, $q = \mathsf{poly}(\lambda)\,(n)$ and $k = O(\log n)$. Let $\bar{m} = O(nk)$ and $m = \bar{m} + 2nk$. Let $s = O(\sqrt{\ell n k}) \cdot \omega(\sqrt{\log n})^2$ be a sufficient large Gaussian parameter. Using the formulation in [MP12], a matrix $\mathbf{R} \leftarrow D \sim \mathbb{Z}^{\bar{m} \times nk}$ is a trapdoor for the matrix $\mathbf{A} := [\bar{\mathbf{A}} | \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}]$. Denote $\mathbf{A}_\mu := [\mathbf{A} | \mathbf{A}_0 + \sum_{i \in [\ell]} \mu_i \mathbf{A}_i]$ for a bit string $\mu \in \{0,1\}^\ell$. Using $\mathbf{R}$, for any target vector $\mathbf{u}$, there exists efficient algorithm $\mathsf{SamPre}(\mathbf{R}, \mathbf{A}_\mu, \mathbf{u})$ which samples a preimage $\mathbf{v}$ such that $\mathbf{A}_\mu \mathbf{v} = \mathbf{u}$ and $\|\mathbf{v}\| \leq s \cdot \sqrt{m}$ [MP12]. We construct a tag-based trapdoor function TD as shown in Figure 8.

```
$\mathsf{TDGen}(1^\lambda)$
    $\bar{\mathbf{A}} \leftarrow \mathbb{Z}_q^{n\times\bar{m}}$
    $\mathbf{R} \leftarrow \mathbb{Z}^{\bar{m}\times nk}$
    $\mathbf{A} := [\bar{\mathbf{A}}|\mathbf{G} - \bar{\mathbf{A}}\mathbf{R}]$
    $\mathbf{A}_i \leftarrow \mathbb{Z}_q^{n\times nk}, i = 0,\dots,\ell$
    $\mathsf{pk} := (\mathbf{A}, \mathbf{A}_0, \dots, \mathbf{A}_\ell)$
    $\mathsf{sk} := \mathbf{R}$
    return $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{TDEval}(\mathsf{pk}, \mu, \rho = \mathbf{v})$
    $\mathbf{A}_\mu := [\mathbf{A}|\mathbf{A}_0 + \sum_{i\in[\ell]} \mu_i \mathbf{A}_i]$
    $\mathbf{u} \leftarrow \mathbf{A}_\mu \mathbf{v}$
    return $y := \mathbf{u}$

$\mathsf{TDInv}(\mathsf{sk}, \mu, y = \mathbf{u})$
    $\mathbf{A}_\mu := [\mathbf{A}|\mathbf{A}_0 + \sum_{i\in[\ell]} \mu_i \mathbf{A}_i]$
    $\mathbf{v} \leftarrow \mathsf{SamPre}(\mathbf{R}, \mathbf{A}_\mu, \mathbf{u})$
    return $\rho := \mathbf{v}$
```

Fig. 8. Our tag-based trapdoor function

Theorem 5. TD *supports domain and pre-image sampling.*

Proof. By [MP12, Theorem 5.5], the output of SamPre is within negligible statistical distance from $D_{\Lambda_{\mathbf{u}}^{\perp}(\mathbf{A}_\mu),s}$, and $D_{\mathbb{Z}^m,s}$ is efficiently samplable. □

Theorem 6. *If the* $\mathsf{SIS}_{q,\beta}$ *problem is hard for large enough* $\beta = O(\ell(nk)^{3/2}) \cdot \omega(\sqrt{\log n})^3$ *where* $\ell \geq 21^\lambda$*, then* TD *is collision-resistant under selective-tag adaptive-image attack.*

Proof. The following proof is essentially adapting the proof of unforgeability of the signature scheme constructed in [MP12, Section 6.2]. We therefore only repeat the essential details here.

Suppose there exists PPT adversary $\mathcal{A}$ which breaks the collision-resistance of TD. Consider a PPT simulator $\mathcal{S}$ which solves a random instance of $\mathsf{SIS}_{q,\beta}$. We first describe how $\mathcal{S}$ simulates the matrices $(\mathbf{A}, \mathbf{A}_0, \dots, \mathbf{A}_\ell)$ in the public key.

$\mathcal{S}$ receives an $\mathsf{SIS}_{q,\beta}$ instance given by $\mathbf{A} = [\bar{\mathbf{A}}|\mathbf{B}] \in \mathbb{Z}_q^{n\times(\bar{m}+nk)}$ and syndrome $\mathbf{u}' \in \mathbb{Z}_q^n$. It will use $\mathcal{A}$ to find some $\mathbf{z} \in \mathbb{Z}^m$ of length $\|\mathbf{z}\| \leq \beta - 1$ such that $\mathbf{A}\mathbf{z} = \mathbf{u}'$ or non-zero $\mathbf{z} \in \mathbb{Z}^m$ of length $\|\mathbf{z}\| \leq \beta$ such that $\mathbf{A}\mathbf{z} = \mathbf{0}$. In either case, it can find $\mathbf{z}' \in \mathbb{Z}m + 1$ of length $\|\mathbf{z}\| \leq \beta$ such that $[\mathbf{A}|\mathbf{u}']\mathbf{z}' = \mathbf{0}$.

At the beginning, $\mathcal{A}$ sends distinct μ_i for $i = 1, \dots, Q$ to $\mathcal{S}$. $\mathcal{S}$ computes the set P of all strings $p \in \{0,1\}^{\leq \ell}$ such that p is a shortest string for which no μ_i has p as a prefix. P can be equivalently viewed as the set of maximal subtrees of $\{0,1\}^{\leq \ell}$ (viewed as a tree) that do not contain any of the μ_i's. P can be computed efficiently [CHKP10] and has size bounded above by $(\ell+1)Q + 1$. Choose $p \leftarrow P$ and let $t = |p| \leq \ell$.

It constructs $(\mathbf{A}, \mathbf{A}_0, \dots, \mathbf{A}_\ell)$ as follows: For $j = 0, \dots, \ell$, choose $\mathbf{R}_j \leftarrow D^{\bar{m}\times nk}_{\mathbb{Z},\omega(\sqrt{\log n})}$ and let

$$\mathbf{A}_j = \mathbf{H}_j\mathbf{G} - \bar{\mathbf{A}}\mathbf{R}_j, \text{ where } \mathbf{H}_j = \begin{cases} h(0) = \mathbf{0} & j > t \\ (-1)^{p_j} \cdot h(u_j) & j \in [t] \\ -\sum_{l\in[t]} p_l \cdot \mathbf{H}_l & j = 0 \end{cases}$$

where $u_1, \dots, u_t$ are units in $\mathbb{Z}_q[x]/(f(x))$ for some monic degree-n polynomial f irreducible over every prime p dividing q such that all non-trivial subset sum of $u_1, \dots, u_t$ is also a unit, and $h : \mathbb{Z}_q[x]/(f(x)) \to \mathbb{Z}_q^{n\times n}$ is an injective ring homomorphism, so that a is a unit in $\mathbb{Z}_q[x]/(f(x))$ if and only if $h(a)$ in $\mathbb{Z}_q^{n\times n}$ is invertible.

$\mathcal{S}$ sends $\mathsf{pk} := (\mathbf{A}, \mathbf{A}_0, \dots, \mathbf{A}_\ell)$ to $\mathcal{A}$. Upon receiving the i-th query $y_i = \mathbf{u}_i$, $\mathcal{S}$ computes $\mathbf{A}_{\mu_i} := [\mathbf{A}|\mathbf{A}_0 + \sum_{j\in[\ell]} \mu_{i,j}\mathbf{A}_j] = [\mathbf{A}|\mathbf{H}\mathbf{G} - \bar{\mathbf{A}}\mathbf{R}]$, where $\mathbf{H} = h(\sum_{j\in[t],\mu_{i,j}\neq p_j} u_j)$ is invertible as p is not a prefix of any μ_i, and $\mathbf{R} = (\mathbf{R}_0 + \sum_{j\in[\ell]} \mu_{i,j}\mathbf{R}_j)$. From [MP12], $\mathbf{R}$ is a trapdoor of $\mathbf{A}_{\mu_i}$ and $\mathcal{S}$ can sample and output $\mathbf{v}_i \leftarrow \mathsf{SamPre}(\mathbf{R}, \mathbf{A}_{\mu_i}, \mathbf{u}_i)$.

Eventually, a successful $\mathcal{A}$ will output distinct $(\mu_1^*, \mathbf{v}_1^*)$ and $(\mu_2^*, \mathbf{v}_2^*)$ such that $\mu_1^*, \mu_2^* \notin (\mu_1, \dots, \mu_Q)$ and $\mathbf{A}_{\mu_1^*}\mathbf{v}_1^* = \mathbf{A}_{\mu_2^*}\mathbf{v}_2^*$. $\mathcal{S}$ computes $\mathbf{R}_1^* = (\mathbf{R}_0 + \sum_{j\in[\ell]} \mu_{1,j}^*\mathbf{R}_j)$ and $\mathbf{R}_2^* = (\mathbf{R}_0 + \sum_{j\in[\ell]} \mu_{2,j}^*\mathbf{R}_j)$, and outputs

$$\mathbf{z} = \begin{bmatrix} \mathbf{I}_{\bar{m}} & -\mathbf{R}_1^* \\ & \mathbf{I}_{nk} \end{bmatrix} \mathbf{v}_1^* - \begin{bmatrix} \mathbf{I}_{\bar{m}} & -\mathbf{R}_2^* \\ & \mathbf{I}_{nk} \end{bmatrix} \mathbf{v}_2^*$$

We argue that $\mathbf{z}$ is a valid solution to the $\mathsf{SIS}_{q,\beta}$ instance.

Suppose both μ^* and μ has prefix p, which happens with probability at least $1/((\ell-1)Q+1)^2 - \mathsf{negl}(\lambda)(1^\lambda)$ since we have assumed $\mu_1^* \neq \mu_i$ and $\mu_2^* \neq \mu_i$ for all i. Then $\mathbf{A}_{\mu_1^*} = [\mathbf{A}| - \bar{\mathbf{A}}\mathbf{R}_1^*]$ where $\mathbf{R}_1^* = (\mathbf{R}_0 + \sum_{j\in[\ell]} \mu_{1,j}^* \mathbf{R}_j)$. Similarly $\mathbf{A}_{\mu_2^*} = [\mathbf{A}| - \bar{\mathbf{A}}\mathbf{R}_2^*]$ where $\mathbf{R}_2^* = (\mathbf{R}_0 + \sum_{j\in[\ell]} \mu_{2,j}^* \mathbf{R}_j)$. In other words, we have $\mathbf{A}\mathbf{z} = \mathbf{0}$, *i.e.*,

$$[\bar{\mathbf{A}}|\mathbf{B}]\left(\begin{bmatrix}\mathbf{I}_{\bar{m}} & -\mathbf{R}_1^* \\ & \mathbf{I}_{nk}\end{bmatrix}\mathbf{v}_1^* - \begin{bmatrix}\mathbf{I}_{\bar{m}} & -\mathbf{R}_2^* \\ & \mathbf{I}_{nk}\end{bmatrix}\mathbf{v}_2^*\right) = \mathbf{0}$$

Since both $\|\mathbf{v}_1^*\|, \|\mathbf{v}_2^*\| \leq s \cdot \sqrt{m} = O(\sqrt{\ell nk}) \cdot \omega(\sqrt{\log n})^2$, and the maximum singular values of $\mathbf{R}_1^*$ and $\mathbf{R}_2^*$ satisfies $s_1(\mathbf{R}_k^*) = O(\sqrt{\ell nk}) \cdot \omega(\sqrt{\log n})$ for $k = 1, 2$ with overwhelming probability, we have $\|\mathbf{z}\| = O(\ell(nk)^{\frac{3}{2}}) \cdot \omega(\sqrt{\log n})^3$ as required. We refer the readers to the proof of [MP12, Theorem 6.1] for details and the proof that $\mathbf{z} \neq \mathbf{0}$. □

C Accountable Ring Signatures

The definition of accountable ring signature is taken from Bootle *et al.* [BCC+15]

C.1 Definition of Accountable Ring Signatures

Definition 15 (Accountable Ring Signatures). *An accountable ring signature scheme is a tuple of seven polynomial-time algorithms* $\mathcal{RS} = (\mathsf{RSetup}, \mathsf{ROKGen}, \mathsf{RUKGen}, \mathsf{RSig}, \mathsf{RVer}, \mathsf{ROpen}, \mathsf{RJud})$.

$\mathsf{RSetup}(1^\lambda) \to \mathsf{pp}$*: given a security parameter* λ*, this algorithm generates the system parameters* pp.

$\mathsf{ROKGen}(\mathsf{pp}) \to (\mathsf{opk}, \mathsf{osk})$*: given the system parameters* pp*, this algorithm creates a key pair* $(\mathsf{opk}, \mathsf{osk})$ *for the opener to trace the signer of a ring signature. We assume that* opk *is uniquely determined by* pp *and* osk*, denoted as* $\mathsf{opk} \leftarrow \mathsf{ROKGen}(\mathsf{pp}, \mathsf{osk})$.

$\mathsf{RUKGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$*: given the system parameters* pp*, this algorithm creates a key pair* $(\mathsf{pk}, \mathsf{sk})$ *for a signer.*

$\mathsf{RSig}(\mathsf{opk}, m, \mathcal{R}, \mathsf{sk}) \to \sigma$*: given the public opening key* opk*, a message* m*, a ring of signers* $\mathcal{R}$*, and a signing key* sk *of a member in* $\mathcal{R}$*, this algorithm creates a ring signature on* m*.*

$\mathsf{RVer}(\mathsf{opk}, m, \mathcal{R}, \sigma) \to b$*: given the public opening key* opk*, a message* m*, a ring of signers* $\mathcal{R}$*, and a candidate ring signature* σ*, this algorithm output a bit* b *indicating the validity of* σ*.*

$\mathsf{ROpen}(\mathsf{osk}, m, \mathcal{R}, \sigma) \to (\mathsf{pk}^*, \psi)/\bot$*: given an opener private key* osk*, a message* m*, a ring* $\mathcal{R}$*, and a ring signature* σ*, this algorithm returns a verification key* pk^* *and a proof* ψ *that the owner of* pk^* *produced* σ*. If any of the inputs is invalid, it returns* $\bot$*.*

$\mathsf{RJud}(\mathsf{opk}, m, \mathcal{R}, \sigma, \mathsf{pk}^*, \psi) \to b$*: given an opener public key* opk*, a message* m*, a ring* $\mathcal{R}$*, a ring signature* σ*, a signer public key* pk^**, and a proof* ψ*, this algorithm returns a bit* b*. When* ψ *is not accepted, or any of the inputs is invalid,* $b = 0$*; otherwise,* $b = 1$*.*

C.2 Security of Accountable Ring Signatures

An accountable ring signature scheme should be correct, fully unforgeable, anonymous, traceable, and has tracing soundness.

Correctness. The scheme is *correct* if and only if, for all $\lambda \in \mathbb{N}$, all $\mathsf{pp} \in \mathsf{RSetup}(1^\lambda)$, all opener key pairs $(\mathsf{opk}, \mathsf{osk}) \leftarrow \mathsf{ROKGen}(\mathsf{pp})$, all user key-pairs $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{RUKGen}(\mathsf{pp})$, messages $m \in \{0,1\}^*$, any subset of signers $\mathcal{R}$, and all signatures $\sigma \in \mathsf{RSig}(\mathsf{opk}, m, \mathcal{R}, \mathsf{sk})$, it holds that $\mathsf{SVer}(\mathsf{opk}, m, \mathcal{R}, \sigma) = 1$.

Definition 16 (Full Unforgeability). *An accountable ring signature scheme* $\mathcal{RS}$ *is* fully unforgeable *if for any PPT adversaries* $\mathcal{A}$ *the probability that the experiment* $\mathsf{Unforgeability}_{\mathcal{A}}^{\mathcal{RS}}(\lambda)$ *evaluates to* 1 *is negligible (in* λ*), where*

Experiment $\mathsf{Unforgeability}_{\mathcal{A}}^{\mathcal{RS}}(\lambda)$

$\mathsf{pp} \leftarrow \mathsf{RSetup}(1^\lambda)$; $(\mathsf{opk}, \mathsf{pk}, m, \mathcal{R}, \sigma, \psi) \leftarrow \mathcal{A}^{\mathsf{RUKGen}, \mathsf{RSig}, \mathsf{Reveal}}(\mathsf{pp})$

Output 1 *if one of the following cases holds:*

1. $\mathsf{pk} \in Q_{\mathsf{RUKGen}} \setminus Q_{\mathsf{Reveal}}$, $(\mathsf{opk}, m, \mathcal{R}, \sigma, \mathsf{pk}) \notin Q_{\mathsf{RSig}}$, *and* $\mathsf{RJud}(\mathsf{opk}, m, \mathcal{R}, \sigma, \mathsf{pk}, \psi) = 1$.
2. $\mathcal{R} \subset Q_{\mathsf{RUKGen}} \setminus Q_{\mathsf{Reveal}}$, $(\mathsf{opk}, m, \mathcal{R}, \sigma, \cdot) \notin Q_{\mathsf{RSig}}$, *and* $\mathsf{RVer}(\mathsf{opk}, m, \mathcal{R}, \sigma) = 1$.

Otherwise, output 0.

- RUKGen *runs* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{RUKGen}(\mathsf{pp})$ *and returns* pk. Q_{RUKGen} *is the set of verification keys* pk *that have been generated by this oracle.*
- RSig *is an oracle that on query* $(\mathsf{opk}, m, \mathcal{R}, \mathsf{pk})$ *returns* $\sigma \leftarrow \mathsf{RSig}(\mathsf{opk}, m, \mathcal{R}, \mathsf{sk})$ *if* $\mathsf{pk} \in \mathcal{R} \cap Q_{\mathsf{RUKGen}}$. Q_{RSig} *contains the queries and responses* $(\mathsf{opk}, m, \mathcal{R}, \sigma, \mathsf{pk})$.
- Reveal *is an oracle that when queried on* $\mathsf{pk} \in Q_{\mathsf{RUKGen}}$ *returns the corresponding signing key* sk. Q_{Reveal} *is the list of verification keys* pk *for which the corresponding signing key has been revealed.*

Definition 17 (Anonymity against Full Key Exposure). *An accountable ring signature scheme* $\mathcal{RS}$ *is* anonymous *if for any PPT adversaries* $\mathcal{A}$ *the probability that the experiment* $\mathsf{Anon}_{\mathcal{A}}^{\mathcal{RS}}(\lambda)$ *evaluates to* 1 *is negligibly close to* $\frac{1}{2}$ *(in* λ*), where*

Experiment $\mathsf{Anon}_{\mathcal{A}}^{\mathcal{RS}}(\lambda)$

$\mathsf{pp} \leftarrow \mathsf{RSetup}(1^\lambda)$; $(\mathsf{opk}, \mathsf{osk}) \leftarrow \mathsf{ROKGen}(\mathsf{pp})$; $b \leftarrow \{0, 1\}$

$b' \leftarrow \mathcal{A}^{\mathsf{Chal}_b, \mathsf{Open}}(\mathsf{pp}, \mathsf{opk})$

Output 1 *if* $b = b'$, *otherwise, output* 0.

- Chal_b *is an oracle that the adversary can only call once. On query* $(m, \mathcal{R}, i_0, i_1)$, *it runs* $\sigma_0 \leftarrow \mathsf{RSig}(\mathsf{opk},\ m,\ \mathcal{R},\ \mathsf{sk}_{i_0})$ *and* $\sigma_1 \leftarrow \mathsf{RSig}(\mathsf{opk},\ m,\ \mathcal{R},\ \mathsf{sk}_{i_1})$. *If* $\sigma_0 \neq \perp$ *and* $\sigma_1 \neq \perp$ *it returns* σ_b, *otherwise, it returns* $\perp$. i_0 *and* i_1 *are two indices of the signer in* $\mathcal{R}$.
- Open *is an oracle that on query* $(m, \mathcal{R}, \sigma)$ *returns* $(\mathsf{pk}, \psi) \leftarrow \mathsf{ROpen}(\mathsf{osk}, m, \mathcal{R}, \sigma)$. *If* σ *was obtained by calling* Chal_b *on* $(m, \mathcal{R})$, *the oracle returns* $\perp$.

Definition 18 (Traceability). *An accountable ring signature scheme* $\mathcal{RS}$ *is* traceable *if for any PPT adversaries* $\mathcal{A}$ *the probability that the experiment* $\mathsf{Trace}_{\mathcal{A}}^{\mathcal{RS}}(\lambda)$ *evaluates to* 1 *is negligible (in* λ*), where*

Experiment $\mathsf{Trace}_{\mathcal{A}}^{\mathcal{RS}}(\lambda)$

$\mathsf{pp} \leftarrow \mathsf{RSetup}(1^\lambda)$; $(\mathsf{osk}, m, \mathcal{R}, \sigma) \leftarrow \mathcal{A}(\mathsf{pp})$; $\mathsf{opk} \leftarrow \mathsf{ROKGen}(\mathsf{pp}, \mathsf{osk})$; $(\mathsf{pk}, \psi) \leftarrow \mathsf{ROpen}(\mathsf{osk}, m, \mathcal{R}, \sigma)$

Output 1 *if* $\mathsf{RVer}(\mathsf{opk}, m, \mathcal{R}, \sigma) = 1 \wedge \mathsf{RJud}(\mathsf{opk}, m, \mathcal{R}, \sigma, \mathsf{pk}, \psi) = 0$

Otherwise, output 0.

Definition 19 (Tracing Soundness). *An accountable ring signature scheme* $\mathcal{RS}$ *has* tracing soundness *if for any PPT adversaries* $\mathcal{A}$ *the probability that the experiment* $\mathsf{TraceSound}_{\mathcal{A}}^{\mathcal{RS}}(\lambda)$ *evaluates to* 1 *is negligible (in* λ*), where*

Experiment $\mathsf{TraceSound}_{\mathcal{A}}^{\mathcal{RS}}(\lambda)$

$\mathsf{pp} \leftarrow \mathsf{RSetup}(1^\lambda)$

$(m, \sigma, \mathcal{R}, \mathsf{opk}, \mathsf{pk}_1, \mathsf{pk}_2, \psi_1, \psi_2) \leftarrow \mathcal{A}(\mathsf{pp})$

Output 1 *if, for all* $i = 1, 2$, $\mathsf{RJud}(\mathsf{opk}, m, \mathcal{R}, \sigma, \mathsf{pk}_i, \psi_i) = 1 \wedge \mathsf{pk}_1 \neq \mathsf{pk}_2$

Otherwise, output 0.

D Sanitizable Signatures

D.1 Definition of Sanitizable Signatures

The following definition of sanitizable signature schemes is slightly modified from [BFF+09, BFLS10].

Definition 20 (Sanitizable Signature Scheme). *A sanitizable signature scheme* $\mathcal{SS} = (\mathsf{KGen}_\mathsf{S}, \mathsf{KGen}_\mathsf{Z}, \mathsf{Sig}, \mathsf{San}, \mathsf{Ver}, \mathsf{Prov}, \mathsf{Jud})$ *consists of eight algorithms:*

Key Generation. *The setup algorithm creates a public parameter for key generations:* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$. *There are two key generation algorithms, one for the signer and one for the sanitizer. Both create a public/private key pair:* $(\mathsf{pk}_\mathsf{S}, \mathsf{sk}_\mathsf{S}) \leftarrow \mathsf{KGen}_\mathsf{S}(\mathsf{pp})$, $(\mathsf{pk}_\mathsf{Z}, \mathsf{sk}_\mathsf{Z}) \leftarrow \mathsf{KGen}_\mathsf{Z}(\mathsf{pp})$.

Signing. *The signing algorithm takes as input a message $m \in \{0,1\}^*$, a signer private key $\mathsf{sk_S}$, a sanitizer public key $\mathsf{pk_Z}$, as well as a description α of the admissible modifications to m by the sanitizer and outputs a signature $\sigma \leftarrow \mathsf{Sig}(\mathsf{sk_S}, \mathsf{pk_Z}, m, \alpha)$. We assume that α can be recovered from any σ.*

Sanitizing. *The sanitizing algorithm takes as input a message $m \in \{0,1\}^*$, a description δ of the desired modifications to m, a signature σ, the signer public key $\mathsf{pk_S}$, and a sanitizer private key $\mathsf{sk_Z}$. It modifies the message m according to the modification instruction δ, and outputs a new signature σ' for the modified message $m' = \delta(m)$ or possibly $\perp$ in case of an error, i.e., $\{(m', \sigma'), \perp\} \leftarrow \mathsf{San}(\mathsf{pk_S}, \mathsf{sk_Z}, m, \delta, \sigma)$.*

Verification. *The verification algorithm takes as input a message m, a candidate signature σ, a signer public key $\mathsf{pk_S}$, as well as a sanitizer public key $\mathsf{pk_Z}$ and outputs a bit b, i.e. $b \leftarrow \mathsf{Ver}(\mathsf{pk_S}, \mathsf{pk_Z}, m, \sigma)$.*

Proof. *The proof algorithm takes as input a signer private key $\mathsf{sk_S}$, a message m, a signature σ, and a sanitizer public key $\mathsf{pk_Z}$ and outputs a proof π, i.e. $\pi \leftarrow \mathsf{Prov}(\mathsf{sk_S}, \mathsf{pk_Z}, m, \sigma)$.*

Judge. *The judge algorithm takes as input a message m, a signature σ, signer and sanitizer public keys $\mathsf{pk_S}, \mathsf{pk_Z}$, and proof π. It outputs a decision $d \in \{\mathsf{S}, \mathsf{Z}\}$ indicating whether the message-signature pair was created by the signer or the sanitizer, i.e. $d \leftarrow \mathsf{Jud}(\mathsf{pk_S}, \mathsf{pk_Z}, m, \sigma, \pi)$.*

For a sanitizable signature scheme the usual correctness properties should hold, saying that genuinely signed or sanitized messages are accepted and that a genuinely created proof by the signer leads the judge to decide in favor of the signer. For a formal approach to correctness see [BFF+09].

D.2 Security of Sanitizable Signatures

Here we recall the security notions of sanitizable signatures given by Brzuska *et al.* [BFF+09, BFLS10], namely, unforgeability, privacy, immutability, accountability, transparency, and unlinkability. It is known that signer and sanitizer accountability together implies unforgeability and that unlinkability implies privacy. On the other hand, (proof-restricted) transparency implies (proof-restricted) privacy. Since both of our schemes satisfy signer and sanitizer accountability, we omit the definition of unforgeability and privacy.

Immutability. Informally, this property says that a malicious sanitizer cannot change inadmissible blocks. This is formalized in a model where the malicious sanitizer $\mathcal{A}$ interacts with the signer to obtain signatures σ_i for messages m_i, descriptions α_i and keys $\mathsf{pk}_{\mathsf{Z},i}$ of its choice. Eventually, the attacker stops, outputting a valid pair $(\mathsf{pk}_\mathsf{Z}^*, m^*, \sigma^*)$ such that message m^* is not a "legitimate" transformation of one of the m_i's under the same key $\mathsf{pk}_\mathsf{Z}^* = \mathsf{pk}_{\mathsf{Z},i}$. The latter is formalized by requiring that for each query $\mathsf{pk}_\mathsf{Z}^* \neq \mathsf{pk}_{\mathsf{Z},i}$ or $m^* \notin \{\delta(m_i) \mid \delta \text{ with } \alpha_i(\delta) = 1\}$ for the value α_i in σ_i. This requirement enforces that block-divided messages m^* and m_i differ by at least one inadmissible block. Observe that this definition covers also the case where the adversary interacts with several sanitizers simultaneously, because it can query the signer for several sanitizer keys $\mathsf{pk}_{\mathsf{Z},i}$.

Definition 21 (Immutability). *A sanitizable signature scheme $\mathcal{SS}$ is said to be* immutable *if for all PPT adversaries $\mathcal{A}$ the probability that the experiment $\mathsf{Immut}_\mathcal{A}^{\mathcal{SS}}(\lambda)$ evaluates to 1 is negligible (in λ), where*

```
Experiment $\mathsf{Immut}_\mathcal{A}^{\mathcal{SS}}(\lambda)$
    $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$; $(\mathsf{pk_S}, \mathsf{sk_S}) \leftarrow \mathsf{KGen_S}(\mathsf{pp})$
    $(\mathsf{pk}_\mathsf{Z}^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sig}(\mathsf{sk_S},\cdot,\cdot,\cdot), \mathsf{Prov}(\mathsf{sk_S},\cdot,\cdot,\cdot)}(\mathsf{pp}, \mathsf{pk_S})$
        where $(\mathsf{pk}_{\mathsf{Z},i}, m_i, \alpha_i)$ and $\sigma_i$ denote the queries and answers to and from oracle Sig.
    Output 1 if $\mathsf{Ver}(\mathsf{pk_S}, \mathsf{pk}_\mathsf{Z}^*, m^*, \sigma^*) = 1$ and for all $i$ the following holds:
            $\mathsf{pk}_\mathsf{Z}^* \neq \mathsf{pk}_{\mathsf{Z},i} \lor m^* \notin \{\delta(m_i) \mid \delta \text{ with } \alpha_i(\delta) = 1\}$
    Else output 0.
```

Accountability. This property demands that the origin of a (possibly sanitized) signature should be undeniable. We distinguish between *sanitizer-accountability* and *signer-accountability* and formalize each security property in the following. *Signer-accountability* says that, if a message and its signature have not been sanitized, then even a malicious signer should not be able to make the judge accuse the sanitizer.

In the sanitizer-accountability game let $\mathcal{A}_{\mathsf{San}}$ be an adversary playing the role of the malicious sanitizer. $\mathcal{A}_{\mathsf{San}}$ has access to Sig and Prov oracle and it succeeds if it outputs a valid message signature pair such that m^*, σ^*, together with a key pk_Z^* (with $(\mathsf{pk}_\mathsf{Z}^*, m^*)$ such that the output is different from pairs $(\mathsf{pk}_{\mathsf{Z},i}, m_i)$ previously queried to the Sig oracle). Moreover, it is required that the proof produced by the signer via Prov still leads the judge to decide "$\mathtt{S}$", *i.e.*, that the signature has been created by the signer.

Definition 22 (Sanitizer-Accountability). *A sanitizable signature scheme* $\mathcal{SS}$ *is* sanitizer-accountable *if for all PPT adversaries* $\mathcal{A}$ *the probability that the experiment* $\mathsf{San\text{-}Acc}_\mathcal{A}^\mathcal{SS}(\lambda)$ *evaluates to* 1 *is negligible (in* λ*), where*

```
Experiment $\mathsf{San\text{-}Acc}_\mathcal{A}^\mathcal{SS}(\lambda)$
    $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$; $(\mathsf{pk}_\mathsf{S}, \mathsf{sk}_\mathsf{S}) \leftarrow \mathsf{KGen}_\mathsf{S}(\mathsf{pp})$
    $(\mathsf{pk}_\mathsf{Z}^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sig}(\mathsf{sk}_\mathsf{S},\cdot,\cdot,\cdot),\mathsf{Prov}(\mathsf{sk}_\mathsf{S},\cdot,\cdot,\cdot)}(\mathsf{pp}, \mathsf{pk}_\mathsf{S})$
        where $(m_i, \alpha_i, \mathsf{pk}_{\mathsf{Z},i})$ and $\sigma_i$ denote the queries and answers to and from oracle $\mathsf{Sig}$
    $\pi \leftarrow \mathsf{Prov}(\mathsf{sk}_\mathsf{S}, \mathsf{pk}_\mathsf{Z}^*, m^*, \sigma^*)$
    Output 1 if for all $i$ the following holds:
        $(\mathsf{pk}_\mathsf{Z}^*, m^*) \neq (\mathsf{pk}_{\mathsf{Z},i}, m_i) \wedge \mathsf{Ver}(\mathsf{pk}_\mathsf{S}, \mathsf{pk}_\mathsf{Z}^*, m^*, \sigma^*) = 1 \wedge \mathsf{Jud}(\mathsf{pk}_\mathsf{S}, \mathsf{pk}_\mathsf{Z}^*, m^*, \sigma^*, \pi) \neq \mathtt{Z}$
    else output 0.
```

In the signer-accountability game a malicious signer $\mathcal{A}_\mathsf{Sig}$ gets a public sanitizing key pk_Z as input and has access to a sanitizing oracle, which takes as input tuples $(m_i, \delta_i, \sigma_i, \mathsf{pk}_{\mathsf{S},i})$ and returns (m_i', σ_i'). Eventually, the adversary $\mathcal{A}_\mathsf{Sig}$ outputs a tuple $(\mathsf{pk}_\mathsf{S}^*, m^*, \sigma^*, \pi^*)$ and is considered successful if Jud accuses the sanitizer for the new key-message pair $\mathsf{pk}_\mathsf{S}^*, m^*$ with a valid signature σ^*.

Definition 23 (Signer-Accountability). *A sanitizable signature scheme* $\mathcal{SS}$ *is said to be* signer-accountable *if for all PPT adversaries* $\mathcal{A}$ *the probability that the experiment* $\mathsf{Sig\text{-}Acc}_\mathcal{A}^\mathcal{SS}(\lambda)$ *outputs* 1 *is negligible (in* λ*), where*

```
Experiment $\mathsf{Sig\text{-}Acc}_\mathcal{A}^\mathcal{SS}(\lambda)$
    $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$; $(\mathsf{pk}_\mathsf{Z}, \mathsf{sk}_\mathsf{Z}) \leftarrow \mathsf{KGen}_\mathsf{Z}(\mathsf{pp})$
    $(\mathsf{pk}_\mathsf{S}^*, m^*, \sigma^*, \pi^*) \leftarrow \mathcal{A}^{\mathsf{San}(\cdot,\mathsf{sk}_\mathsf{Z},\cdot,\cdot,\cdot)}(\mathsf{pp}, \mathsf{pk}_\mathsf{Z})$
        where $(\mathsf{pk}_{\mathsf{S},i}, m_i, \delta_i, \sigma_i)$ and $(m_i', \sigma_i')$ denote the queries and answers to and from oracle $\mathsf{San}$.
    Output 1 if for all $i$ the following holds:
        $(\mathsf{pk}_\mathsf{S}^*, m^*) \neq (\mathsf{pk}_{\mathsf{S},i}, m_i') \wedge \mathsf{Ver}(\mathsf{pk}_\mathsf{S}^*, \mathsf{pk}_\mathsf{Z}, m^*, \sigma^*) = 1 \wedge \mathsf{Jud}(\mathsf{pk}_\mathsf{S}^*, \mathsf{pk}_\mathsf{Z}, m^*, \sigma^*, \pi^*) \neq \mathtt{S}$
    else output 0.
```

Transparency. Informally, this property says that one cannot decide whether a signature has been sanitized or not. Formally, this is defined in a game where an adversary $\mathcal{A}$ has access to Sig, San, and Prov oracles with which the adversary can create signatures for (sanitized) messages and learn proofs. In addition, $\mathcal{A}$ gets access to a $\mathsf{Sig/San}$ box which contains a secret random bit $b \in \{0, 1\}$ and which, on input a message m, a modification information δ and a description α behaves as follows:

- for $b = 0$ runs the signer algorithm to create $\sigma \leftarrow \mathsf{Sig}(m, \mathsf{sk}_\mathsf{S}, \mathsf{pk}_\mathsf{S}, \alpha)$, then runs the sanitizer algorithm and returns the sanitized message m' with the new signature σ', and
- for $b = 1$ acts as in the case $b = 0$ but also signs m' from scratch with the signing algorithm to create a signature σ' and returns the pair (m', σ').

Adversary $\mathcal{A}$ eventually produces an output a, the guess for b. A sanitizable signature is now *transparent* if for all efficient algorithms $\mathcal{A}$ the probability for a right guess $a = b$ in the above game is negligibly close to $\frac{1}{2}$. Below we also define a relaxed version called *proof-restricted transparency.*

Definition 24 ((Proof-Restricted) Transparency). *A sanitizable signature scheme* $\mathcal{SS}$ *is said to be* proof-restrictedly transparent *if for all PPT adversaries* $\mathcal{A}$ *the probability that the experiment* $\mathsf{Trans}_\mathcal{A}^\mathcal{SS}(\lambda)$ *evaluates to* 1 *is negligibly (in* λ*) bigger than* $\frac{1}{2}$*, where*

```
Experiment $\mathsf{Trans}_\mathcal{A}^\mathcal{SS}(\lambda)$
  $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$; $(\mathsf{pk}_\mathsf{S}, \mathsf{sk}_\mathsf{S}) \leftarrow \mathsf{KGen}_\mathsf{S}(\mathsf{pp})$; $(\mathsf{pk}_\mathsf{Z}, \mathsf{sk}_\mathsf{Z}) \leftarrow \mathsf{KGen}_\mathsf{Z}(\mathsf{pp})$; $b \leftarrow \{0, 1\}$
  $a \leftarrow \mathcal{A}^{\mathsf{Sig}(\mathsf{sk}_\mathsf{S},\cdot,\cdot,\cdot),\mathsf{San}(\cdot,\mathsf{sk}_\mathsf{Z},\cdot,\cdot,\cdot),\mathsf{Prov}(\mathsf{sk}_\mathsf{S},\cdot,\cdot,\cdot),\mathsf{Sig/San}(\cdot,\cdot,\cdot,\mathsf{sk}_\mathsf{S},\mathsf{sk}_\mathsf{Z},b)}(\mathsf{pp}, \mathsf{pk}_\mathsf{S}, \mathsf{pk}_\mathsf{Z})$
   where $M_{\mathsf{Sig/San}}$ and $M_\mathsf{Prov}$ denote the sets of messages output by the $\mathsf{Sig/San}$
   and queried to the $\mathsf{Prov}$ oracle respectively.
  Output 1 if $\big(a = b \wedge M_{\mathsf{Sig/San}} \cap M_\mathsf{Prov} = \emptyset\big)$; else output 0
```

Unlinkability. This security notion demands that it is not feasible to use the signatures to identify sanitized message-signature pairs originating from the same source. This should even hold if the adversary itself provides the two source message-signature pairs and modifications of which one is sanitized. It is required that the two modifications yield the same sanitized message, because otherwise predicting the source is easy, of course. This, however, is beyond the scope of signature schemes: the scheme should only prevent that *signatures* can be used to link data.

In the formalization of [BFLS10], the adversary can access a signing oracle and a sanitizer oracle (and a proof oracle since this step depends on the signer private key and may leak valuable information). The adversary is also allowed to query a left-or-right oracle $\mathsf{LoRSanit}$ which is initialized with a secret random bit b and keys $\mathsf{pk_S}, \mathsf{sk_Z}$. The adversary may query this oracle on tuples $((m_0, \delta_0, \sigma_0), (m_1, \delta_1, \sigma_1))$ and returns $\mathsf{San}(m_b, \delta_b, \sigma_b, \mathsf{pk_S}, \mathsf{sk_Z})$ if $\mathsf{Ver}(m_i, \sigma_i, \mathsf{pk_S}, \mathsf{pk_Z}) = 1$ for $i = 0, 1$, $\alpha_0 = \alpha_1$ and if the modifications map to the same message, *i.e.*, $\alpha_0(\delta_0) = 1$, $\alpha_1(\delta_1) = 1$ and $\delta_0(m_0) = \delta_1(m_1)$. Otherwise, the oracle returns $\perp$. The adversary should eventually predict the bit b significantly better than with the guessing probability of $\frac{1}{2}$.

Definition 25 (Unlinkability). *A sanitizable signature scheme $\mathcal{SS}$ is* unlinkable *if for all PPT adversaries $\mathcal{A}$ the probability that the experiment $\mathsf{Link}_{\mathcal{A}}^{\mathcal{SS}}(\lambda)$ outputs 1 is negligibly (in λ) bigger than $\frac{1}{2}$, where*

Experiment $\mathsf{Link}_{\mathcal{A}}^{\mathcal{SS}}(\lambda)$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$; $(\mathsf{pk_S}, \mathsf{sk_S}) \leftarrow \mathsf{KGen_S}(\mathsf{pp})$; $(\mathsf{pk_Z}, \mathsf{sk_Z}) \leftarrow \mathsf{KGen_Z}(\mathsf{pp})$; $b \leftarrow \{0, 1\}$
$a \leftarrow \mathcal{A}^{\mathsf{Sig}(\mathsf{sk_S},\cdot,\cdot,\cdot),\mathsf{San}(\cdot,\mathsf{sk_Z},\cdot,\cdot,\cdot),\mathsf{Prov}(\mathsf{sk_S},\cdot,\cdot,\cdot),\mathsf{LoRSanit}(\mathsf{sk_S},\mathsf{sk_Z},\cdot,\cdot,b)}(\mathsf{pp}, \mathsf{pk_S}, \mathsf{pk_Z})$
if $a = b$ *then output* 1, *else output* 0.

E Security Proofs for Rerandomizable Tagging Scheme

User-Accountability. We assume the existence of EUF-CMA secure signature scheme Σ and pseudorandom function F by the assumption that one-way function exists. Suppose there exists PPT adversary $\mathcal{A}$ which breaks the user-accountability of $\mathcal{RT}$. Consider a PPT simulator $\mathcal{S}$ which acts as the adversary in the EUF-CMA game of Σ. $\mathcal{S}$ simulates the user-accountability game for $\mathcal{A}$ as follows:

- $\mathcal{S}$ receives pk^* and gains access to the signing oracle $\mathsf{SSig}(\mathsf{sk}^*, \cdot)$. It sets $\mathsf{pk}_\Sigma := \mathsf{pk}^*$ and simulates the pseudorandom function with a table.
- To simulate the Tag oracle on query $(\mathsf{pk}_{\mathtt{U},i}, m_i)$ where $\mathsf{pk}_{\mathtt{U},i} = (\mathsf{pk_C}, \mathsf{pk_{TD}}, \mathsf{pk}_e)$, it samples random tuple $(q_{i,1}, q_{i,2}, q_{i,3}, r_{i,1}, r_{i,2}, r_{i,3})$ and record it in the table. It computes $\rho_{i,1} \leftarrow g(r_{i,1})$, $\rho_{i,2} \leftarrow g(r_{i,2})$, $\mu_i \leftarrow \mathsf{CEval}(\mathsf{pk_C}, m_i; \rho_{i,1})$, and $y_i \leftarrow \mathsf{TDEval}(\mathsf{pk_{TD}}, \mu_i, \rho_{i,2})$. It encrypts $c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_e, m_i; r_{i,3})$. It then uses the signing oracle to obtain a signature σ_i on $(\mathsf{pk}_{\mathtt{U},i}, y_i, q_{i,1}, q_{i,2}, q_{i,3}, c_i)$. It outputs a tag $\tau_i = (\rho_{i,1}, \rho_{i,2}, q_{i,1}, q_{i,2}, q_{i,3}, c_i, \sigma_i)$.
- To simulate the TProv oracle on query $(\mathsf{pk}_{\mathtt{U},i}, m_i, \tau_i)$ where $\tau_i = (\rho_{i,1}, \rho_{i,2}, q_{i,1}, q_{i,2}, q_{i,3}, c_i, \sigma_i)$, it checks whether $(q_{i,1}, q_{i,2}, q_{i,3}, r_{i,1}, r_{i,2}, r_{i,3})$ exists on the table for some $(r_{i,1}, r_{i,2}, r_{i,3})$. If so, it outputs $(r_{i,1}, r_{i,2}, r_{i,3})$. Otherwise, it samples random $(r_{i,1}, r_{i,2}, r_{i,3})$, records $(q_{i,1}, q_{i,2}, q_{i,3}, r_{i,1}, r_{i,2}, r_{i,3})$ in the table, and returns $(r_{i,1}, r_{i,2}, r_{i,3})$.
- Eventually, $\mathcal{A}$ returns $(\mathsf{pk}_{\mathtt{U}}^*, m^*, \tau^*)$ where $\tau^* = (\rho_1^*, \rho_2^*, q_1^*, q_2^*, q_3^*, c^*, \sigma^*)$.
- $\mathcal{S}$ outputs $((\mathsf{pk}_{\mathtt{U}}^*, y^*, q_1^*, q_2^*, q_3^*, c^*), \sigma^*)$ where $y^* \leftarrow \mathsf{TDEval}(\mathsf{pk_{TD}}, \mu^*, \rho_2^*)$ and $\mu^* \leftarrow \mathsf{CEval}(\mathsf{pk_C}, m^*; \rho_1^*)$.

Let $(r_1^*, r_2^*, r_3^*) = \pi^* \leftarrow \mathsf{TProv}(\mathsf{sk}_{\mathtt{I}}, \mathsf{pk}_{\mathtt{U}}^*, \tau^*)$. With non-negligible probability, we have $(\mathsf{pk}_{\mathtt{U}}^*, m^*) \neq (\mathsf{pk}_{\mathtt{U},i}, m_i)$ for all i, $\mathsf{TVer}(\mathsf{pk}_{\mathtt{I}}, \mathsf{pk}_{\mathtt{U}}^*, \tau^*) = 1$, and $\mathsf{TJud}(\mathsf{pk}_{\mathtt{I}}, \mathsf{pk}_{\mathtt{U}}^*, \tau^*, \pi) \neq \mathtt{U}$.

We argue that $((\mathsf{pk}_{\mathtt{U}}^*, y^*, q_1^*, q_2^*, q_3^*, c^*), \sigma^*)$ is a valid forgery to Σ. This happens when either any of the component in $(\mathsf{pk}_{\mathtt{U}}^*, y^*, q_1^*, q_2^*, q_3^*, c^*)$ was not queried to SSig before.

Suppose $\mathsf{pk}_{\mathtt{U}}^* = \mathsf{pk}_{\mathtt{U},i}$ for some i, then by the first winning condition, we have $m^* \neq m_i$. The judgment suggests that $y^* \neq y_i$ or $(\mu^*, \rho_2^*) = (\mu_i, \rho_{i,2})$. Suppose the first case happens, we observe that y^* was never queried to the signing oracle before. In the second case, we can assume $q_1^* = q_{i,1}$ and thus $\rho_1^* = \rho_{i,1}$, for otherwise q_1^* was never queried to SSig. However, this violates that $m^* \neq m_i$ as $\mu^* = \mathsf{CEval}(\mathsf{pk_C}, m^*; \rho_1^*) = \mathsf{CEval}(\mathsf{pk_C}, m_i; \rho_1^*) = \mu_i$ (since $\rho_1^* = \rho_{i,1}$) implies $m^* = m_i$. We thus assume $\mathsf{pk}_{\mathtt{U}}^* \neq \mathsf{pk}_{\mathtt{U},i}$ for all i, which means that $\mathsf{pk}_{\mathtt{U}}^*$ was never queried to the signing oracle before.

Issuer-Accountability. Suppose there exists PPT adversary $\mathcal{A}$ which breaks the issuer-accountability of $\mathcal{RT}$. Consider a PPT simulator $\mathcal{S}$ which acts as the adversary in the collision-resistance game of TD. $\mathcal{S}$ simulates the issuer-accountability game for $\mathcal{A}$ as follows:

- $\mathcal{S}$ samples a random bit $b \leftarrow \{0,1\}$ and runs $\mathsf{TCGen}(1^\lambda, b)$ to generate pk_C and one of the trapdoors $\mathsf{sk}_{\mathsf{C},b}$ for the chameleon hash.
- Suppose $\mathcal{A}$ queries the ReTag oracle at most Q times. $\mathcal{S}$ samples random messages $m_i^\diamond \leftarrow \{0,1\}^\lambda$ and random $\rho_{i,1}^\diamond \leftarrow \{0,1\}^{2\lambda}$ and computes $\mu_i' \leftarrow \mathsf{CEval}(\mathsf{pk}_\mathsf{C}, m_i^\diamond; \rho_{i,1}^\diamond)$ for $i = 1, \ldots, Q$. All μ_i' are distinct with overwhelming probability.
- $\mathcal{S}$ sends $(\mu_1', \ldots, \mu_Q')$ to the challenger of the collision-resistance game of TD, and receives from the latter a public key pk_TD.
- To simulate the ReTag oracle on query $(\mathsf{pk}_{\mathtt{I},i}, m_i, m_i', \tau_i)$, where $\mathsf{pk}_{\mathtt{I},i} = \mathsf{pk}_{i,\Sigma}$ and $\tau_i = (\rho_{i,1}, \rho_{i,2}, q_{i,1}, q_{i,2}, q_{i,3}, c_i, \sigma_i)$, $\mathcal{S}$ computes $\mu_i := \mathsf{CEval}(\mathsf{pk}_\mathsf{C}, m_i; \rho_{i,1})$ and $y_i := \mathsf{TDEval}(\mathsf{pk}_\mathsf{TD}, \mu_i, \rho_{i,2})$. It queries the inversion oracle of TD with image y_i and receives $\rho_{i,2}'$. By definition of the collision-resistance game, we have $y_i = \mathsf{TDEval}(\mathsf{pk}_\mathsf{TD}, \mu_i', \rho_{i,2}')$. On the other hand, it computes $\rho_{i,1}' \leftarrow \mathsf{CInv}(\mathsf{sk}_\mathsf{C}, i, \rho_{i,1}^\diamond, m_i')$. $\mathcal{S}$ thus outputs $\tau_i' = (\rho_{i,1}', \rho_{i,2}', q_{i,1}, q_{i,2}, q_{i,3}, c_i, \sigma_i)$.
- Eventually, $\mathcal{A}$ outputs $(\mathsf{pk}_\mathtt{I}^*, m^*, \tau^*, \pi^*)$ for some $\mathsf{pk}_\mathtt{I}^* = \mathsf{pk}_\Sigma^*$, $\tau^* = (\rho_1^*, \rho_2^*, q_1^*, q_2^*, q_3^*, c^*, \sigma^*)$ and $\pi^* = (\hat{r}_1, \hat{r}_2, \hat{r}_3)$ such that, for all i, it holds that $(\mathsf{pk}_\mathtt{I}^*, m^*) \neq (\mathsf{pk}_{\mathtt{I},i}, m_i)$, $\mathsf{TVer}(\mathsf{pk}_\mathtt{I}^*, \mathsf{pk}_\mathtt{U}, \tau^*) = 1$, and $\mathsf{TJud}(\mathsf{pk}_\mathtt{I}^*, \mathsf{pk}_\mathtt{U}, \tau^*, \pi^*) \neq \mathtt{I}$.
- $\mathcal{S}$ computes $\hat{\rho}_1 \leftarrow g(\hat{r}_1)$, $\hat{\rho}_2 \leftarrow g(\hat{r}_2)$, $\hat{m} \leftarrow \mathsf{Ext}(\mathsf{pk}_e, c^*, \hat{r}_3)$, $\mu^* \leftarrow \mathsf{CEval}(\mathsf{pk}_\mathsf{C}, m^*; \rho_1^*)$, $\hat{\mu} \leftarrow \mathsf{CEval}(\mathsf{pk}_\mathsf{C}, \hat{m}; \hat{\rho}_1)$. We argue that $\mathcal{S}$ wins the collision-resistance game of TD by outputting $((\mu^*, \rho_2^*), (\hat{\mu}, \hat{\rho}_2))$.

By the condition $\mathsf{TJud}(\mathsf{pk}_\mathtt{I}^*, \mathsf{pk}_\mathtt{U}, \tau^*, \pi^*) \neq \mathtt{I}$, we have $y^* = \hat{y}$, where $y^* \leftarrow \mathsf{TDEval}(\mathsf{pk}_\mathsf{TD}, \mu^*, \rho_2^*)$ and $\hat{y} \leftarrow \mathsf{TDEval}(\mathsf{pk}_\mathsf{TD}, \hat{\mu}, \hat{\rho}_2)$, but $(\mu^*, \rho_2^*) \neq (\hat{\mu}, \hat{\rho}_2)$. It remains to argue that $\mu^*, \hat{\mu} \notin (\mu_1', \ldots, \mu_Q')$ with overwhelming probability.

Suppose that $\mu^* = \mu_i'$ for some i, we have $\mu^* = \mathsf{CEval}(\mathsf{pk}_\mathsf{C}, m^*; \rho_1^*) = \mathsf{CEval}(\mathsf{pk}_\mathsf{C}, m_i^\diamond; \rho_{i,1}^\diamond) = \mu_i'$. Since $m_i^\diamond$ is uniformly random, $(m^*, \rho_1^*) \neq (m_i^\diamond, \rho_{i,1}^\diamond)$ with overwhelming probability, thus, using pk_C and the collision (m^*, ρ_1^*) and $(m_i^\diamond, \rho_{i,1}^\diamond)$, there is an efficient algorithm using which $\mathcal{S}$ can output the other trapdoor $\mathsf{sk}_{\mathsf{C},1\oplus b}$ for the chameleon hash with probability at least $\frac{1}{2}$.

Finally, we argue that the case $\hat{\mu} = \mu_i'$ for some i happens with negligible probability. By the definition of the chameleon hash function, the distribution of μ_i' is identical to the distribution of $\rho_{i,1}^\diamond$, which is the uniform distribution over $\{0,1\}^{2\lambda}$. On the other hand, possible value of $\hat{\mu}$ only comes from $\{0,1\}^\lambda$ since it is computed from the pseudorandomness $\hat{\rho}_2 := g(\hat{r}_2)$, thus, the probability that they are equal is at most $\frac{2^\lambda}{2^{2\lambda}} = 2^{-\lambda}$, which is negligible.

Proof-Restricted Transparency. We assume the existence of pseudorandom generator g_1, and pseudorandom function F by the assumption that one-way function exists. Suppose there exists PPT adversary $\mathcal{A}$ which breaks the proof-restricted transparency of $\mathcal{RT}$. Consider a PPT simulator $\mathcal{S}$ which acts as a distinguisher of the pseudorandom generator g and a sequence of hybrid experiments.

- Game 0: This is the original proof-restricted transparency game.
- Game 1: $\mathcal{S}$ generates all keys honestly except that it simulates F by a table: On query q to F, it checks whether (q, r) appears on the table for some r. If so, it outputs r. Otherwise, it samples $r \leftarrow \{0,1\}^\lambda$ and outputs r.
- Game 2: $\mathcal{S}$ replaces the pseudorandom generator g_1 by a random function.
- Game 3: $\mathcal{S}$ replaces the ciphertext c output by the $\mathsf{Tag}/\mathsf{ReTag}_b$ oracle in the case $b = 0$ by an encryption of m'.

We argue that all experiments are computationally indistinguishable. The indistinguishability between Game 0 and 1 follows from the security of the pseudorandom function F. The indistinguishability between Game 1 and 2 follows from the security of the pseudorandom generator g_1. The indistinguishability between Game 2 and 3 follows from the CPA-security of $\mathcal{E}$. Finally, if $\mathcal{A}$ can distinguish the cases of the $\mathsf{Tag}/\mathsf{ReTag}_b$ oracle, $\mathcal{S}$ can distinguish ρ_2 from ρ_2'. This happens with negligible probability by the definition of the tag-based trapdoor function.

F Security Proofs for Accountable Ring Signature

Unforgeability. If a PPT adversary $\mathcal{A}$ can break the unforgeability by outputting a forgery, we can simply truncate the ciphertext and its proof from the forgery, which gives a forgery of the underlying scheme of Bose *et al.* [BDR15]. Simply put, the $\mathcal{SPE}$ ciphertext and the corresponding proof do not tamper the unforgeability.

Anonymity. If a PPT adversary $\mathcal{A}$ can break the anonymity under full key exposure of $\mathcal{RS}$, a PPT simulator $\mathcal{S}$ can use $\mathcal{A}$ to break the CCA-security of $\mathcal{SPE}$ as follows. $\mathcal{S}$ generates crs of $\mathcal{GS}$ in the simulation setting. This way of generating the crs is indistinguishable from that in the real scheme, by the hiding property of $\mathcal{GS}$. It then generates the remaining public parameters honestly, and receives opk as the public key of $\mathcal{SPE}$ from an SPE challenger. This is possible as the opk in the real scheme only depends on the security parameter λ. $\mathcal{S}$ answers the queries to the opening oracle by first redirecting the decryption steps to the decryption oracle of $\mathcal{SPE}$, and then simulating the proofs itself. $\mathcal{S}$ answers the queries to the challenge oracle honestly except that the proofs are now simulated as in the proof of BDR [BDR15], and the ciphertexts are obtained by redirecting the public keys to the challenge oracle of $\mathcal{SPE}$ (which we assume without loss of generality can be called twice). The simulated proofs are indistinguishable to those in the real scheme, for otherwise we can construct an adversary to break the hiding property of $\mathcal{GS}$. Thus, if $\mathcal{A}$ can distinguish the challenge oracle of $\mathcal{RS}$, it reduces to $\mathcal{S}$ distinguishing the challenge oracle of $\mathcal{SPE}$.

Traceability. Suppose a PPT adversary $\mathcal{A}$ breaks the traceability of $\mathcal{RS}$. By the definition of $\mathcal{GS}$, there exists an extractor which extracts the public key (A, B) encrypted in the ciphertexts (e_a, e_b) output by the adversary. By the perfect correctness of $\mathcal{SPE}$, (e_a, e_b) must decrypt to (A, B) respectively. Finally, by the completeness of $\mathcal{GS}$, the generated proofs must verify to 1, which is a contradiction.

Tracing Soundness. Suppose a PPT adversary $\mathcal{A}$ can generate an $\mathcal{RS}$ signature which opens to two signers. By the definition of $\mathcal{GS}$, there exists an extractor which extracts the opener secret key osk from the proofs output by the adversary. By the soundness of $\mathcal{GS}$, both $((A_0, r_{a,0}), (B_0, r_{b,0}))$ and $((A_1, r_{a,1}), (B_1, r_{b,1}))$ are message-randomness tuples encrypted to e_a and e_b. However, by the perfect correctness of $\mathcal{SPE}$, this is impossible.

G Security Proofs for the First Construction

Immutability. Suppose there exists PPT adversary $\mathcal{A}$ which breaks the immutability of $\mathcal{SS}$. Consider a PPT simulator $\mathcal{S}$ acting as the adversary in the EUF-CMA game of $\varSigma$. $\mathcal{S}$ receives pk^* and gains access to the signing oracle $\mathsf{SSig}(\mathsf{sk}^*, \cdot)$. $\mathcal{S}$ sets $\mathsf{pk_f} := \mathsf{pk}^*$ and generates other keys honestly.

When $\mathcal{A}$ queries $\mathsf{Sig}(\mathsf{sk_S}, \cdot, \cdot, \cdot)$, $\mathcal{S}$ computes everything honestly except that it computes σ_f as $\sigma_f \leftarrow \mathsf{SSig}(\mathsf{sk}^*, m_f)$. When $\mathcal{A}$ queries $\mathsf{Prov}(\mathsf{sk_S}, \cdot, \cdot, \cdot)$, $\mathcal{S}$ computes everything honestly.

Eventually, $\mathcal{A}$ outputs $(\mathsf{pk}^*_\mathtt{Z}, m^*, \sigma^*)$. By construction, we have $\sigma^* = (\sigma^*_f, \tau^*, \alpha^*)$ and $m^*_f = (f_\alpha(m^*), \mathsf{pk}^*_\mathtt{Z}, \alpha^*)$, such that $\mathsf{SVer}(\mathsf{pk_f}, m^*_f, \sigma^*_f) = 1$, and $\mathsf{TVer}(\mathsf{pk}_\mathtt{I}, \mathsf{pk}^*_\mathtt{U}, m^*, \tau^*) = 1$. With non-negligible probability, we have $\mathsf{Ver}(\mathsf{pk_S}, \mathsf{pk}^*_\mathtt{Z}, m^*, \sigma^*) = 1$, and for all i, $\mathsf{pk}^*_\mathtt{Z} \neq \mathsf{pk}_{\mathtt{Z},i}$ or $m^* \notin \{\delta(m_i) \mid \delta \text{ with } \alpha_i(\delta) = 1\}$ for the value α_i in σ_i. Thus, $\mathcal{S}$ outputs (m^*_f, σ^*_f) and wins the EUF-CMA game of $\varSigma$ with the same advantage as $\mathcal{A}$.

Sanitizer-Accountability. Suppose there exists PPT adversary $\mathcal{A}$ which breaks the sanitizer-accountability of $\mathcal{SS}$. Consider a PPT simulator $\mathcal{S}$, which chooses at the beginning a random bit $b \leftarrow \{0, 1\}$. If $b = 0$, $\mathcal{S}$ acts as the adversary in the EUF-CMA game of $\varSigma$. $\mathcal{S}$ receives pk^* and gains access to the signing oracle $\mathsf{SSig}(\mathsf{sk}^*, \cdot)$. $\mathcal{S}$ sets $\mathsf{pk_f} := \mathsf{pk}^*$ and generates other keys honestly. Else, $\mathcal{S}$ acts as the adversary in the user-accountability game of $\mathcal{RT}$. $\mathcal{S}$ receives pk^* and gains access to the tagging oracle $\mathsf{Tag}(\cdot, \mathsf{sk}^*, \cdot)$ and proof oracle $\mathsf{TProv}(\mathsf{sk}^*, \cdot, \cdot, \cdot)$. $\mathcal{S}$ sets $\mathsf{pk}_\mathtt{I} := \mathsf{pk}^*$ and generates other keys honestly. In both cases, when $\mathcal{A}$ queries $\mathsf{Prov}(\mathsf{sk_S}, \cdot, \cdot, \cdot)$, $\mathcal{S}$ computes everything honestly.

If $b = 0$, when $\mathcal{A}$ queries $\mathsf{Sig}(\mathsf{sk_S}, \cdot, \cdot, \cdot)$, $\mathcal{S}$ computes everything honestly except that it computes σ_f as $\sigma_f \leftarrow \mathsf{SSig}(\mathsf{sk}^*, m_f)$. If $b = 1$, when $\mathcal{A}$ queries $\mathsf{Sig}(\mathsf{sk_S}, \cdot, \cdot, \cdot)$, $\mathcal{S}$ computes everything honestly except that it computes τ as $\tau \leftarrow \mathsf{Tag}(\mathsf{sk}^*, \mathsf{pk}_\mathtt{U}, m)$ and π_τ as $\pi_\tau \leftarrow \mathsf{TProv}(\mathsf{sk}^*, \mathsf{pk}_\mathtt{U}, m, \tau)$.

Eventually, $\mathcal{A}$ outputs $(\mathsf{pk}^*_\mathtt{Z}, m^*, \sigma^*)$. By construction, we have $\sigma^* = (\sigma^*_f, \tau^*, \alpha^*)$ and $m^*_f = (f_\alpha(m^*), \mathsf{pk}^*_\mathtt{Z}, \alpha^*)$, such that $\mathsf{SVer}(\mathsf{pk_f}, m^*_f, \sigma^*_f) = 1$, and $\mathsf{TVer}(\mathsf{pk}_\mathtt{I}, \mathsf{pk}^*_\mathtt{U}, m^*, \tau^*) = 1$.

Let $\pi \leftarrow \mathsf{Prov}(\mathsf{sk_S}, m^*, \sigma^*, \mathsf{pk}^*_\mathtt{Z})$. For all i, with non-negligible probability, we have $(\mathsf{pk}^*_\mathtt{Z}, m^*) \neq (\mathsf{pk}_{\mathtt{Z},i}, m_i)$, $\mathsf{Ver}(\mathsf{pk_S}, \mathsf{pk}^*_\mathtt{Z}, m^*, \sigma^*) = 1$, and $\mathsf{Jud}(\mathsf{pk_S}, \mathsf{pk}^*_\mathtt{Z}, m^*, \sigma^*, \pi) \neq \mathtt{Z}$. By the judgment, we have $\mathsf{TJud}(\mathsf{pk}_\mathtt{I}, \mathsf{pk}^*_\mathtt{U}, m^*, \tau^*, \pi_\tau) = \mathtt{I}$.

Suppose (m^*_f, σ^*_f) is not a message-signature pair queried to and returned from the signing oracle before. If $\mathcal{S}$ guessed $b = 0$, it outputs (m^*_f, σ^*_f) and wins the EUF-CMA game of $\varSigma$ with the same advantage as $\mathcal{A}$. Otherwise, it has guessed wrongly and aborts.

On the other hand, suppose (m_f^*, σ_f^*) is a message-signature pair queried to and returned from the signing oracle before. It implies that $m_f^* = (f_\alpha(m^*), \mathsf{pk}_\mathsf{Z}^*, \alpha^*) = (f_\alpha(m_i), \mathsf{pk}_{\mathsf{Z},i}, \alpha_i) = m_{f,i}$. Thus $(\mathsf{pk}_\mathtt{U}^*, m^*) \neq (\mathsf{pk}_{\mathtt{U},i}, m_i)$ (since $m^* \neq m_i$). If in addition, $\mathcal{S}$ guessed $b = 1$, then $\mathcal{S}$ outputs $(\mathsf{pk}_\mathtt{U}^*, m^*, \tau^*)$ and wins the user-accountability game of $\mathcal{RT}$. Otherwise, it has guessed wrongly and aborts.

Since the cases $b = 0$ and $b = 1$ are indistinguishable from the view of $\mathcal{A}$, the chance of $\mathcal{S}$ guessing correctly is at least $\frac{1}{2}$.

Signer-Accountability. Suppose there exists $\mathcal{A}$ which breaks the signer-accountability of $\mathcal{SS}$. Consider a PPT simulator $\mathcal{S}$ acting as the adversary in the issuer-accountability game of $\mathcal{RT}$. $\mathcal{S}$ receives $\mathsf{pk}_\mathtt{U}^*$ and can access the oracle $\mathsf{ReTag}(\cdot, \mathsf{sk}_\mathtt{U}^*, \cdot, \cdot)$. $\mathcal{S}$ sets $\mathsf{pk}_\mathtt{U} := \mathsf{pk}_\mathtt{U}^*$ and generates other keys honestly. When $\mathcal{A}$ queries the oracle $\mathsf{San}(\cdot, \mathsf{sk}_\mathsf{Z}, \cdot, \cdot, \cdot)$, $\mathcal{S}$ computes everything honestly except that it computes τ' as $\tau' \leftarrow \mathsf{ReTag}(\mathsf{pk}_\mathtt{I}, \mathsf{sk}_\mathtt{U}^*, m, \tau)$. Eventually, $\mathcal{A}$ outputs $(\mathsf{pk}_\mathsf{S}^*, m^*, \sigma^*, \pi^*)$. By construction, we have $\sigma^* = (\sigma_f^*, \tau^*, \alpha^*)$ and $m_f^* = (f_\alpha(m^*), \mathsf{pk}_\mathsf{Z}^*, \alpha^*)$, such that $\mathsf{SVer}(\mathsf{pk}_\mathsf{f}, m_f^*, \sigma_f^*) = 1$, and $\mathsf{TVer}(\mathsf{pk}_\mathtt{I}^*, \mathsf{pk}_\mathtt{U}^*, m^*, \tau^*) = 1$. For all i, with non-negligible probability, we have $(\mathsf{pk}_\mathsf{S}^*, m^*) \neq (\mathsf{pk}_{\mathsf{S},i}, m_i')$, $\mathsf{Ver}(\mathsf{pk}_\mathsf{S}^*, \mathsf{pk}_\mathsf{Z}, m^*, \sigma^*) = 1$, and $\mathsf{Jud}(\mathsf{pk}_\mathsf{S}^*, \mathsf{pk}_\mathsf{Z}, m^*, \sigma^*, \pi^*) \neq \mathtt{S}$. By the judgment, we have $\mathsf{TJud}(m^*, \mathsf{pk}_\mathtt{I}, \mathsf{pk}_\mathtt{U}^*, \tau^*, \pi_\tau^*) = \mathtt{U}$. $\mathcal{S}$ outputs $(\mathsf{pk}_\mathtt{I}^*, m^*, \tau^*, \pi_\tau^*)$ to win the issuer-accountability game of $\mathcal{RT}$ with the same advantage as $\mathcal{A}$.

Proof-Restricted Transparency. Suppose there exists PPT adversary $\mathcal{A}$ which breaks the proof-restricted transparency of $\mathcal{SS}$. Consider a PPT simulator $\mathcal{S}$ acting as the adversary in the proof-restricted transparency game of $\mathcal{RT}$. $\mathcal{S}$ receives $(\mathsf{pk}_\mathtt{I}^*, \mathsf{pk}_\mathtt{U}^*)$ and gains access to the tagging oracle $\mathsf{Tag}(\cdot, \mathsf{sk}_\mathtt{I}^*, \cdot)$, the re-tagging oracle $\mathsf{ReTag}(\cdot, \mathsf{sk}_\mathtt{U}^*, \cdot, \cdot)$ the proof oracle $\mathsf{TProv}(\mathsf{sk}_\mathtt{I}^*, \cdot, \cdot, \cdot)$, and the tag-or-re-tag oracle $\mathsf{Tag/ReTag}_b(\cdot, \cdot)$. $\mathcal{S}$ sets $\mathsf{pk}_\mathtt{I} := \mathsf{pk}_\mathtt{I}^*$ and $\mathsf{pk}_\mathtt{U} := \mathsf{pk}_\mathtt{U}^*$ and generates other keys honestly.

When $\mathcal{A}$ queries the oracle $\mathsf{Sig}(\mathsf{sk}_\mathsf{S}, \cdot, \cdot, \cdot)$, $\mathcal{S}$ computes everything honestly except that it computes τ as $\tau \leftarrow \mathsf{Tag}(m, \mathsf{sk}_\mathtt{I}^*, \mathsf{pk}_\mathtt{U})$ and π_τ as $\pi_\tau \leftarrow \mathsf{TProv}(\mathsf{sk}_\mathtt{I}^*, \mathsf{pk}_\mathtt{U}, m, \tau)$. When $\mathcal{A}$ queries the oracle $\mathsf{San}(\cdot, \mathsf{sk}_\mathsf{Z}, \cdot, \cdot, \cdot)$, $\mathcal{S}$ computes everything honestly except that it computes τ' as $\tau' \leftarrow \mathsf{ReTag}(\mathsf{pk}_\mathtt{I}, \mathsf{sk}_\mathtt{U}^*, m, \tau)$. When $\mathcal{A}$ queries the oracle $\mathsf{Prov}(\mathsf{sk}_\mathsf{S}, \cdot, \cdot, \cdot)$, $\mathcal{S}$ computes everything honestly. When $\mathcal{A}$ queries the oracle $\mathsf{Sig/San}(\cdot, \cdot, \cdot)$, $\mathcal{S}$ obtains the tag τ' from $\tau' \leftarrow \mathsf{Tag/ReTag}_b(m, m')$.

Eventually, $\mathcal{A}$ outputs a bit a which is also output by $\mathcal{S}$. Note that since every part except the tag τ' output by the $\mathsf{Tag/ReTag}_b$ oracle has exactly the same distribution, the advantage of $\mathcal{A}$ in distinguishing the sanitized signatures is identical to that of $\mathcal{S}$ in distinguishing the tags from rerandomized tags.

H Security Proofs for the Second Construction

Immutability. Suppose there exists a PPT adversary $\mathcal{A}$ which breaks the immutability of $\mathcal{SS}_2$. Consider a PPT simulator $\mathcal{S}$ acting as the adversary in the sEUF-CMA game of $\mathit{\Sigma}$. $\mathcal{S}$ receives pk^* and gains access to the oracle $\mathsf{SSig}(\mathsf{sk}^*, \cdot)$ of $\mathit{\Sigma}$. $\mathcal{S}$ sets $\mathsf{pk}_\mathsf{f} := \mathsf{pk}^*$ and generates other keys honestly.

In the query phase, $\mathcal{S}$ answers queries to $\mathsf{Prov}(\mathsf{sk}_\mathsf{S}, \cdot, \cdot, \cdot)$ honestly. When $\mathcal{A}$ queries $\mathsf{Sig}(\mathsf{sk}_\mathsf{S}, \cdot, \cdot, \cdot)$, $\mathcal{S}$ computes everything honestly except that it receives σ_f by querying m_f to the oracle $\mathsf{SSig}(\mathsf{sk}^*, \cdot)$.

Eventually, $\mathcal{A}$ outputs a sanitized signature $(\mathsf{pk}_\mathsf{Z}^*, m^*, \sigma^*)$ such that $\mathsf{Ver}(\mathsf{pk}_\mathsf{S}, \mathsf{pk}_\mathsf{Z}^*, m^*, \sigma^*) = 1$ with non-negligible probability. By construction, we have $\sigma^* = (\sigma_f^*, \hat{\sigma}^*, \alpha^*)$ and $m_f^* = (f_\alpha(m^*), \alpha^*, \mathcal{R}^*)$, such that $\mathsf{RVer}(\mathsf{opk}_{\mathcal{RS}}, m^*, \mathcal{R}^*, \hat{\sigma}^*) = 1$ and $\mathsf{SVer}(\mathsf{pk}_\mathsf{f}, m_f^*, \sigma_f^*) = 1$, and for all i, $\mathsf{pk}_\mathsf{Z}^* \neq \mathsf{pk}_{\mathsf{Z},i}$ or $m^* \notin \{\delta(m_i) \mid \delta \text{ with } \alpha_i(\delta) = 1\}$ for the value α_i in σ_i. Thus, $\mathcal{S}$ outputs (m_f^*, σ_f^*) and wins the sEUF-CMA game of $\mathit{\Sigma}$.

Sanitizer-Accountability. Suppose there exists a PPT adversary $\mathcal{A}$ which breaks the sanitizer-accountability of $\mathcal{SS}_2$. Consider a PPT simulator $\mathcal{S}$ who acts as the adversary in the user traceability game or the unforgeability game of $\mathcal{RS}$.

$\mathcal{S}$ receives pp and flips a fair coin $c \in \{0, 1\}$ to decide its behavior.

- If $c = 0$, $\mathcal{S}$ guesses that $\mathcal{A}$ will output $(\mathsf{pk}_\mathsf{Z}^*, m^*, \sigma^*)$ where $\mathsf{pk}_\mathsf{Z}^* \neq \mathsf{pk}_\mathsf{Z}'$. In this case, $\mathcal{S}$ obtains $\mathsf{pk}_{\mathcal{RS}}^*$ from the challenger in the unforgeability game of $\mathcal{RS}$, runs $(\mathsf{opk}_{\mathcal{RS}}^*, \mathsf{osk}_{\mathcal{RS}}^*) \leftarrow \mathsf{ROKGen}(\mathsf{pp})$, runs $(\mathsf{pk}_\mathsf{f}, \mathsf{sk}_\mathsf{f}) \leftarrow \mathsf{SGen}(\mathsf{pp})$, and sets $\mathsf{pk}_\mathsf{S}^* = (\mathsf{pk}_\mathsf{f}{}^*, \mathsf{opk}_{\mathcal{RS}}^*, \mathsf{pk}_\mathsf{Z}^*)$. Remark that the probability for $\mathcal{A}$ to generate a ring signature key pair $(\mathsf{pk}_{\mathcal{RS}}', \mathsf{sk}_{\mathcal{RS}}')$ where $\mathsf{pk}_{\mathcal{RS}}^* = \mathsf{pk}_{\mathcal{RS}}'$ is negligible.
  When $\mathcal{A}$ makes a signing query to $\mathcal{S}$ on message m, $\mathcal{S}$ runs $\sigma_f \leftarrow \mathsf{SSig}(\mathsf{sk}_\mathsf{f}{}^*, m_f, \alpha)$, makes a signing query m to the signing oracle of $\mathcal{RS}$ to obtain $\hat{\sigma}$, and returns $\sigma = (\sigma_f, \hat{\sigma}, \alpha)$.
  When $\mathcal{A}$ makes a proof query to $\mathcal{S}$ with a signature generated with respect to pk_S^*, $\mathcal{S}$ runs $\pi \leftarrow \mathsf{ROpen}(\mathsf{osk}_{\mathcal{RS}}^*, m, \mathcal{R}, \hat{\sigma})$ to answer the query.

Eventually, $\mathcal{A}$ outputs $(\mathsf{pk}_\mathsf{Z}^*, m^*, \sigma^*)$. By construction, we have $\sigma^* = (\sigma_f^*, \hat{\sigma}^*, \alpha^*)$ and $m_f^* = (f_\alpha(m^*), \alpha^*, \mathcal{R}^*)$, such that $\mathsf{RVer}(\mathsf{opk}_{\mathcal{RS}}, m^*, \mathcal{R}^*, \hat{\sigma}^*) = 1$ and $\mathsf{SVer}(\mathsf{pk}_\mathsf{f}, m_f^*, \sigma_f^*) = 1$.

Let $\pi = (\mathsf{pk}'_{\mathcal{RS}}, \psi) \leftarrow \mathsf{ROpen}(\mathsf{osk}_{\mathcal{RS}}^*, m^*, \mathcal{R}, \hat{\sigma}^*)$. For all i, with non-negligible probability, we have $(\mathsf{pk}_\mathsf{Z}^*, m^*) \neq (\mathsf{pk}_{\mathsf{Z},i}, m_i)$, $\mathsf{Ver}(\mathsf{pk}_\mathsf{S}^*, \mathsf{pk}_\mathsf{Z}^*, m^*, \sigma^*) = 1$, and $\mathsf{Jud}(\mathsf{pk}_\mathsf{S}^*, \mathsf{pk}_\mathsf{Z}^*, m^*, \sigma^*, \pi) \neq \mathsf{Z}$. By the judgment, we have $\mathsf{pk}'_{\mathcal{RS}} = \mathsf{pk}_{\mathcal{RS}}^* \neq \mathsf{pk}_\mathsf{Z}^*$ and $\mathsf{RJud}(\mathsf{opk}_{\mathcal{RS}}, m, \mathcal{R}, \hat{\sigma}, \mathsf{pk}_{\mathcal{RS}}^*, \psi) = 0$. Hence, $\mathcal{S}$ outputs $(m^*, \hat{\sigma}^*)$ as a forgery for $\mathcal{RS}$.

- If $c = 1$, $\mathcal{S}$ guesses that $\mathcal{A}$ will output $(\mathsf{pk}_\mathsf{Z}^*, m^*, \sigma^*)$ where $\mathsf{pk}_\mathsf{Z}^* = \mathsf{pk}'_\mathsf{Z}$. In this case, $\mathcal{S}$ runs $(\mathsf{pk}_{\mathcal{RS}}^*, \mathsf{sk}_{\mathcal{RS}}^*) \leftarrow \mathsf{RUKGen}(\mathsf{pp})$, runs $(\mathsf{opk}_{\mathcal{RS}}^*, \mathsf{osk}_{\mathcal{RS}}^*) \leftarrow \mathsf{ROKGen}(\mathsf{pp})$, runs $(\mathsf{pk}_\mathsf{f}, \mathsf{sk}_\mathsf{f}) \leftarrow \mathsf{SGen}(\mathsf{pp})$, and sets $\mathsf{pk}_\mathsf{S}^* = (\mathsf{pk}_\mathsf{f}{}^*, \mathsf{opk}_{\mathcal{RS}}^*, \mathsf{pk}_\mathsf{Z}^*)$.

  When $\mathcal{A}$ makes a signing query to $\mathcal{S}$ on (m, α), $\mathcal{S}$ runs $\sigma \leftarrow \mathsf{Sig}(\mathsf{sk}_\mathsf{S}^*, \mathsf{pk}_\mathsf{Z}, m, \alpha)$ and returns σ.

  When $\mathcal{A}$ makes a proof query to $\mathcal{S}$ with a signature generated with respect to pk_S^*, $\mathcal{S}$ runs $\pi \leftarrow \mathsf{ROpen}(\mathsf{osk}_{\mathcal{RS}}^*, m, \mathcal{R}, \hat{\sigma})$ to answer the query.

  Eventually, $\mathcal{A}$ outputs $(\mathsf{pk}_\mathsf{Z}^*, m^*, \sigma^*)$. By construction, we have $\sigma^* = (\sigma_f^*, \hat{\sigma}^*, \alpha^*)$ and $m_f^* = (f_\alpha(m^*), \alpha^*, \mathcal{R}^*)$, such that $\mathsf{RVer}(\mathsf{opk}_{\mathcal{RS}}, m^*, \mathcal{R}^*, \hat{\sigma}^*) = 1$ and $\mathsf{SVer}(\mathsf{pk}_\mathsf{f}, m_f^*, \sigma_f^*) = 1$.

  Let $\pi = (\mathsf{pk}'_{\mathcal{RS}}, \psi) \leftarrow \mathsf{ROpen}(\mathsf{osk}_{\mathcal{RS}}^*, m^*, \mathcal{R}, \hat{\sigma}^*)$. For all i, with non-negligible probability, we have $(\mathsf{pk}_\mathsf{Z}^*, m^*) \neq (\mathsf{pk}_{\mathsf{Z},i}, m_i)$, $\mathsf{Ver}(\mathsf{pk}_\mathsf{S}, \mathsf{pk}_\mathsf{Z}^*, m^*, \sigma^*) = 1$, and $\mathsf{Jud}(\mathsf{pk}_\mathsf{S}, \mathsf{pk}_\mathsf{Z}^*, m^*, \sigma^*, \pi) \neq \mathsf{Z}$. By the judgment, we have $\mathsf{pk}_{\mathcal{RS}}^* = \mathsf{pk}'_{\mathcal{RS}}$ and $\mathsf{RJud}(\mathsf{opk}_{\mathcal{RS}}, m, \mathcal{R}, \hat{\sigma}, \mathsf{pk}_{\mathcal{RS}}^*, \psi) = 0$. This is a break to the traceability of $\mathcal{RS}$.

Signer-Accountability. Suppose there exists a PPT adversary $\mathcal{A}$ which breaks the signer-accountability of $\mathcal{SS}_2$. Consider a PPT simulator $\mathcal{S}$ who acts as the adversary in the unforgeability game of $\mathcal{RS}$. $\mathcal{S}$ receives pp and gets access to a signing oracle of $\mathcal{RS}$. $\mathcal{S}$ obtains $(\mathsf{pk}_{\mathcal{RS}}^*, \mathsf{opk}_{\mathcal{RS}})$, and sets $\mathsf{pk}_\mathsf{Z}^* = \mathsf{pk}_{\mathcal{RS}}^*$. $\mathcal{A}$ receives pp and can do anything with pp including generating signers and signatures by itself.

When $\mathcal{A}$ makes a sanitizing query to $\mathcal{S}$, $\mathcal{S}$ extracts σ_f from the query, forwards the sanitized message to the signing oracle of $\mathcal{RS}$ to obtain $\hat{\sigma}$, and returns $(\sigma_f, \hat{\sigma}, \alpha)$ to $\mathcal{A}$.

Eventually, $\mathcal{A}$ outputs $(\mathsf{pk}_\mathsf{Z}^*, m^*, \sigma^*, \pi^*)$. By construction, we have $\sigma^* = (\sigma_f^*, \hat{\sigma}^*, \alpha^*)$ and $m_f^* = (f_\alpha(m^*), \alpha^*, \mathcal{R}^*)$, such that $\mathsf{RVer}(\mathsf{opk}_{\mathcal{RS}}, m^*, \mathcal{R}^*, \hat{\sigma}^*) = 1$ and $\mathsf{SVer}(\mathsf{pk}_\mathsf{f}, m_f^*, \sigma_f^*) = 1$.

Let $\pi = (\mathsf{pk}'_{\mathcal{RS}}, \psi) \leftarrow \mathsf{ROpen}(\mathsf{osk}_{\mathcal{RS}}^*, m^*, \mathcal{R}, \hat{\sigma}^*)$. For all i, with non-negligible probability, we have $(\mathsf{pk}_\mathsf{Z}^*, m^*) \neq (\mathsf{pk}_{\mathsf{Z},i}, m_i)$, $\mathsf{Ver}(\mathsf{pk}_\mathsf{S}, \mathsf{pk}_\mathsf{Z}^*, m^*, \sigma^*) = 1$, and $\mathsf{Jud}(\mathsf{pk}_\mathsf{S}, \mathsf{pk}_\mathsf{Z}^*, m^*, \sigma^*, \pi) = \mathsf{Z}$. By the judgment, we have $\mathsf{pk}'_{\mathcal{RS}} = \mathsf{pk}_{\mathcal{RS}}^*$ and $\mathsf{RJud}(\mathsf{opk}_{\mathcal{RS}}, m, \mathcal{R}, \hat{\sigma}, \mathsf{pk}_{\mathcal{RS}}^*, \psi) = 1$. Hence, (m^*, σ^*) is a successful forgery for $\mathcal{RS}$ in the first case in Definition 16.

Proof-Restricted Transparency. Suppose there exists $\mathcal{A}$ which breaks the proof-restricted transparency of $\mathcal{SS}$. Consider a PPT simulator $\mathcal{S}$ acting as the adversary in the anonymity game of $\mathcal{RS}$. $\mathcal{S}$ receives $(\mathsf{pp}, \mathsf{opk})$, and gains access to the challenge oracle $\mathsf{Chal}_b(\cdot,\cdot,\cdot,\cdot)$ and opening oracle $\mathsf{ROpen}(\mathsf{osk},\cdot,\cdot,\cdot)$ where $b \in \{0,1\}$ is chosen by $\mathcal{S}$. $\mathcal{S}$ generates other keys honestly, and use them to simulate the Sig and San oracles. When $\mathcal{A}$ queries $\mathsf{Sig/San}(\cdot, \mathsf{sk}_\mathsf{Z}, \cdot,\cdot,\cdot)$, we consider a sequence of hybrids: In the first hybrid, $\mathcal{S}$ simulates $\mathsf{Sig/San}$ using the keys generated by itself. In the k-th intermediate hybrids, $\mathcal{S}$ replaces $\hat{\sigma}$ returned by the k-th query to the $\mathsf{Sig/San}$ oracle by the challenge signature of the anonymity game of $\mathcal{RS}$. When $\mathcal{A}$ queries $\mathsf{Prov}(\mathsf{sk}_\mathsf{S}, \cdot,\cdot,\cdot), \mathcal{S}$ computes π as $\mathsf{ROpen}(m, \mathcal{R}, \hat{\sigma}, \mathsf{osk})$.

Eventually, $\mathcal{A}$ outputs a bit a which is also output by $\mathcal{S}$. If $\mathcal{A}$ can distinguish the original proof-restricted transparency games for $b = 0, 1$, then it must also be able to distinguish the k-th hybrid from the $(k+1)$-th for some k. In such case, $\mathcal{S}$ breaks the anonymity of $\mathcal{RS}$.

Unlinkability. To argue the unlinkability of our scheme, consider the tuples $(m_0, \delta_0, \sigma_0)$ and $(m_1, \delta_1, \sigma_1)$ submitted by the adversary to $\mathsf{LoRSanit}(\mathsf{sk}_\mathsf{S}, \mathsf{sk}_\mathsf{Z}, \cdot, \cdot, b)$. By the definition of unlinkability, we have $\delta_0(m_0) = \delta_1(m_1)$. Thus, the distributions of the ring signatures are identical regardless of the cases $b = 0$ and $b = 1$. Furthermore, observe that $m_{f,0} = (f_\alpha(m_0), \alpha, \mathcal{R}) = (f_\alpha(\delta_0(m_0)), \alpha, \mathcal{R}) = (f_\alpha(\delta_0(m_0)), \alpha, \mathcal{R}) = (f_\alpha(m_0), \alpha, \mathcal{R}) = m_{f,0}$. Since Σ is deterministic, it should be the case that $\sigma_{f,0} = \sigma_{f,1}$. If $\mathcal{A}$ submits distinct $\sigma_{f,0}$ and $\sigma_{f,1}$, $\mathcal{S}$ can use $\mathcal{A}$ to break the sEUF-CMA-security of Σ. Finally, $\alpha_0 = \alpha_1$. Thus, the probability that $\mathcal{A}$ distinguishes the cases $b = 0$ and $b = 1$ is zero.