An extended abstract of this paper appears at PKC'19.

Efficient Invisible and Unlinkable Sanitizable Signatures

(Full Version)

Xavier Bultel[1], Pascal Lafourcade[2], Russell W. F. Lai[3], Giulio Malavolta[3], Dominique Schröder[3], and Sri Aravinda Krishnan Thyagarajan[3]

[1] Univ Rennes, CNRS, IRISA
[2] University Clermont Auvergne, LIMOS, France
[3] Friedrich-Alexander University Erlangen-Nürnberg, Erlangen, Germany

Abstract. Sanitizable signatures allow designated parties (the sanitizers) to apply arbitrary modifications to some restricted parts of signed messages. A secure scheme should not only be unforgeable, but also protect privacy and hold both the signer and the sanitizer accountable. Two important security properties that are seemingly difficult to achieve simultaneously and efficiently are invisibility and unlinkability. While invisibility ensures that the admissible modifications are hidden from external parties, unlinkability says that sanitized signatures cannot be linked to their sources. Achieving both properties simultaneously is crucial for applications where sensitive personal data is signed with respect to data-dependent admissible modifications. The existence of an efficient construction achieving both properties was recently posed as an open question by Camenisch et al. (PKC'17).
In this work, we propose a solution to this problem with a two-step construction. First, we construct (non-accountable) invisible and unlinkable sanitizable signatures from signatures on equivalence classes and other basic primitives. Second, we put forth a generic transformation using verifiable ring signatures to turn any non-accountable sanitizable signature into an accountable one while preserving all other properties. When instantiating in the generic group and random oracle model, the efficiency of our construction is comparable to that of prior constructions, while providing stronger security guarantees.

1 Introduction

Sanitizable signature schemes introduced by Ateniese *et al.* [ACdT05] are signature schemes that allow a certain degree of controlled malleability: The signer signs messages along with some "admissible modifications" with respect to another party called the sanitizer. The sanitizer can (only) convert a given message-signature pair into one that is admissible. When necessary, the signer can (dis)prove the authorship of a given signature to the public which is modelled by a party called the judge. Over the years, the originally informal security properties [ACdT05] were formalized [BFF+09, BFLS10] and strengthened [KSS16]. Beyond unforgeability, sanitizable signatures provide one with meaningful privacy guarantees, which are important when signing and sanitizing sensitive data. Furthermore, accountability of the signatures prevents the parties from misbehaving as they may eventually get caught. New properties, such as invisibility, were recently proposed [CDK+17, BCD+17]. To summarize, we recall the security properties which we consider in this work:

Immutability: The sanitizer cannot modify non-admissible messages.
Accountability: The signer cannot accuse the sanitizer (vice versa) of signing.
Transparency: Non-sanitized and sanitized signatures are indistinguishable.
Invisibility: The class of admissible modifications are hidden from external parties.
Unlinkability: Sanitized signatures cannot be linked to their sources.

Applications. Ateniese *et al.* [ACdT05] suggested a wide range of applications of sanitizable signatures, including multicast transmission, database outsourcing, protecting health information, and secure routing. As an example, we highlight the importance of invisibility and unlinkability of sanitizable signatures for signing medical records. Suppose that a physician signs medical records of patients using a sanitizable signature scheme. The patients can then sanitize the medical record for different purposes. For example, they may 1) remove the personal information and delegate the

anonymized record for analysis; 2) remove everything except for the personal information for financing purposes, in such a way that the receivers are convinced of the authenticity of the record. As discussed in [BFLS10], *unlinkability* ensures that colluding receivers cannot reconstruct the full medical records since they cannot link records sanitized from the same source.

However, suppose that the admissible modifications chosen by the physician, are data-dependent. For instance, patients suffering from certain sensitive medical condition that might possibly lead them to be discriminated against, may be allowed to change the fields corresponding to these conditions to NO, while other patients not suffering from any of these conditions are not allowed to change any of the fields to YES. Such a policy of assigning admissible modifications prevents the former patients from facing discrimination when revealing such conditions is not necessary, while preventing the latter patients from getting hold of drugs which are otherwise only issued to patients suffering from those conditions. The security property *invisibility* is crucial for such a scenario, since the receiver of a sanitized medical record can otherwise easily tell whether the corresponding patient suffers from a sensitive condition by just checking whether changing the corresponding field in the record is modifiable or not.

1.1 Open Problem

As discussed above, achieving both unlinkability and invisibility is desirable for certain applications. Obviously, realizing both notions simultaneously is rather easy from a theoretical point of view using common "encrypt and prove" techniques. Although the feasibility is clear, doing so efficiently turns out to be challenging.

One obvious starting point to answer this question is the idea to lift an existing invisible sanitizable signature scheme to an unlinkable one. Following this path does not seem to be fruitful, because existing invisible constructions adopt the "chameleon-hash-then-sign" paradigm: The main ingredients in this approach are a signature scheme for which the signer has the secret key, and a chameleon hashing scheme[4] for which the sanitizer has the trapdoor. To sign a message, the signer first splits the message into ℓ message blocks, some of which are "admissible", meaning that they are allowed to be changed by the sanitizer, while some are not. The signer then computes the chameleon hashes of the individual message blocks, in such a way that the sanitizer can recover the trapdoors corresponding to the admissible blocks, and sign the hash values. Later, the sanitizer can change the admissible blocks by using the trapdoors to "explain" the hash values with new messages.

Under the "chameleon-hash-then-sign" paradigm, we can see that signatures are inherently linkable. This is because all signatures which are sanitized from a fresh signature contain the same set of hash values. One can of course hide the hash values by using generic non-interactive zero-knowledge arguments, but that would not yield a practical scheme. Therefore, [CDK+17] and [BCD+17] posed the following open problem:

> *"How to construct (efficient) sanitizable signature schemes which are simultaneously unlinkable and invisible?"*

In this work, we answer this question by constructing the first efficient invisible and unlinkable sanitizable signature scheme.

1.2 Our Techniques

To solve the problem of constructing an efficient unlinkable and invisible sanitizable signature scheme, we suggest a modular approach visualized with the help of Figure 1.

First, we decouple the problem by presenting a generic transformation that turns any "weak" sanitizable signature scheme, which is not accountable, into an accountable one while preserving or upgrading all other properties. Our transformation is very efficient as it only requires a verifiable [LX03] ring signature[5]. Recall that a ring signature scheme allows a signer to sign messages on behalf of an ad-hoc group picked during signature generation. Verifiability in this context means that a signer can (dis)prove the authorship of a given signature a posteriori. The basic idea of our transformation is as follows.

[4] A chameleon hashing scheme allows to generate a probabilistic hash function H together with a trapdoor. With the latter, one can efficiently compute a randomness r when given any message m and hash value h such that $h = H(m, r)$.

[5] We remark that verifiable ring signatures can be implemented from unique [FZ12], linkable [LWW04], accountable [XY04], or traceable [FS07] ring signatures, so the transformation also works with these kinds of signature.

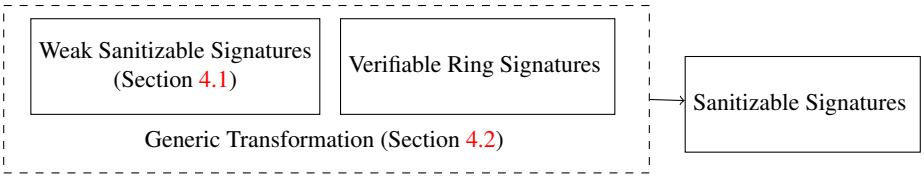

Fig. 1: Outline of our approach.

To sign (resp. sanitize), the signer (resp. the sanitizer) runs the sign (resp. sanitize) algorithm of the weak sanitizable signature scheme, and signs the whole output (ignoring the underlying structure) with a verifiable ring signature scheme, where the ring is composed by the verification keys of the signer and the sanitizer. The resulting scheme is accountable because the signer can (dis)prove the authorship of a certain signature using the accountability property of the ring signature itself. Our transformation does not only preserve the underlying properties of the (non-accountable) sanitizable signature scheme, but it also strengthens some of them: If the underling scheme is weakly immutable (resp. weakly unlinkable), then the resulting scheme is immutable (resp. unlinkable). Loosely speaking, weak immutability considers a forgery as legitimate if it corresponds to a pre-determined sanitizer, as opposed to *any* sanitizer. On the other hand, weak unlinkability restricts the queries of the adversary to the sanitization oracle to consist exclusively of honestly generated signatures.

Next, we tackle the main problem of constructing an invisible and unlinkable (non-accountable) sanitizable signature scheme. The (long-term) public and secret keys of the signer are the verification and signing keys of a certain signature scheme, which we refer to as the outer-layer scheme. To sign a message, the signer splits the message into ℓ message blocks, and generates ℓ pairs of signing and verification keys of an inner-layer signature scheme. Naturally, the signer signs each message block with the corresponding inner signing key, and signs the verification keys with its (long-term) outer secret key. To allow sanitization, the signer additionally delegates the inner signing keys corresponding to the admissible blocks by encrypting them under the sanitizer public key. Note that all message blocks are treated equally, which is critical for invisibility, except for the generation of the ciphertext. By the semantic security of the encryption scheme, the signature is still invisible to the eyes of an external observer.

To generate signatures for sanitized messages, the sanitizer simply uses the delegated inner signing keys to sign the modified message blocks. However, the resulting sanitized signature is now linkable since the outer signature on the inner verification keys and the keys themselves remain unchanged. To resolve this issue, we need to craft an inner signature scheme with some special properties. Our inner signatures scheme is very similar to the Boneh–Lynn–Shacham (BLS) signature scheme [BLS01] and works as follows: The public key consists of two group elements (G_1^x, G_1^{xy}) and the secret key $y \in \mathbb{Z}_q$ can be used to sign a message m by computing $\sigma := \mathsf{H}(m)^y \in G_2$. The verification is done using the pairing

$$e\left(G_1^x, \sigma\right) = e\left(G_1^{xy}, \mathsf{H}(m)\right).$$

The difference with respect to BLS lies in the extra term of the public key whose role will appear clear in a moment. The property that we need is that the keys and the signatures are publicly re-randomizable, i.e., one can compute consistent scalings of both the signature and the public key

$$(\sigma, (G_1^x, G_1^{xy})) \mapsto (\sigma, (G_1^x, G_1^{xy}))^r := \left(\sigma^r, \left(G_1^{rx}, G_1^{rx \cdot y}\right)\right).$$

It is easy to see that the resulting key-signature pair is still consistent, i.e., the verification checks out. Unfortunately, it turns out that the re-randomization strategy of above is too simplistic and it is prone to mix-and-match attacks. We therefore devise a slightly more sophisticated re-randomization procedure

$$(\sigma, (G_1^x, G_1^{xy})) \mapsto \left(\sigma^s, \left(G_1^{rx}, G_1^{rx \cdot sy}\right)\right).$$

which scales the two elements of the public keys by two different scalars r and rs respectively. Fortunately, this does not affect the correctness of the scheme.

The last obstacle towards decorrelating signed and sanitized signatures is a mechanism to publicly rerandomize the outer signature so that it is consistent with the rerandomized inner verification keys. More concretely, the problem is to rerandomize signatures of $(G_1^{x_1}, \ldots, G_1^{x_\ell})$ and $(G_1^{x_1 y_1}, \ldots, G_1^{x_\ell y_\ell})$ to signatures of $(G_1^{rx_1}, \ldots, G_1^{rx_\ell})$ and

Sig. SK	San. SK	Sig. PK	San. PK	Signature	Proof
$(\ell+1)\ \mathbb{Z}_q^*$	$2\ \mathbb{Z}_q^*$	$1\ \mathbb{G}_1+$ $\ell\ \mathbb{G}_2$	$2\ \mathbb{G}_1$	$(\ell+5)\ \mathbb{Z}_q^*+$ $(2\ell+11)\ \mathbb{G}_1+$ $(\ell+2)\ \mathbb{G}_2$	$2\ \mathbb{Z}_q^*+$ $3\ \mathbb{G}_1$

Table 1: Size of the parameters.

Op.	Signing	Sanitizing	Verifying	Proving	Judging
Exp.	$(4\ell+11)\ \mathbb{G}_1+$ $(\ell+2)\ \mathbb{G}_2$	$(2\ell+14)\ \mathbb{G}_1+$ $(\ell+2)\ \mathbb{G}_2$	$8\ \mathbb{G}_1$	$3\ \mathbb{G}_1$	$4\ \mathbb{G}_1$
Pairings	-	-	$4\ell+6$	-	-

Table 2: Dominating operations in algorithms.

$(G_1^{rsx_1y_1}, \ldots, G_1^{rsx_\ell y_\ell})$ respectively. It turns out that equivalence class signatures (EQS) [HS14] provide exactly such functionality.

1.3 Our Results

To summarize, in this paper we present the following results:

- We present the first *efficient* sanitizable signature scheme which simultaneously achieves unlinkability and invisibility. This resolves an open problem posed by Camenisch *et al.* [CDK+17]. Our construction is over type-III pairing groups. It uses an equivalence class signature (EQS) scheme, a public-key encryption (PKE) scheme, a hash function (modeled as a random oracle) with images living in $\mathbb{G}_2$, and a verifiable ring signature (VRS) scheme. We suggest to instantiate our construction with the EQS scheme of Fuchsbauer, Hanser, and Slamanig [FHS18], the PKE scheme obtained by applying the Fujisaki-Okamoto transformation [FO13] to the ElGamal encryption scheme [ElG84], and the VRS scheme of Bultel and Lafourcade [BL17]. The efficiency of such an instantiation is summarized in Table 1 and 2.
- We construct weak sanitizable signatures from equivalence class signatures and other basic primitives. The scheme is weak in the sense that it satisfies weak immutability, weak unlinkability, strong proof-restricted transparency[6], and strong invisibility, but not accountability.
- We present a generic transformation from weak sanitizable signatures to fully-fledged sanitizable signatures, using VRS. Fully-fledged sanitizable signatures satisfy immutability, unlinkability, strong proof-restricted transparency, strong invisibility, and strong accountability. The transformation is very efficient as it ignores the structure of the underlying weak sanitizable signature scheme. This allows the future design of sanitizable signatures to focus on achieving other properties while not worrying about accountability.

1.4 Related work

An alternative definition of accountability called *non-interactive public accountability* was given by Brzuska *et al.* [BPS13]. This variant of accountability is mutually exclusive with transparency. Several existing works [BFLS10, FKM+16, LZCS16, BL17] proposed schemes that are both transparent and unlinkable. Recently, Krenn *et al.* [KSS16] propose the "strong" versions of unforgeability, (non-interactive public) accountability, and transparency.

The above works do not consider the notion of invisibility which dates back to the original work by Ateniese *et al.* [ACdT05], and was formalized by Camenisch *et al.* [CDK+17]. Beck *et al.* [BCD+17] refined the notion to strong invisibility and proposed a scheme that is both strongly invisible and strongly accountable. Recently Fischlin *et al.* [FH18] show that an invisible (but not unlinkable nor transparent) sanitizable signature scheme can be obtained from any public key encryption scheme.

[6] Our construction actually achieves perfect strong (non-proof-restricted) transparency.

Miyazaki *et al.* [MHI08] also considered "invisible sanitizable signatures" which is actually a different primitive known as *redactable signatures* [DPSS16] as discussed in [CDK+17]. Extensions of sanitizable signatures such as the multi-sanitizer setting [CJL12] and a setting where the modification capabilities of the sanitizer are limited [CJ10] were also considered. Other primitives related to sanitizable signatures include homomorphic signatures [JMSW02, JWL12], redactable signatures [DPSS16, PS14, BBD+10], and proxy signatures [WP03, Shi06, OTO99]. To the best of our knowledge, none of the existing works present an efficient sanitizable signature scheme that simultaneously achieves all of the five security properties, and in particular unlinkability and invisibility together.

2 Preliminaries

Throughout this work we denote by $\lambda \in \mathbb{N}$ the security parameter and by $\mathsf{poly}(\lambda)$ any function that is bounded by a polynomial in λ. We denote any function that is negligible in the security parameter by $\mathsf{negl}(\lambda)$. We say that an algorithm is PPT if it is modelled as a probabilistic Turing machine whose running time is bounded by some function $\mathsf{poly}(\lambda)$. Given a set S, we denote by $x \leftarrow S$ the sampling of and element uniformly at random from S, and we denote by $x \leftarrow \mathcal{A}(\mathsf{in})$ the output of the algorithm $\mathcal{A}$ on input in. The elements of the set $\{1, \ldots, n\}$ are succinctly represented as $[n]$. Next we define the necessary notions for understanding our constructions.

2.1 Class-Hiding Groups

Let $\mathcal{BG} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, g_T, e, q) \leftarrow \mathsf{BGGen}(1^\lambda)$ be the description of a multiplicative bilinear group of prime order q generated by some efficient PPT algorithm $\mathsf{BGGen}(1^\lambda)$. Let $\bar{X} = (X_1, \ldots, X_\ell) \in \mathbb{G}_1^\ell$ and $\rho \in \mathbb{Z}_q$. We write $\bar{X} := (X_1, \ldots, X_\ell)^\rho := (X_1^\rho, \ldots, X_\ell^\rho)$. We then define the equivalence relation

$$\mathcal{R} := \{(\bar{M}, \bar{N}) : \exists \ell > 1,\ \rho \in \mathbb{Z}_q^* \ s.t.\ (\bar{M}, \bar{N}) \in \mathbb{G}_1^\ell \times \mathbb{G}_1^\ell \wedge \bar{N} = \bar{M}^\rho\}.$$

For a vector $\bar{M} \in \mathbb{G}_1^\ell$ for some $\ell > 1$, its equivalence class is defined by

$$[\bar{M}]_\mathcal{R} := \{\bar{N} \in \mathbb{G}_1^\ell : (\bar{M}, \bar{N}) \in \mathcal{R}\}.$$

Next we define the notion of class hiding for a relation $\mathcal{R}$, which intuitively says that it should be hard to distinguish elements from the same equivalence class from randomly sampled group elements[7].

Definition 1 (Class-Hiding). *A relation $\mathcal{R}$ is said to be class-hiding if for all $\ell > 1$ and for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\lambda)$ such that*

$$\left| \mathsf{Pr}\left[b' = b : \begin{array}{l} b \leftarrow \{0,1\}; \mathcal{BG} \leftarrow \mathsf{BGGen}(1^\lambda); (M, M_0) \leftarrow (\mathbb{G}_1^\ell)^2; \\ M_1 \leftarrow [M]_\mathcal{R}; b' \leftarrow \mathcal{A}(\mathcal{BG}, M, M_b) \end{array} \right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda)\,.$$

The following lemma was proven (in different wordings) by Hanser and Slamanig [HS14]:

Lemma 1 ([HS14]). *$\mathcal{R}$ is class-hiding if and only if the DDH assumption holds in $\mathbb{G}_1$.*

2.2 Equivalence Class Signatures

Equivalence class signatures allow users to sign representatives of the equivalence classes defined above, such that a representative and its corresponding signature can be adapted to give a fresh signature of a random representative in the same class. Below, we recall the formal definition of equivalence class signatures [HS14].

[7] Class-hiding was originally introduced [HS14] as a property of equivalence class signatures.

Definition 2 (EQS). *An equivalence class signature (EQS) scheme is defined with respect to a bilinear group description $\mathcal{BG}$ and a message length $\ell > 1$. An EQS scheme is a tuple of* PPT *algorithms* $(\mathsf{KGen}, \mathsf{Sign}, \mathsf{ChgRep}, \mathsf{Vf}, \mathsf{VfKey})$ *defined as follows:*

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(\mathcal{BG}, 1^\ell)$*: The key generation algorithm inputs a group $\mathcal{BG}$ and the message length 1^ℓ. It outputs a key pair* $(\mathsf{pk}, \mathsf{sk})$.

$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \bar{M})$*: The signing algorithm inputs the secret key* sk *and a message $\bar{M} \in \mathbb{G}_1^\ell$. It outputs a signature σ on the equivalence class $[\bar{M}]_\mathcal{R}$.*

$\sigma' \leftarrow \mathsf{ChgRep}(\mathsf{pk}, \bar{M}, \sigma, \rho)$*: The change representation algorithm inputs the public key* pk*, a message $\bar{M} \in \mathbb{G}_1^\ell$, a signature σ on the equivalence class $[\bar{M}]_\mathcal{R}$, and a scalar ρ. It outputs a new signature σ' on the (same) equivalence class $[\bar{M}^\rho]_\mathcal{R} = [\bar{M}]_\mathcal{R}$.*

$b \leftarrow \mathsf{Vf}(\mathsf{pk}, \bar{M}, \sigma)$*: The signature verification algorithm inputs the public key* pk*, a message $\bar{M} \in \mathbb{G}_1^\ell$, and a signature σ. It returns $b = 1$ if σ is a valid signature under* pk *on the equivalence class $[\bar{M}]_\mathcal{R}$, and $b = 0$ otherwise.*

$b \leftarrow \mathsf{VfKey}(\mathsf{pk}, \mathsf{sk})$*: The key verification algorithm inputs a public key* pk *and a secret key* sk*. It returns $b = 1$ if the keys are consistent and $b = 0$ otherwise.*

We refer the reader to [HS14] for a formal treatment of correctness. We define existential unforgeability under random message attacks (EUF-CMA) in the following.

Definition 3 (EUF-CMA). *An EQS scheme is said to be existentially unforgeable under chosen message attacks (EUF-CMA) if for all $\ell > 1$, for all $n \in \mathsf{poly}(\lambda)$, and for all PPT adversaries $\mathcal{A}$,*

$$\Pr\left[\begin{array}{l|l} 1 = \mathsf{Vf}(\mathsf{pk}, M^*, \sigma^*) \wedge & \mathcal{BG} \leftarrow \mathsf{BGGen}(1^\lambda); \\ \forall M \in Q : [M]_\mathcal{R} \neq [M^*]_\mathcal{R} & : (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(\mathcal{BG}, 1^\ell); \\ & (M^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}, \cdot)}(\mathsf{pk}) \end{array}\right] \leq \mathsf{negl}(\lambda)\,.$$

Next we recall the notion of signature adaptation which captures the fact that signatures output by ChgRep are distributed like fresh signatures on the new representative.

Definition 4 (Perfect Signature Adaptation). *An EQS scheme is said to perfectly adapt signatures if for all tuples $(\mathsf{sk}, \mathsf{pk}, \bar{M}, \sigma, \rho)$ such that $\mathsf{VfKey}(\mathsf{pk}, \mathsf{sk}) = 1$, $\mathsf{Vf}(\mathsf{pk}, \bar{M}, \sigma) = 1$, $\bar{M} \in \mathbb{G}_1^\ell$ for some $\ell > 1$, and $\rho \leftarrow \mathbb{Z}_q^*$ it holds that*

$$\mathsf{ChgRep}(\mathsf{pk}, \bar{M}, \sigma, \rho) \text{ and } \mathsf{Sign}(\mathsf{sk}, \bar{M}^\rho)$$

are identically distributed.

2.3 Verifiable Ring Signatures

Ring Signatures allow users to sign a message anonymously within a group of users, where the group is chosen upon signature creation in an ad-hoc way. *Verifiable Ring Signatures* (VRS) allow each user of the group to prove *a posteriori* whether he is the signer of a given message or not. VRS was formally defined and constructed in [LX03]. Below, we recall the syntax of VRS.

Definition 5 (Verifiable Ring Signature (VRS)). *A* Verifiable Ring Signature *(VRS) scheme is a tuple of six algorithms* $\mathsf{VRS} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Prove}, \mathsf{Judge})$ *defined as follows:*

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*: On input the security parameter 1^λ, return the public parameters* pp.

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(\mathsf{pp})$*: On input the public parameters* pp*, return a pair of signer public/private keys* $(\mathsf{pk}, \mathsf{sk})$.

$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, L, m)$*: On input the secret key* sk*, a ring L, and a message m, return a signature σ on the message m under the set of public keys L.*

$b \leftarrow \mathsf{Verify}(L, m, \sigma)$*: On input a ring L, a message m, and a signature σ, return a bit b or the distinguished symbol $\perp$.*

$\pi \leftarrow \mathsf{Prove}(L, m, \sigma, \mathsf{pk}, \mathsf{sk})$*: On input a ring L, a message m, a signature σ, a public key* pk*, and a secret key* sk*, return a proof π.*

$b \leftarrow \mathsf{Judge}(L, m, \sigma, \mathsf{pk}, \pi)$: *On input a ring L, a message m, a signature σ, a public key* pk*, and a proof π, return a bit b or the distinguished symbol $\perp$. By convention, if $b = 1$ (resp. 0) then π proves that σ was (resp. was not) generated by the signer corresponding to the public key* pk.

A VRS is required to be (strongly) unforgeable, (strongly) accountable, anonymous, and (strongly) non-seizable. For their formal definitions we refer to Appendix A.

3 Definition of Sanitizable Signatures

In the following we recall the syntax of sanitizable signatures. Let the signer and the sanitizer be denoted by S and Z respectively. Throughout this work we consider messages $m = (m_1, \ldots, m_\ell)$ to be tuples of ℓ parts for some $\ell > 1$, where $m_k \in \{0,1\}^*$ for all $k \in \ell$, and represent the admissible modification as a bit string $\alpha = \alpha_1 \| \ldots \| \alpha_\ell \in \{0,1\}^\ell$. We write $\alpha_k = 1$ if and only if the k-th block is admissible. For ease of exposition, we sometimes write $k \in \alpha$ instead of $\alpha_k = 1$.

Let δ be a function which maps a message m to another message $m' = \delta(m)$. Also, we say that δ is an admissible modification, denoted by $\alpha(\delta) = 1$, if and only if for all messages m and $m' = \delta(m)$, it holds that $m'_k = m_k$ for all $k \in \alpha$.

Definition 6 (Sanitizable Signature Scheme). *A sanitizable signature scheme consists of the* PPT *algorithms* $(\mathsf{Setup}, \mathsf{KGen_S}, \mathsf{KGen_Z}, \mathsf{Sign}, \mathsf{San}, \mathsf{Verify}, \mathsf{Prove}, \mathsf{Judge})$.

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$: *The setup algorithm inputs the security parameter 1^λ and the (maximum) length 1^ℓ of the messages and creates a public parameter* pp.

$(\mathsf{pk_S}, \mathsf{sk_S}) \leftarrow \mathsf{KGen_S}(\mathsf{pp})$: *The signer key generation algorithm inputs the public parameter* pp *and outputs* $(\mathsf{pk_S}, \mathsf{sk_S})$*, the public and secret key of the signer respectively.*

$(\mathsf{pk_Z}, \mathsf{sk_Z}) \leftarrow \mathsf{KGen_Z}(\mathsf{pp})$: *The sanitizer key generation algorithm inputs the public parameter* pp *and outputs* $(\mathsf{pk_Z}, \mathsf{sk_Z})$*, the public and secret key of the sanitizer respectively.*

$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk_S}, \mathsf{pk_Z}, m, \alpha)$: *The signing algorithm inputs a message $m \in (\{0,1\}^*)^\ell$, 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 σ.*

$\sigma' \leftarrow \mathsf{San}(\mathsf{pk_S}, \mathsf{sk_Z}, m, \delta, \sigma)$: *The sanitizing algorithm takes as input a message $m \in (\{0,1\}^*)^\ell$, 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 outputs a new signature σ' .*

$b \leftarrow \mathsf{Verify}(\mathsf{pk_S}, \mathsf{pk_Z}, m, \sigma)$: *The verification algorithm inputs a message m, a signature σ, a signer public key* $\mathsf{pk_S}$*, as well as a sanitizer public key* $\mathsf{pk_Z}$ *and outputs a bit b.*

$\pi \leftarrow \mathsf{Prove}(\mathsf{sk_S}, \mathsf{pk_Z}, m, \sigma)$: *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.*

$d \leftarrow \mathsf{Judge}(\mathsf{pk_S}, \mathsf{pk_Z}, m, \sigma, \pi)$: *The judge algorithm inputs 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.*

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].

3.1 Unlinkability and Invisibility

In the original definition of unlinkability by Brzuska *et al.* [BFLS10], the property was modeled using an experiment where the adversary gets access to, among other oracles, a "left-or-right sanitize" oracle $\mathsf{LoRSanit}\mathcal{O}$, which inputs two message-modification-signature tuples and outputs a sanitized signature produced from one of the tuples. The adversary's task is to decide which tuple is used for the sanitization.

$\mathsf{Sign}\mathcal{O}(\mathsf{pk}_\mathsf{Z}, m, \alpha)$

$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_\mathsf{S}^\dagger, \mathsf{pk}_\mathsf{Z}, m, \alpha)$

$L := L \| \{(\mathsf{pk}_\mathsf{S}^\dagger, \mathsf{pk}_\mathsf{Z}, m, \alpha, \sigma)\}$

return σ

$\mathsf{Prove}\mathcal{O}(\mathsf{pk}_\mathsf{Z}, m, \sigma)$

if $\begin{cases} \mathsf{pk}_\mathsf{Z} = \mathsf{pk}_\mathsf{Z}^\dagger \\ (m, \sigma) \in \mathcal{Q} \end{cases}$ **then return** $\bot$

// Only triggered in transparency experiment.

$\beta := \mathsf{Vf}(\mathsf{pk}_\mathsf{S}^\dagger, \mathsf{pk}_\mathsf{Z}, m, \sigma)$

if $\beta = 0$ **then return** $\bot$

$\pi \leftarrow \mathsf{Prove}(\mathsf{sk}_\mathsf{S}^\dagger, \mathsf{pk}_\mathsf{Z}, m, \sigma)$

return π

$\mathsf{San}\mathcal{O}'(\mathsf{pk}_\mathsf{S}, m, \delta, \sigma)$

if $\mathsf{pk}_\mathsf{S} \neq \mathsf{pk}_\mathsf{S}^\dagger$ **then**

  $(m', \sigma') \leftarrow \mathsf{San}(\mathsf{pk}_\mathsf{S}, \mathsf{sk}_\mathsf{Z}^\dagger, m, \delta, \sigma)$

  return (m', σ')

elseif $\exists \alpha \; s.t. \; (m, \sigma, \alpha) \in \mathcal{R} \wedge \delta \in \alpha$ **then**

  $(m', \sigma') \leftarrow \mathsf{San}(\mathsf{pk}_\mathsf{S}, \mathsf{sk}_\mathsf{Z}^\dagger, m, \delta, \sigma)$

  $\mathcal{R} \leftarrow \mathcal{R} \| \{(m', \sigma', \alpha)\}$

  return (m', σ')

endif

return $\bot$

$\mathsf{wLoRSanit}\mathcal{O}_b(i_0, \delta_0, i_1, \delta_1)$

if $L[i_0] = \epsilon \vee L[i_1] = \epsilon$ **then**

  return $\bot$

endif

$(\mathsf{pk}_{\mathsf{S},0}, \mathsf{pk}_{\mathsf{Z},0}, m_0, \alpha_0, \sigma_0) := L[i_0]$

$(\mathsf{pk}_{\mathsf{S},1}, \mathsf{pk}_{\mathsf{Z},1}, m_1, \alpha_1, \sigma_1) := L[i_1]$

foreach $\beta \in \{0, 1\}$ **do**

  $\sigma'_\beta \leftarrow \mathsf{San}(\mathsf{pk}_\mathsf{S}^\dagger, \mathsf{sk}_\mathsf{Z}^\dagger, m_\beta, \delta_\beta, \sigma_\beta)$

endfor

if $\begin{cases} \alpha_0 = \alpha_1 \\ \delta_0 \in \alpha_0 \wedge \delta_1 \in \alpha_1 \\ \delta_0(m_0) = \delta_1(m_1) \end{cases}$ **then**

  $L := L \| \{(\mathsf{pk}_\mathsf{S}^\dagger, \mathsf{pk}_\mathsf{Z}^\dagger, m_b, \alpha_b, \sigma_b)\}$

  return σ_b

endif

return $\bot$

$\mathsf{San}\mathcal{O}(\mathsf{pk}_\mathsf{S}, m, \delta, \sigma)$

$\alpha \leftarrow \mathsf{ExtAdm}(\mathsf{pk}_\mathsf{S}, \mathsf{sk}_\mathsf{Z}^\dagger, \sigma)$

if $\begin{cases} \mathsf{Verify}(\mathsf{pk}_\mathsf{S}, \mathsf{pk}_\mathsf{Z}^\dagger, m, \sigma) = 1 \\ \delta \in \alpha \end{cases}$ **then**

  $\sigma' \leftarrow \mathsf{San}(\mathsf{pk}_\mathsf{S}, \mathsf{sk}_\mathsf{Z}^\dagger, m, \delta, \sigma)$

  $L := L \| \{(\mathsf{pk}_\mathsf{S}, \mathsf{pk}_\mathsf{Z}^\dagger, \delta(m), \alpha, \sigma')\}$

  return σ'

endif

return $\bot$

$\mathsf{LoRAdm}\mathcal{O}_b(\mathsf{pk}_\mathsf{Z}, m, \alpha_0, \alpha_1)$

if $\begin{cases} |\alpha_0| = |\alpha_1| = |m| \\ \mathsf{pk}_\mathsf{Z} = \mathsf{pk}_\mathsf{Z}^\dagger \vee \alpha_0 = \alpha_1 \end{cases}$ **then**

  $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_\mathsf{S}^\dagger, \mathsf{pk}_\mathsf{Z}, m, \alpha_b)$

  if $\mathsf{pk}_\mathsf{Z} = \mathsf{pk}_\mathsf{Z}^\dagger$ **then**

    $\mathcal{R} := \mathcal{R} \| \{(m, \sigma, \alpha_0 \circ \alpha_1)\}$

  endif

  return σ

endif

return $\bot$

$\mathsf{Sign/San}\mathcal{O}_b(m, \delta, \alpha)$

if $\delta \notin \alpha$ **then return** $\bot$

$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_\mathsf{S}^\dagger, \mathsf{pk}_\mathsf{S}^\dagger, m, \alpha)$

$\sigma'_\beta \leftarrow \begin{cases} \mathsf{Sign}(\mathsf{sk}_\mathsf{S}^\dagger, \mathsf{pk}_\mathsf{S}^\dagger, \delta(m), \alpha) & b = 0 \\ \mathsf{San}(\mathsf{pk}_\mathsf{S}^\dagger, \mathsf{sk}_\mathsf{Z}^\dagger, m, \delta, \alpha) & b = 1 \end{cases}$

$\mathcal{Q} = \mathcal{Q} \| (\delta(m), \sigma')$

return σ'

$\mathsf{LoRSanit}\mathcal{O}_b(m_0, \delta_0, \sigma_0, m_1, \delta_1, \sigma_1)$

$b_\beta \leftarrow \mathsf{Verify}(\mathsf{pk}_\mathsf{S}^\dagger, \mathsf{pk}_\mathsf{Z}^\dagger, m_\beta, \sigma_\beta), \; \forall \beta \in \{0, 1\}$

if $b_0 = 0 \vee b_1 = 0$ **then return** $\bot$

foreach $\beta \in \{0, 1\}$ **do**

  $\alpha_\beta \leftarrow \mathsf{ExtAdm}(\mathsf{pk}_\mathsf{S}^\dagger, \mathsf{sk}_\mathsf{Z}^\dagger, \sigma_\beta)$

  $\sigma'_\beta \leftarrow \mathsf{San}(\mathsf{pk}_\mathsf{S}^\dagger, \mathsf{sk}_\mathsf{Z}^\dagger, m_\beta, \delta_\beta, \sigma_\beta)$

endfor

if $\begin{cases} \alpha_0 = \alpha_1 \\ \delta_0 \in \alpha_0 \wedge \delta_1 \in \alpha_1 \\ \delta_0(m_0) = \delta_1(m_1) \end{cases}$ **then**

  $L := L \| \{(\mathsf{pk}_\mathsf{S}^\dagger, \mathsf{pk}_\mathsf{Z}^\dagger, m_b, \alpha_b, \sigma_b)\}$

  return σ_b

endif

return $\bot$

Fig. 2: Oracles for Sanitizable Signatures

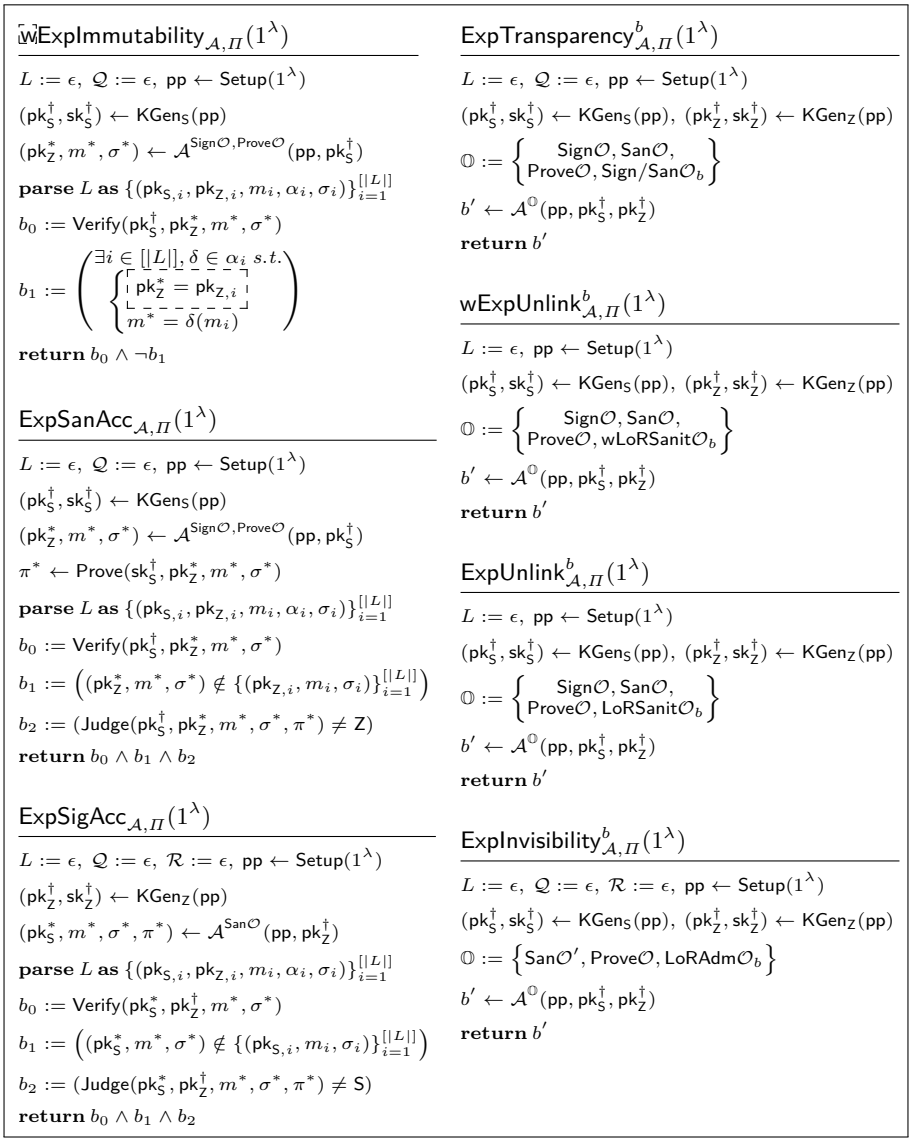

Fig. 3: Security Experiments for Sanitizable Signatures. Oracles are defined in Figure 2.

To define $\mathsf{LoRSanit}\mathcal{O}$, Brzuska *et al.* assumed that the description of admissible modifications α can be recovered from a valid signature, so that $\mathsf{LoRSanit}\mathcal{O}$ can recover the admissible modifications from both input signatures and check whether they are equal. Note that if this check is omitted, then the adversary can trivially decide which signature is used by querying the sanitize oracle $\mathsf{San}\mathcal{O}$ on the output of $\mathsf{LoRSanit}\mathcal{O}$.

Brzuska *et al.* did not explicitly state if such recovery can be done publicly or requires a secret key. Indeed, in all existing constructions of unlinkable sanitizable signatures [BFLS10], the recovery mechanism is public, which violates invisibility. Therefore, to achieve unlinkability and invisibility simultaneously, we must explicitly state that the admissible modifications can be recovered from a valid signature (hopefully only) with the corresponding sanitizer secret key. We say that a sanitizable signature scheme has privately extractable admissible modifications, if there exists a PPT algorithm ExtAdm which performs the following:

$\alpha \leftarrow \mathsf{ExtAdm}(\mathsf{pk_S}, \mathsf{sk_Z}, \sigma)$: The admissible modifications extraction algorithm inputs a signer public key $\mathsf{pk_S}$, a sanitizer secret key $\mathsf{sk_Z}$, and a signature. It outputs a description α of the admissible modifications.

In what follows, we only consider sanitizable signature schemes which have privately extractable admissible modifications.

3.2 Security of Sanitizable Signatures

We require a sanitizable signature scheme to be immutable, strongly accountable, strongly invisible, strongly proof-restrictedly transparent, and unlinkable. Different variations of these properties were defined in the literature [BFF+09, BFLS10, KSS16, BCD+17]. We will recall the definitions below for completeness. Additionally, we define the notions of weak immutability and weak unlinkability, which are achieved by our first construction. Our second construction then upgrades these properties to their regular counterparts.

We remark that (strong) unforgeability and privacy were considered in the literature. It is known that (strong) signer accountability and (strong) sanitizer accountability together imply (strong) unforgeability, while (strong) proof-restricted transparency implies proof-restricted privacy [BFF+09, KSS16]. Unlinkability is also shown to imply privacy [BFLS10]. We therefore do not consider unforgeability and privacy explicitly.

Immutability. Immutability requires that a malicious sanitizer cannot change inadmissible blocks. That is, an adversary should not be able to produce a forgery $(\mathsf{pk}_\mathsf{Z}^*, m^*, \sigma^*)$, such that m^* cannot be produced by any admissible modifications delegated to pk_Z^*. Note that the set of admissible modifications of a signature is bound to (the public key of) the sanitizer to which the signature is issued. We also consider a relaxed notion called weak immutability, where a forgery is not considered valid if m^* can be produced by a modification which is admissible for *some* (not necessarily pk_Z^*) sanitizers.

Definition 7 (Immutability [BFF+09]). *A sanitizable signature scheme Π is said to be* immutable *if for all PPT adversaries $\mathcal{A}$, the probability that the experiment* $\mathsf{Pr}\left[\mathsf{ExpImmutability}_{\mathcal{A},\Pi}(1^\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$ *where* $\mathsf{ExpImmutability}_{\mathcal{A},\Pi}(1^\lambda)$ *is defined in Figure 3. Additionally, we say that Π is* weakly *immutable if, in the experiment* $\mathsf{wExpImmutability}_{\mathcal{A},\Pi}(1^\lambda)$*, the condition* $\boxed{\mathsf{pk}_\mathsf{Z}^* = \mathsf{pk}_{\mathsf{Z},i}}$ *in the dashed box is dropped.*

Strong Transparency. Transparency means that sanitized signatures look like non-sanitized signatures. Rigorously speaking, transparency cannot be achieved if one is given oracle access to a prove oracle, which distinguishes sanitized signatures from fresh signatures. A relaxed notion, known as proof-restricted transparency is thus considered, which requires that one cannot decide whether a signature is sanitized or fresh, without the help of the prove oracle.

Definition 8 (Strong (Proof-Restricted) Transparency [KSS16]). *A sanitizable signature scheme Π is* strongly proof-restrictedly transparent *if for all PPT adversaries $\mathcal{A}$,*

$$\left|\mathsf{Pr}\left[\mathsf{ExpTransparency}^0_{\mathcal{A},\Pi}(1^\lambda) = 1\right] - \mathsf{Pr}\left[\mathsf{ExpTransparency}^1_{\mathcal{A},\Pi}(1^\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{ExpTransparency}^b_{\mathcal{A},\Pi}(1^\lambda)$ *is defined in Figure 3. If*

$$\mathsf{Pr}\left[\mathsf{ExpTransparency}^0_{\mathcal{A},\Pi}(1^\lambda) = 1\right] = \mathsf{Pr}\left[\mathsf{ExpTransparency}^1_{\mathcal{A},\Pi}(1^\lambda) = 1\right]$$

then we say Π is perfectly strongly proof-restrictedly transparent. Furthermore, if the step $\boxed{\mathcal{Q} = \mathcal{Q} \| (\delta(m), \sigma')}$ *in the* $\mathsf{Sign}/\mathsf{San}\mathcal{O}_b$ *oracle (highlighted in the dashed box) is dropped, so that $\mathcal{Q}$ remains empty throughout the experiment, then we simply say Π is perfectly strongly transparent.*

Strong Accountability. This property demands that the origin of a (possibly sanitized) signature should be undeniable by the signer.

Definition 9 (Strong Sanitizer-Accountability [KSS16]). *A sanitizable signature scheme Π is* strongly sanitizer-accountable *if for all PPT adversaries $\mathcal{A}$,*

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

where $\mathsf{ExpSanAcc}_{\mathcal{A},\Pi}(1^\lambda)$ *is defined in Figure 3.*

Definition 10 (Strong Signer-Accountability [KSS16]). *A sanitizable signature scheme Π is* strongly signer-accountable *if for all PPT adversaries $\mathcal{A}$,*

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

where $\mathsf{ExpSigAcc}_{\mathcal{A},\Pi}(1^\lambda)$ *is defined in Figure 3.*

Invisibility. Invisibility requires that the admissible modifications of a signature are hidden from an external observer.

Definition 11 (Strong Invisibility [BCD+17]). *A sanitizable signature scheme Π is* strongly invisible *if for all PPT adversaries $\mathcal{A}$,*

$$|\mathsf{Pr}\left[\mathsf{ExpInvisibility}^0_{\mathcal{A},\Pi}(1^\lambda) = 1\right] - \mathsf{Pr}\left[\mathsf{ExpInvisibility}^1_{\mathcal{A},\Pi}(1^\lambda) = 1\right]| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{ExpInvisibility}^b_{\mathcal{A},\Pi}(1^\lambda)$ *is defined in Figure 3.*

Unlinkability. Unlinkability means that one cannot decide the source of a given sanitized signature, unless it is revealed trivially by the message. The notion is modeled by considering an experiment where the adversary is given a "left-or-right sanitize" oracle $\mathsf{LoRSanit}\mathcal{O}$ which, on input two signatures, sanitizes one of them and returns the resulting signature. We also consider a relaxed notion called weak unlinkability, where the adversary is only allowed to query $\mathsf{LoRSanit}\mathcal{O}$ on honestly generated signatures.

Definition 12 (Weak Unlinkability). *A sanitizable signature scheme* SS *is* weakly unlinkable *if for all PPT adversaries $\mathcal{A}$,*

$$|\mathsf{Pr}\left[\mathsf{wExpUnlink}^0_{\mathcal{A},\mathsf{SS}}(1^\lambda) = 1\right] - \mathsf{Pr}\left[\mathsf{wExpUnlink}^1_{\mathcal{A},\mathsf{SS}}(1^\lambda) = 1\right]| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{wExpUnlink}^b_{\mathcal{A},\mathsf{SS}}(1^\lambda)$ *is defined in Figure 3.*

Definition 13 (Unlinkability [BFF+09]). *A sanitizable signature scheme* SS *is* unlinkable *if for all PPT adversaries $\mathcal{A}$,*

$$|\mathsf{Pr}\left[\mathsf{ExpUnlink}^0_{\mathcal{A},\mathsf{SS}}(1^\lambda) = 1\right] - \mathsf{Pr}\left[\mathsf{ExpUnlink}^1_{\mathcal{A},\mathsf{SS}}(1^\lambda) = 1\right]| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{ExpUnlink}^b_{\mathcal{A},\mathsf{SS}}(1^\lambda)$ *is defined in Figure 3.*

4 Construction

We propose a two-step construction of sanitizable signatures with immutability, strong accountability, strong proof-restricted transparency, strong invisibility, and unlinkability. In the first step, using equivalence class signatures and other basic primitives, we construct a scheme with weak immutability, perfect strong transparency, strong invisibility, and weak unlinkability. This scheme does not achieve accountability. Next, we show how one can transform any schemes with these properties one with all desirable properties, using verifiable ring signatures.

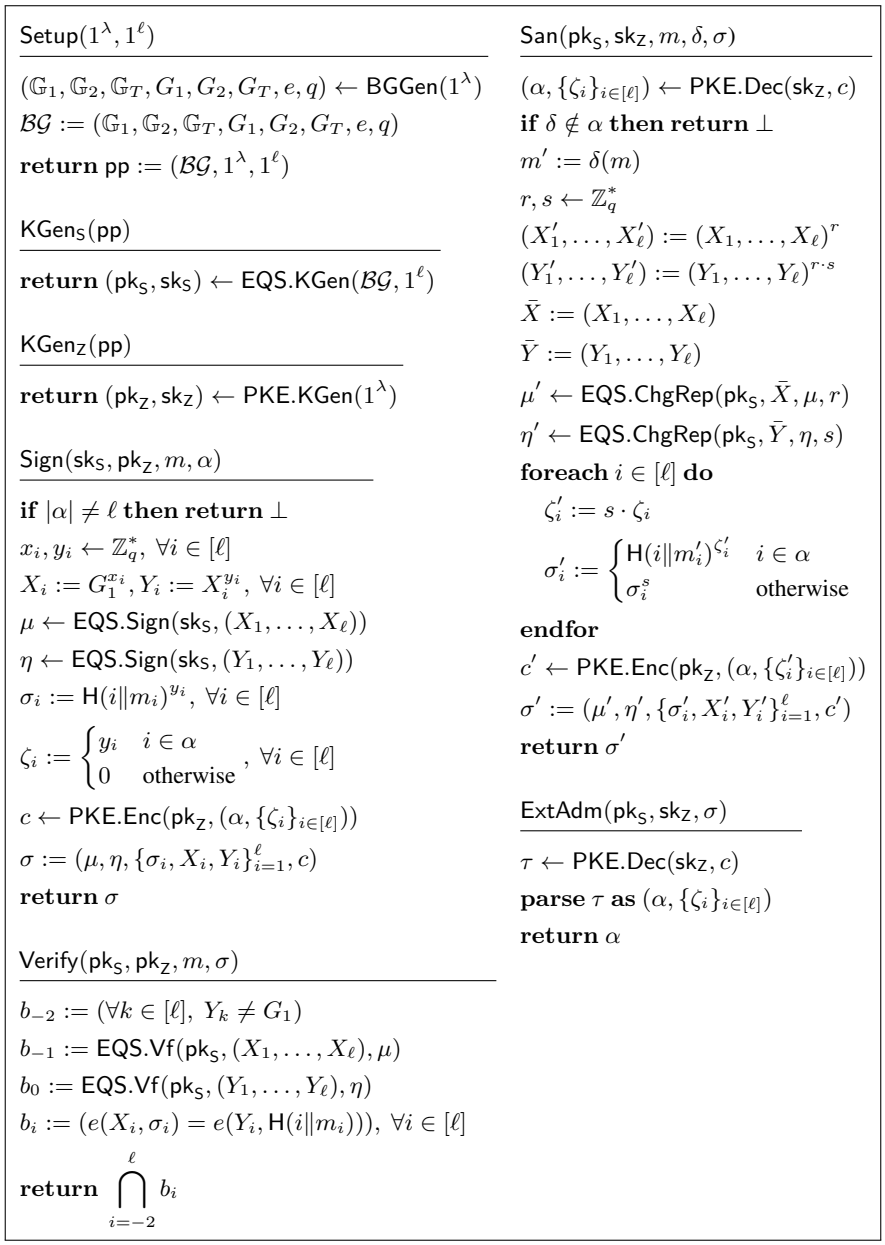

Fig. 4: Construction of Weak Sanitizable Signatures. Prove and Judge always output ϵ (the empty string) and S (the signer) respectively, and are omitted.

4.1 Construction I: Achieving Unlinkability and Invisibility

Let $\ell > 1$ be an integer. Let EQS be an equivalence class signature scheme, $\mathsf{H} : \{0,1\}^* \to \mathbb{G}_2$ be a hash function (to be modeled as a random oracle), and PKE be a public-key encryption scheme. We present in Figure 4 a construction of sanitizable signatures Π_1. The construction satisfies weak immutability, strong invisibility, perfect strong transparency, and weak unlinkability, but not accountability.

Informally, the signer issues signatures as follows. On input a message $m = m_1 \| \dots \| m_\ell$, the signer samples ℓ fresh BLS-like public and secret keys, which are used to sign the ℓ messages. Concretely, the i-th public key consists of a tuple $(X_i, Y_i) \in \mathbb{G}_1^2$ with $Y_i = X_i^{y_i}$ for some $y_i \in \mathbb{Z}_q$, and the secret key is y_i. It then signs the vectors $\bar{X} = (X_1, \dots, X_\ell)$ and $\bar{Y} = (Y_1, \dots, Y_\ell)$ using EQS. Next, in the same way as in [BCD+17], it encrypts the BLS-like secret keys y_i corresponding to the admissible message blocks using the PKE public key of the sanitizer. Finally, it outputs the signature which consists of two EQS signatures, ℓ BLS-like signatures and public keys, and a PKE ciphertext.

To sanitize, the sanitizer decrypts the PKE ciphertext and obtains the BLS-like secret keys corresponding to the admissible blocks, which are then used to sign the corresponding modified messages. Using the homomorphic property of the BLS-like scheme, the sanitizer can rerandomize $\bar{X}$ and $\bar{Y}$ to $\bar{X}^r$ and $\bar{Y}^{r \cdot s}$ respectively, and rerandomize the signatures accordingly so that they are compatible with the new public keys. Using the signature adaptation properties of EQS, it can also obtain fresh-looking EQS signatures on $\bar{X}^r$ and $\bar{Y}^{r \cdot s}$ respectively. Finally, the sanitizer re-encrypts the new BLS-like secret keys, and outputs the signature.

Since we do not aim to provide accountability, the prove algorithm always returns the empty string ϵ and the judge algorithm always outputs S. The correctness of Π_1 follows trivially from the correctness of the building blocks. Below, we state our main theorem and we defer its proof to Section 5.

Theorem 1. *Let $q > 2^\lambda$. If EQS is EUF-CMA-secure, then Π_1 is weakly immutable in the generic group and random oracle model. If PKE is IND-CCA-secure, then Π_1 is strongly invisible. If EQS perfectly adapts signatures, then Π_1 is perfectly strongly transparent (and hence also perfectly strongly proof-restrictedly transparent). If the equivalence relation $\mathcal{R}$ is class-hiding, EQS perfectly adapt signatures, and PKE is correct and is IND-CCA-secure, then Π_1 is weakly unlinkable in the generic group random oracle model.*

Finally, we remark that it is trivial to extend the construction to the multi-sanitizer setting by encrypting the (possibly different subsets of) BLS-like keys for different sanitizers.

4.2 Construction II: Generic Transformation for Accountability

In this section, we show a generic transformation (Figure 5), from any weakly immutable, non-accountable, strongly invisible, strongly proof-restricted transparent, and weakly unlinkable schemes Π_1, to an immutable, strongly accountable, strongly invisible, strongly proof-restricted transparent, and unlinkable scheme Π_2, using a verifiable ring signature scheme VRS.

An overview of the transform follows. The signer signs the public key of the sanitizer and the message in σ_{SS} using Π_1, then signs σ_{SS} in σ_{VRS} using VRS, where the ring contains the public key of the signer and the public key of the sanitizer. The signer outputs the signature $\sigma = (\sigma_{\mathsf{SS}}, \sigma_{\mathsf{VRS}})$. To sanitize, the sanitizer sanitizes σ_{SS} using Π_1 to produce σ'_{SS}, then signs σ'_{SS} in σ'_{VRS} using VRS. The sanitized signature is $\sigma' = (\sigma'_{\mathsf{SS}}, \sigma'_{\mathsf{VRS}})$. To verify any signature $\sigma = (\sigma_{\mathsf{SS}}, \sigma_{\mathsf{VRS}})$, the verifier uses the verification algorithm of Π_1 on σ_{SS} and the verification algorithm of VRS on σ_{VRS}. To prove that a signature $\sigma' = (\sigma'_{\mathsf{SS}}, \sigma'_{\mathsf{VRS}})$ is sanitized, the signer proves that he did not generate σ'_{VRS} using the prove algorithm of VRS, which gives accountability.

Next, we sketch why the other security properties are preserved. For conciseness, we omit the qualitative attributes such as *strong* and *proof-restricted* in the following discussion. First, since Π_1 is weakly immutable, the sanitizer is not able to forge the part σ'_{SS} of a sanitized signature for a non-admissible message nor changing the sanitizer public key (which is signed as a message of Π_1). This implies that the resulting scheme is immutable. Next, since Π_1 is transparent, one cannot guess whether a signature is sanitized or not from the part σ_{SS}. On the other hand, since VRS is anonymous, one cannot guess whether the part σ_{VRS} was created by the signer or by the sanitizer. Combining both properties, we conclude that one cannot guess whether the signature was sanitized or not, *i.e.*, Π_2 is transparent. Thirdly, Π_1 is invisible, so the part σ_{SS} hides all information about the possible modifications of the message. Moreover, The signature σ_{VRS} contains no information about the modifiable parts of the message. This implies that Π_2 is invisible.

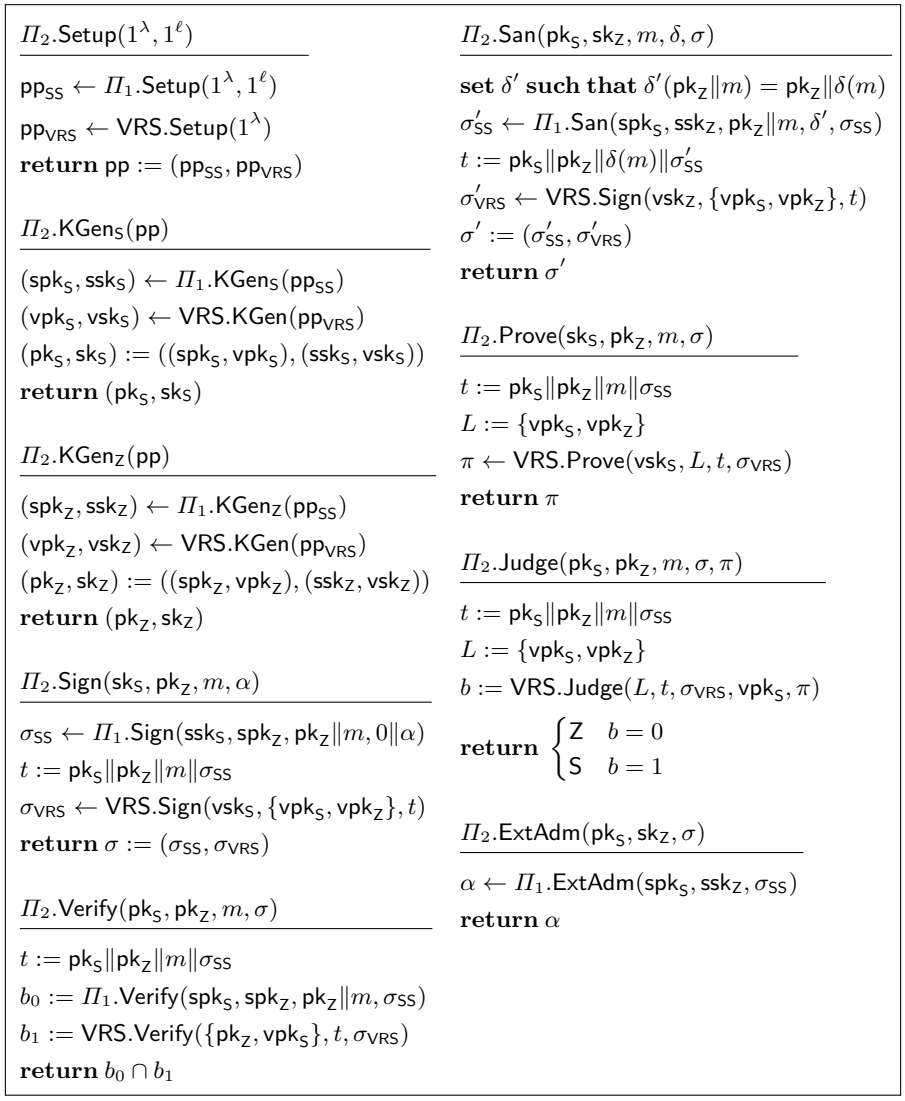

Fig. 5: Generic Transformation from Weak to Fully-Fledged Sanitizable Signatures.

Finally, Π_1 is unlinkable, so the first part σ'_{SS} of a sanitized signature hides any information about the original signature, and the second part σ'_{VRS} does not depend on the original signature, so our resulting scheme Π_2 is also unlinkable. Note that Π_1 is *weakly* unlinkable in the sense that it is no longer secure if the adversary is allowed to send fresh signatures to the oracle $\mathsf{LoRSanit}\mathcal{O}$. This does not impact the security of Π_2, because σ_{SS} is signed in σ_{VRS}, so to produce a fresh signature $(\sigma'_{\mathsf{SS}}, \sigma'_{\mathsf{VRS}})$, the adversary should be able to forge σ'_{VRS}, which is supposed to be hard under the hypothesis that the VRS scheme is unforgeable.

The correctness of Π_2 follows trivially from the correctness of Π_1 and VRS. Below, we state the formal security results for the construction. We refer to Appendix B for the formal security proofs.

Theorem 2. *If Π_1 is weakly immutable, then Π_2 is immutable. If Π_1 is weakly unlinkable, and* VRS *is strongly unforgeable, then Π_2 is unlinkable. If Π_1 is strongly invisible then Π_2 is strongly invisible. If* VRS *is strongly accountable, then Π_2 is strongly signer accountable. If* VRS *is strongly non-seizable, then Π_2 is strongly sanitizer accountable. If* VRS *is anonymous and Π_1 is strongly proof-restrictedly transparent, then Π_2 is strongly proof-restrictedly transparent.*

We remark that verifiable ring signatures can be constructed generically from linkable ring signatures [LWW04], which in turn can be generically constructed from unique ring signatures [FZ12]. It is also possible to use any stronger primitive such as traceable [FS07] or accountable [XY04] ring signatures, as long as the signers are accountable. Furthermore, the transform can be easily extended to a multi-sanitizer setting by signing with respect to a ring which consists of the signer and multiple sanitizers. Depending on the variant of ring signatures used, we obtain different flavors of accountability. As the implications are straightforward, we do not elaborate further.

5 Security Proof for Construction I

The following proof uses the generic group model abstraction of Shoup [Sho97] and we refer the reader to [BB04] for a comprehensive introduction to the bilinear group model. Here we state the central lemma useful for proving facts about generic attackers.

Lemma 2 (Schwartz-Zippel). *Let $F(X_1, \ldots, X_m)$ be a non-zero polynomial of degree $d \geq 0$ over a field $\mathbb{F}$. Then the probability that $F(x_1, \ldots, x_m) = 0$ for randomly chosen values $(x_1, \ldots, x_m)$ in $\mathbb{F}^n$ is bounded from above by $\frac{d}{|\mathbb{F}|}$.*

5.1 Weak Immutability

Proof (Weak Immutability). To prove that Π_1 is weakly immutable, we first show the generic hardness of the following problem.

Lemma 3. *Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_1, G_2, G_T, e, q) \leftarrow \mathsf{BGGen}(1^\lambda)$ with $q > 2^\lambda$, and $a, b, c \leftarrow \mathbb{Z}_q$. For all generic group adversary $\mathcal{A}$, the probability that $\mathcal{A}$ on input $(G_1, G_1^a, G_1^b, G_2, G_2^b, G_2^c)$ outputs $(G_1^u, G_1^v, G_1^x, G_1^y, G_2^z)$ such that*

$$\begin{cases} au - x = 0 \\ bv - y = 0 \\ cy - xz = 0 \\ v \neq 0 \end{cases}$$

is negligible.

Proof. Let $(G_1^u, G_1^v, G_1^x, G_1^y, G_2^z)$ be the output of $\mathcal{A}$. Since $\mathcal{A}$ is generic, it holds that

$$\begin{aligned} u &= u_1 + u_a a + u_b b \\ v &= v_1 + v_a a + v_b b \\ x &= x_1 + x_a a + x_b b \\ y &= y_1 + y_a a + y_b b \\ z &= z_1 + z_b b + z_c c \end{aligned}$$

for some coefficients $u_1, u_a, u_b, v_1, v_a, v_b, x_1, x_a, x_b, y_1, y_a, y_b, z_1, z_b, z_c \in \mathbb{Z}_q$. By the relation $au - x = 0$, we have $-x_1 + (u_1 - x_a)a - x_b b + u_a a^2 + u_b ab = 0$. Note that $f(A, B) := -x_1 + (u_1 - x_a)A - x_b B + u_a A^2 + u_b AB$ is a quadratic polynomial in the variables A and B. Suppose f is not a zero polynomial, by the Schwartz-Zippel lemma (Lemma 2), for $a, b \leftarrow \mathbb{Z}_q$, the probability that $f(a, b) = 0$ is upper bounded by $2/q < 2^{1-\lambda}$ which is negligible. Therefore we can assume that f is always zero. In particular, we have $x_1 = x_b = 0$. Similarly, by examining the relation $bv - y = 0$, we can assume that $v_1 = y_b$, and $y_1 = y_a = 0$. We can therefore write $x = x_a a$ and $y = y_b b$. Next, we examine the relation $cy - xz = 0$, which implies

$$y_b bc - x_a z_1 a - x_a z_b ab - x_a z_c ac = 0.$$

Using the Schwartz-Zippel lemma again, we can assume that $y_b = 0$. However, this means that $v = v_1 = y_b = 0$, which contradicts with the fourth relation $v \neq 0$. □

Now, suppose there exists a generic group adversary $\mathcal{A}$ against the weak immutability of Π_1. We construct a generic group adversary $\mathcal{C}$ which solves the problem defined in Lemma 3. $\mathcal{C}$ receives as challenge $(G_1, G_1^a, G_1^b, G_2, G_2^c)$ from its challenger. It then simulates the $\mathsf{ExpImmutability}$ experiment for $\mathcal{A}$ by setting the public parameters and the signer keys honestly. Without loss of generality, assume that $\mathcal{A}$ makes $Q_1 = \mathsf{poly}(\lambda)$ signing oracle queries and $Q_2 = \mathsf{poly}(\lambda)$ random oracle queries for $\mathsf{H}()$. $\mathcal{C}$ additionally samples $i^\dagger, j^\dagger \leftarrow [Q_1]$ as a guess of which Sign oracle query $\mathcal{A}$ will attack against, $k^\dagger \leftarrow [\ell]$ as the index of the inadmissible block that will be modified in the forgery message, $l^\dagger \leftarrow [Q_2]$ as a guess of which $\mathsf{H}()$ oracle query $\mathcal{A}$ will include as the inadmissible modification in the forgery message.

Answering Random Oracle Queries Upon receiving (k_l, m_l) as the l-th distinct query to the $\mathsf{H}()$ oracle, if $l \neq l^\dagger$, $\mathcal{C}$ answers the query by picking $t_l \leftarrow \mathbb{Z}_q^*$ and return $h_l := G_2^{t_l} \in \mathbb{G}_2$ to $\mathcal{A}$. If $l = l^\dagger$, then $\mathcal{C}$ sets $h_l = G_2^c$ where G_2^c was received as a challenge as described above. If (k_l, m_l) was a message that was queried previously, then reply with the same response as before.

Answering Sign Oracle Queries Upon receiving $(\mathsf{pk}_{\mathsf{Z},}, m_i, \alpha_i)$ as the i-th query to the $\mathsf{Sign}\mathcal{O}$ oracle, if $i \neq i^\dagger$ and $i \neq j^\dagger$, $\mathcal{C}$ answers the query honestly by running the procedures as defined in the $\mathsf{Sign}\mathcal{O}$ oracle.

In the case $i = i^\dagger$ or $i = j^\dagger$, $\mathcal{C}$ generates the signature honestly except for the following changes:

1. If $i = i^\dagger$, then $\mathcal{C}$ picks the elements $X_{i^\dagger,1}, \ldots, X_{i^\dagger,\ell}$ as follows. $\mathcal{C}$ picks $X_{i^\dagger,k^\dagger} = G_1^a$ which it had received from its challenger in the beginning. For all other $k \in [\ell] \setminus \{k^\dagger\}$, $\mathcal{C}$ generates the X_k honestly by picking $x_{i^\dagger,k} \leftarrow \mathbb{Z}_q^*$ and setting $X_{i^\dagger,k} = G_1^{x_{i^\dagger,k}}$ (as done in the $\mathsf{Sign}\mathcal{O}$ oracle). The rest of the signature is generated as in the $\mathsf{Sign}\mathcal{O}$ oracle.
2. Suppose $i = j^\dagger$. If $k^\dagger \in \alpha_{j^\dagger}$, or $(k^\dagger, m_{j^\dagger,k^\dagger}) = (k_{l^\dagger}, m_{l^\dagger})$, then abort. Otherwise, let $t^\dagger$ be such that $H(k^\dagger \| m_{j^\dagger,k^\dagger}) = G_2^{t^\dagger}$. $\mathcal{C}$ first generates $X_{i^\dagger,1}, \ldots, X_{i^\dagger,\ell}$ by picking $x_{j^\dagger,k} \leftarrow \mathbb{Z}_q^*$ and setting $X_{j^\dagger,k} := G_1^{x_{j^\dagger,k}}$ for all $k \in [\ell]$. Then $\mathcal{C}$ picks the elements $Y_{i^\dagger,1}, \ldots, Y_{i^\dagger,\ell}$ as follows. $\mathcal{C}$ picks $Y_{j^\dagger,k^\dagger} = G_1^b$ which it had received from its challenger in the beginning. It then generates $\sigma_{j^\dagger,k^\dagger}$ as $(G_2^b)^{\frac{t^\dagger}{x_{j^\dagger,k^\dagger}}}$. For all other $k \neq k^\dagger$, $\mathcal{C}$ generates the $Y_{j^\dagger,k}$ and the rest of the signature honestly as done in the $\mathsf{Sign}\mathcal{O}$ oracle. Note that as we assume $k^\dagger \notin \alpha_{j^\dagger}$ in this case, the value $y_{j^\dagger,k^\dagger}$ is not needed to generate the signature. Therefore the signature can be simulated faithfully.

Answering Prove Oracle Queries. The Prove oracle is trivially simulatable since the Prove algorithm always returns ϵ.

Clearly, assuming that $\mathcal{C}$ did not abort, $\mathcal{C}$ simulates the $\mathsf{wExpImmutability}$ experiment for $\mathcal{A}$ faithfully. Eventually, $\mathcal{A}$ outputs $(\mathsf{pk}_{\mathsf{Z}}^*, m^*, \sigma^*)$ as a forgery such that $\mathsf{Verify}(\mathsf{pk}_{\mathsf{S}}^\dagger, \mathsf{pk}_{\mathsf{Z}}^*, m^*, \sigma^*) = 1$, and $m_k^* \neq m_{i,k}$ for some i, k such that $k \notin \alpha_i$. Since $Q_1, \ell \in \mathsf{poly}(\lambda)$, with non-negligible probability it holds that $m_{k^\dagger}^* \neq m_{j^\dagger,k^\dagger}$ and $k^\dagger \notin \alpha_{j^\dagger}$. Moreover, since $Q_2 \in \mathsf{poly}(\lambda)$, with non-negligible probability it holds that $(k^\dagger, m_{k^\dagger}^*) = (k_{l^\dagger}, m_{l^\dagger})$. If that is the case, then the abort conditions in the above procedures of answering sign oracle queries are never triggered.

Parse σ^* as $(\mu^*, \eta^*, \{\sigma_j^*, X_j^*, Y_j^*\}_{j=1}^{\ell}, c^*)$. By the EUF-CMA-security of EQS, with overwhelming probability we have that $[X_1^*, \ldots, X_\ell^*]_{\mathcal{R}} = [X_{i^*,1}, \ldots, X_{i^*,\ell}]_{\mathcal{R}}$ for some i^*, and $[Y_1^*, \ldots, Y_\ell^*]_{\mathcal{R}} = [Y_{j^*,1}, \ldots, Y_{j^*,\ell}^*]_{\mathcal{R}}$ for some j^* (otherwise we can construct an adversary against the EUF-CMA-security of EQS by simply outputting μ or η). Therefore, there exists $r, s \in \mathbb{Z}_q$ such that $(X_{i^*,1}, \ldots, X_{i^*,\ell})^r = (X_1^*, \ldots, X_\ell^*)$ and $(Y_{j^*,1}, \ldots, Y_{j^*,\ell}^*)^{r \cdot s} = (Y_1^*, \ldots, Y_\ell^*)$.

Since $Q_1 \in \mathsf{poly}(\lambda)$, it happens with non-negligible probability that $(i^\dagger, j^\dagger) = (i^*, j^*)$. Suppose this is the case. Let k' be arbitrary with $k' \neq k^\dagger$. $\mathcal{C}$ extracts G_1^r and G_1^{rs} by computing

$$(X^*_{k'})^{\frac{1}{x_{i^*,k'}}} = G_1^{r\cdot\left(x_{i^*,k'}\right)\cdot\frac{1}{x_{i^*,k'}}} = G_1^r$$
$$(Y^*_{k'})^{\frac{1}{\left(x_{j^*,k'}\right)\cdot\left(y_{j^*,k'}\right)}} = (G_1^{r\cdot s})^{\frac{\left(x_{j^*,k'}\right)\cdot\left(y_{j^*,k'}\right)}{\left(x_{j^*,k'}\right)\cdot\left(y_{j^*,k'}\right)}} = G_1^{r\cdot s}.$$

Since $\mathsf{Verify}(\mathsf{pk}_\mathsf{S}, \mathsf{pk}^*_\mathsf{Z}, m^*, \sigma^*) = 1$, this implies that $Y^*_{k^\dagger} = G_1^{r\cdot s\cdot b} \neq G_1$. This means that $r \cdot s \neq 0$. Furthermore, we have

$$\begin{aligned}
e(X^*_{k^\dagger}, \sigma^*_{k^\dagger}) &= e(Y^*_{k^\dagger}, \mathsf{H}(k^\dagger \| m^*_{k^\dagger})) \\
e(X^r_{i^\dagger,k^\dagger}, \sigma^*_{k^\dagger}) &= e(Y^{r\cdot s}_{j^\dagger,k^\dagger}, G_2^c) \\
e(G_1^{r\cdot a}, \sigma^*_{k^\dagger}) &= e(G_1^{r\cdot s\cdot b}, G_2^c) \\
\sigma^*_{k^\dagger} &= G_2^{\frac{s\cdot b\cdot c}{a}}
\end{aligned}$$

Now, set $\mathcal{C}$ outputs $(G_1^u, G_1^v, G_1^x, G_1^y, G_2^z) := (G_1^r, G_1^{r\cdot s}, G_1^{r\cdot a}, G_1^{r\cdot s\cdot b}, G_2^{\frac{s\cdot b\cdot c}{a}})$. By a routine calculation, one can verify that $au - x = 0$, $bv - y = 0$, $cy - xz = 0$ and $v \neq 0$. Since $\mathcal{A}$ only performs generic group operations, so does $\mathcal{C}$, which contradicts with Lemma 3. □

5.2 Strong Invisibility

Proof (Strong Invisibility). We prove strong invisibility by hybrid argument. We define an intermediate experiment Hyb^b which is identical to $\mathsf{ExpInvisibility}^b_{\mathcal{A},\Pi}(1^\lambda)$ for both $b \in \{0,1\}$, except for the following changes: When answering $\mathsf{LoRAdm}\mathcal{O}_b$ oracle queries, the challenger signs with respect to the policy $\alpha_0 \circ \alpha_1$ instead of α_b. We argue that, in the view of the adversary, the experiments Hyb^b and $\mathsf{ExpInvisibility}^b_{\mathcal{A},\Pi}(1^\lambda)$ are computationally indistinguishable for $b \in \{0,1\}$. Suppose that is the case, since obviously Hyb^0 is functionally equivalent to Hyb^1, it holds that $\mathsf{ExpInvisibility}^0_{\mathcal{A},\Pi}(1^\lambda)$ is computationally indistinguishable to $\mathsf{ExpInvisibility}^1_{\mathcal{A},\Pi}(1^\lambda)$.

Before proving the claim above, we state two key observations. First, note that the signatures returned by the $\mathsf{LoRAdm}\mathcal{O}_b$ oracle in the all experiments are identically distributed if $\mathsf{pk}_\mathsf{Z} \neq \mathsf{pk}^\dagger_\mathsf{Z}$ (since now it must hold that $\alpha_0 = \alpha_1$ for the oracle to not abort). In the case $\mathsf{pk}_\mathsf{Z} = \mathsf{pk}^\dagger_\mathsf{Z}$, the signatures returned by the oracle are almost identically distributed, except for the ciphertext c. In particular, in the experiment $\mathsf{ExpInvisibility}^b_{\mathcal{A},\Pi}(1^\lambda)$, the ciphertext is an encryption of the message $(\alpha_b, \{\zeta_{b,i}\}_{i=0}^\ell)$, where $\zeta_{b,i} = x_i$ for all $i \in \alpha_b$, and is zero otherwise. On the other hand, in Hyb^b, the ciphertext is an encryption of the message $(\alpha_0 \circ \alpha_1, \{\zeta_{b,i}\}_{i=0}^\ell)$, where $\zeta_{b,i} = x_i$ for all $i \in \alpha_0 \circ \alpha_1$, and is zero otherwise.

The second observation is that, due to the restriction imposed on the $\mathsf{San}\mathcal{O}'$ oracle, the values x_i for all $i \in (\alpha_b - (\alpha_0 \circ \alpha_1))$ are never used in any experiments.

With the above observations, we show how one can construct an algorithm $\mathcal{C}$, which breaks the IND-CCA-security of PKE, using a distinguisher which distinguishes Hyb^b from $\mathsf{ExpInvisibility}^b_{\mathcal{A},\Pi}(1^\lambda)$. $\mathcal{C}$ receives a public key pk_PKE from the IND-CCA challenger, and acts as the challenger of either the experiment $\mathsf{ExpInvisibility}^b_{\mathcal{A},\Pi}(1^\lambda)$ or Hyb^b by setting $\mathsf{pk}^\dagger_\mathsf{Z} := \mathsf{pk}_\mathsf{PKE}$ and generating other keys honestly.

Let $(\mathsf{pk}_\mathsf{Z}, m_j, \alpha_{j,0}, \alpha_{j,1})$ be the$j$-th query to the $\mathsf{LoRAdm}\mathcal{O}_b$ oracle. $\mathcal{C}$ answers the query honestly if $\mathsf{pk}_\mathsf{Z} \neq \mathsf{pk}^\dagger_\mathsf{Z}$. In the case where $\mathsf{pk}_\mathsf{Z} = \mathsf{pk}^\dagger_\mathsf{Z}$, $\mathcal{C}$ generates the ciphertext c_j in the following way. It samples $y_{j,i} \leftarrow_\$ \mathbb{Z}_q^*$ for all $i \in [\ell]$, and

prepares $\zeta_{b,j,i} := \begin{cases} y_{j,i} & i \in \alpha_{j,b} \\ 0 & \text{otherwise} \end{cases}, \forall i \in [\ell]$ $\quad$ $\zeta'_{b,j,i} := \begin{cases} y_{j,i} & i \in \alpha_{j,0} \circ \alpha_{j,1} \\ 0 & \text{otherwise} \end{cases}, \forall i \in [\ell]$

$\tau_{b,j} := (\alpha_{j,b}, \{\zeta_{b,j,i}\}_{i=1}^\ell)$ $\quad$ $\tau'_{b,j} := (\alpha_{j,0} \circ \alpha_{j,1}, \{\zeta'_{b,j,i}\}_{i=1}^\ell)$

and queries the $\mathsf{Enc}\mathcal{O}_b$ oracle provided by the IND-CCA challenger on $(\tau_{b,j}, \tau'_{b,j})$ and receive c_j. $\mathcal{C}$ generates the rest of the signature honestly.

Upon receiving a query $(\mathsf{pk}_\mathsf{S}, m, \delta, \sigma)$ to the $\mathsf{San}\mathcal{O}'$ oracle, $\mathcal{C}$ parses σ as $(\mu, \eta, \{\sigma_i, X_i, Y_i\}_{i=1}^{\ell}, c)$ and checks if $c = c_j$ for some j. If so, it uses $\{y_{j,i}\}_{i \in \alpha_{j,0} \cap \alpha_{j,1}}$ to answer the oracle query. If not, it queries the $\mathsf{Dec}\mathcal{O}$ oracle provided by the IND-CCA challenger on c, receives $\tau = (\alpha, \{\zeta_i\}_{i \in \ell})$, and uses it to answer the oracle query.

Clearly, depending on the choice of the IND-CCA challenger, our adversary simulates either the experiment $\mathsf{Exp}\mathsf{sInvis}^{b}_{\Pi,\mathcal{A}}(1^\lambda)$ or Hyb^b faithfully. Therefore, if there exists a distinguisher which distinguishes the two experiments with a certain probability, then our adversary can guess the choice of the IND-CCA challenger with the same probability. □

5.3 Perfect Strong Transparency

Proof (Perfect Strong Transparency). We show that the construction is perfectly strongly transparent through hybrid argument. First, observe that the Prove algorithm, and hence also the $\mathsf{Prove}\mathcal{O}$ oracle, always returns the empty string ϵ, it is safe to drop the step $\mathcal{Q} = \mathcal{Q} \| (\delta(m), \sigma')$ in the $\mathsf{Sign}/\mathsf{San}\mathcal{O}_b$ oracle.

Now, let $Q = \mathsf{poly}(\lambda)$ be the number of queries that the adversary $\mathcal{A}$ make to the $\mathsf{Sign}/\mathsf{San}\mathcal{O}_b$ oracle. We define the hybrids $\mathsf{Hyb}_0, \ldots, \mathsf{Hyb}_q$ as follows. The hybrid Hyb_0 is identical to $\mathsf{Exp}\mathsf{Trans}^0_{\Pi,\mathcal{A}}(1^\lambda)$. For $j \in [Q]$, Hyb_j is almost identical to Hyb_{j-1}, except that in the former the j-th query to the $\mathsf{Sign}/\mathsf{San}\mathcal{O}_b$ is answered as in $\mathsf{Exp}\mathsf{Trans}^1_{\Pi,\mathcal{A}}(1^\lambda)$. That is, the first j signatures returned by $\mathsf{Sign}/\mathsf{San}\mathcal{O}_b$ are sanitized, while the last $Q - j$ signatures are freshly signed. Note that Hyb_Q is identical to $\mathsf{Exp}\mathsf{Trans}^1_{\Pi,\mathcal{A}}(1^\lambda)$. Obviously, if $\mathsf{Pr}\left[\mathsf{Hyb}_{j-1} = 1\right] = \mathsf{Pr}\left[\mathsf{Hyb}_j = 1\right]$ for all $j \in [Q]$, then $\mathsf{Pr}\left[\mathsf{Exp}\mathsf{Trans}^0_{\Pi,\mathcal{A}}(1^\lambda) = 1\right] = \mathsf{Pr}\left[\mathsf{Exp}\mathsf{Trans}^1_{\Pi,\mathcal{A}}(1^\lambda) = 1\right]$.

Fix $j \in [Q]$. In the following, we show that $\mathsf{Pr}\left[\mathsf{Hyb}_{j-1} = 1\right] = \mathsf{Pr}\left[\mathsf{Hyb}_j = 1\right]$. Let (m, δ, α) be the j-th query of $\mathcal{A}$ to the $\mathsf{Sign}/\mathsf{San}\mathcal{O}_b$ oracle. If $\delta \notin \alpha$, then the oracle returns $\perp$ in both experiments and thus the equality holds trivially. Otherwise, let $m' := \delta(m)$, and let σ' be the response. In Hyb_{j-1}, the signature σ' is drawn from a distribution $\mathcal{D}$ where

$$\mathcal{D} := \left\{ \sigma : \begin{array}{l} x_i, y_i \leftarrow \mathbb{Z}_q^*,\ X_i := G_1^{x_i},\ Y_i := X_i^{y_i},\ \forall i \in [\ell] \\ \mu \leftarrow \mathsf{EQS.Sign}(\mathsf{sk}_\mathsf{S}^\dagger, (X_1, \ldots, X_\ell)) \\ \eta \leftarrow \mathsf{EQS.Sign}(\mathsf{sk}_\mathsf{S}^\dagger, (Y_1, \ldots, Y_\ell)) \\ \sigma_i \leftarrow \mathsf{H}(i \| m_i')^{y_i},\ \forall i \in [\ell] \\ \zeta_i := \begin{cases} y_i & i \in \alpha \\ 0 & \text{otherwise} \end{cases},\ \forall i \in [0, \ell] \\ \tau := (\alpha, \{\zeta_i\}_{i \in [\ell]}) \\ c \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_\mathsf{Z}^\dagger, \tau) \\ \sigma := (\mu, \eta, \{\sigma_i, X_i, Y_i\}_{i=1}^{\ell}, c) \end{array} \right\}.$$

Replacing x_i and y_i with $r \cdot x_i$ and $s \cdot y_i$ respectively for some $r, s \leftarrow \mathbb{Z}_q^*$, we obtain a distribution $\mathcal{D}' = \mathcal{D}$ where

$$\mathcal{D}' := \left\{ \sigma : \begin{array}{l} r, s \leftarrow \mathbb{Z}_q^* \\ x_i, y_i \leftarrow \mathbb{Z}_q^*,\ X_i := G_1^{x_i},\ Y_i := X_i^{y_i},\ \forall i \in [\ell] \\ \mu \leftarrow \mathsf{EQS.Sign}(\mathsf{sk}_\mathsf{S}^\dagger, (X_1, \ldots, X_\ell)^r) \\ \eta \leftarrow \mathsf{EQS.Sign}(\mathsf{sk}_\mathsf{S}^\dagger, (Y_1, \ldots, Y_\ell)^{r \cdot s}) \\ \sigma_i \leftarrow \mathsf{H}(i \| m_i')^{s \cdot y_i},\ \forall i \in [\ell] \\ \zeta_i := \begin{cases} s \cdot y_i & i \in \alpha \\ 0 & \text{otherwise} \end{cases},\ \forall i \in [\ell] \\ \tau := (\alpha, \{\zeta_i\}_{i \in [\ell]}) \\ c \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_\mathsf{Z}^\dagger, \tau) \\ \sigma := (\mu, \eta, \{\sigma_i, X_i^r, Y_i^{r \cdot s}\}_{i=1}^{\ell}, c) \end{array} \right\}.$$

By the perfect adaption of EQS, the distribution of $\mathsf{EQS.Sign}(\mathsf{sk}_\mathsf{S}^\dagger, (X_1, \ldots, X_\ell)^r)$ and $\mathsf{EQS.Sign}(\mathsf{sk}_\mathsf{S}^\dagger, (Y_1, \ldots, Y_\ell)^{r \cdot s})$ is identical to that of $\mathsf{ChgRep}(\mathsf{pk}_\mathsf{S}^\dagger, (X_1, \ldots, X_\ell), \mu', r)$ and $\mathsf{ChgRep}(\mathsf{pk}_\mathsf{S}^\dagger, (Y_1, \ldots, Y_\ell), \eta', r \cdot s)$, where $\mu' \leftarrow$

$\mathsf{EQS.Sign}(\mathsf{sk}_{\mathsf{S}}^{\dagger}, (X_1, \ldots, X_\ell))$ and $\eta' \leftarrow \mathsf{EQS.Sign}(\mathsf{sk}_{\mathsf{S}}^{\dagger}, (Y_1, \ldots, Y_\ell))$. Therefore, we obtain a distribution $\mathcal{D}'' = \mathcal{D}'$ with

$$\mathcal{D}'' := \left\{ \sigma : \begin{array}{l} r, s \leftarrow \mathbb{Z}_q^* \\ x_i, y_i \leftarrow \mathbb{Z}_q^*,\ X_i := G_1^{x_i},\ Y_i := X_i^{y_i},\ \forall i \in [\ell] \\ \mu' \leftarrow \mathsf{EQS.Sign}(\mathsf{sk}_{\mathsf{S}}^{\dagger}, (X_1, \ldots, X_\ell)) \\ \eta' \leftarrow \mathsf{EQS.Sign}(\mathsf{sk}_{\mathsf{S}}^{\dagger}, (Y_1, \ldots, Y_\ell)) \\ \mu \leftarrow \mathsf{ChgRep}(\mathsf{pk}_{\mathsf{S}}^{\dagger}, (X_1, \ldots, X_\ell), \mu', r) \\ \eta \leftarrow \mathsf{ChgRep}(\mathsf{pk}_{\mathsf{S}}^{\dagger}, (Y_1, \ldots, Y_\ell), \eta', r \cdot s) \\ \sigma_i \leftarrow \mathsf{H}(i \| m_i')^{s \cdot y_i},\ \forall i \in [\ell] \\ \zeta_i := \begin{cases} s \cdot y_i & i \in \alpha \\ 0 & \text{otherwise} \end{cases},\ \forall i \in [\ell] \\ \tau := (\alpha, \{\zeta_i\}_{i \in [\ell]}) \\ c \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{Z}}^{\dagger}, \tau) \\ \sigma := (\mu, \eta, \{\sigma_i, X_i^r, Y_i^{r \cdot s}\}_{i=1}^{\ell}, c) \end{array} \right\}.$$

Note that in Hyb_j, the signature σ' is drawn exactly from $\mathcal{D}''$. Therefore we can conclude that Hyb_{j-1} and Hyb_j are functionally equivalent. □

5.4 Weak Unlinkability

Proof (Weak Unlinkability). To show that the experiments $\mathsf{wExpUnlink}_{\Pi,\mathcal{A}}^{b}(1^\lambda)$, where $b \in \{0,1\}$, are computationally indistinguishable in the view of the adversary we define the following sequence of hybrids.

Hyb_0^b : Defined as $\mathsf{wExpUnlink}_{\Pi,\mathcal{A}}^{b}(1^\lambda)$.

Hyb_1^b : Defined as Hyb_0^b, except that an additional list $\tilde{L}$ is initialized empty at the beginning of the experiment. Then, when a ciphertext $c \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{Z}}^{\dagger}, \tau)$ is generated in the subroutine Sign of the oracle $\mathsf{Sign}\mathcal{O}$, a new entry $(c, \tau, \{x_i, y_i\}_{i \in [\ell]})$ is added to $\tilde{L}$. The $\mathsf{wLoRSanit}$ oracle runs the following modified version of the subroutine $\tilde{\mathsf{San}}$: On input a certain $\tilde{c}$, it first checks whether there is an entry $(\tilde{c}, \tilde{\tau}, \{\tilde{x}_i, \tilde{y}_i\}_{i \in [\ell]})$ in $\tilde{L}$ and, if so, proceeds by setting $\tau = \tilde{\tau}$. Otherwise the algorithm aborts.

Hyb_2^b : Defined as Hyb_1^b, except that the ciphertext c is computed as $c \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{Z}}^{\dagger}, (\alpha, 0^\ell))$ in the $\mathsf{Sign}\mathcal{O}$ oracle. The suboutine $\tilde{\mathsf{San}}$ also computes $c' \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{Z}}^{\dagger}, (\alpha, 0^\ell))$.

Hyb_3^b : Defined as Hyb_2^b, except that the subroutine $\tilde{\mathsf{San}}$ is modified to compute the signatures μ' and η' as $\mu' \leftarrow \mathsf{EQS.Sign}(\mathsf{sk}_{\mathsf{S}}^{\dagger}, (X_1', \ldots, X_\ell'))$ and $\eta' \leftarrow \mathsf{EQS.Sign}(\mathsf{sk}_{\mathsf{S}}^{\dagger}, (Y_1', \ldots, Y_\ell'))$, respectively.

Hyb_4^b : Defined as Hyb_3^b, except that the subroutine $\tilde{\mathsf{San}}$ samples a fresh tuple $(Z_1, \ldots, Z_\ell) \leftarrow \mathbb{G}_1^\ell$ is sampled and $(X_1', \ldots, X_\ell') := (Z_1, \ldots, Z_\ell)$ and $(Y_1', \ldots, Y_\ell') := (Z_1^{\tilde{y}_1}, \ldots, Z_\ell^{\tilde{y}_\ell})^s$.

Hyb_5^b : Defined as Hyb_4^b, except that the subroutine $\tilde{\mathsf{San}}$ samples a tuple $(w_1, \ldots, w_\ell) \leftarrow \mathbb{Z}_q^\ell$ and computes σ_i' as $\mathsf{H}(m_i')^{w_i}$ and $(Y_1', \ldots, Y_\ell')$ as $(Z_1^{w_1}, \ldots, Z_\ell^{w_\ell})$.

Observe that in the experiment Hyb_5^b the output of the $\mathsf{wLoRSanit}$ oracle in Hyb_5^b is completely decorrelated from the random coin b. It follows that for all PPT adversaries we have that

$$\left| \mathsf{Pr}\left[\mathsf{Hyb}_5^0 = 1\right] - \mathsf{Pr}\left[\mathsf{Hyb}_5^1 = 1\right] \right| \leq \mathsf{negl}(\lambda).$$

We now proceed by showing the indistinguishability of each pair of hybrids.

Lemma 4. *Suppose* PKE *is correct. Then for all PPT* $\mathcal{A}$ *and* $b \in \{0,1\}$ *it holds that*

$$\left| \mathsf{Pr}\left[\mathsf{Hyb}_0^b = 1\right] - \mathsf{Pr}\left[\mathsf{Hyb}_1^b = 1\right] \right| \leq \mathsf{negl}(\lambda).$$

Proof (of Lemma 4). The experiments differ in the fact that in Hyb_1^b the $\mathsf{LoRSanit}$ oracle aborts when queried on some $\tilde{c}$ such that no entry $(\tilde{c}, \cdot)$ is present in $\tilde{L}$. This implies that $\tilde{c}$ was not produced by the signing oracle $\mathsf{Sign}\mathcal{O}$, therefore also Hyb_0^b aborts on the same input. The indistinguishability follows by the correctness of the encryption scheme. □

Lemma 5. *Suppose* PKE *is IND-CCA secure. Then for all PPT* $\mathcal{A}$ *and* $b \in \{0,1\}$,

$$\left|\Pr\left[\mathsf{Hyb}_1^b = 1\right] - \Pr\left[\mathsf{Hyb}_2^b = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

Proof (of Lemma 5). The lemma follows by a simple reduction to the (multiple-message) CCA-security of the encryption scheme. On input a public key $\mathsf{pk}_{\mathsf{Z}}^{\dagger}$, the reduction computes c by querying the challenger on $(\tau, (\alpha, 0^{\ell}))$ and plugging in the corresponding ciphertext c^*. The ciphertext c' is computed analogously. The oracle $\mathsf{San}\mathcal{O}$ is simulated by passing the input ciphertexts $\tilde{c}$ to the decryption oracle and setting τ to be the corresponding output. It is easy to show that in the one case the reduction perfectly simulates the distributions of Hyb_1^b and in the other case it is identical to Hyb_2^b. By the CCA-security of the encryption scheme the claim follows. □

Lemma 6. *If* EQS *perfectly adapts signatures, then for all PPT* $\mathcal{A}$ *and* $b \in \{0,1\}$,

$$\left|\Pr\left[\mathsf{Hyb}_2^b = 1\right] - \Pr\left[\mathsf{Hyb}_3^b = 1\right]\right| = 0.$$

Proof (of Lemma 6). Trivial. □

Lemma 7. *Let* $q > 2^{\lambda}$. *For all generic group adversary* $\mathcal{A}$ *and* $b \in \{0,1\}$ *it holds that*

$$\left|\Pr\left[\mathsf{Hyb}_3^b = 1\right] - \Pr\left[\mathsf{Hyb}_4^b = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

Proof (of Lemma 7). First observe that in Hyb_4^b (with a slight notation abuse)

$$(Y'_1, \ldots, Y'_\ell) = (Z_1^{\tilde{y}_1}, \ldots, Z_\ell^{\tilde{y}_\ell})^s = (X'_1, \ldots, X'_\ell)^{s(\tilde{y}_1, \ldots, \tilde{y}_\ell)}$$

whereas in Hyb_3^b

$$(Y'_1, \ldots, Y'_\ell) = (Y_1, \ldots, Y_\ell)^{r \cdot s} = (X_1^{\tilde{y}_1}, \ldots, X_\ell^{\tilde{y}_\ell})^{r \cdot s} = (X'_1, \ldots, X'_\ell)^{s(\tilde{y}_1, \ldots, \tilde{y}_\ell)}.$$

Therefore if the tuple $(X'_1, \ldots, X'_\ell)$ has the same distribution in both experiments the indistinguishability follows. It remains to show that

$$\begin{pmatrix} G_1, & X_1, \ldots, X_\ell \\ & Z_1, \ldots, Z_\ell \end{pmatrix} \approx \begin{pmatrix} G_1, & X_1, \ldots, X_\ell \\ & X_1^r, \ldots, X_\ell^r \end{pmatrix}$$

over the random choice of $(Z_1, \ldots, Z_\ell, r)$. It is easy to show that the two distributions are indistinguishable by the hardness of the decisional Diffie-Hellman problem [DH76]. For completeness, and as a warm-up for the proof of the next lemma, we show that this holds in the generic group model. For ease of exposition, we denote symbolically $X_k := G_1^{x_k}$ and $Z_k := G_1^{z_k}$ for all $k \in [\ell]$. For the left distribution we can rewrite all equations that the adversary learns as symbolic degree 1 polynomials

$$\mathfrak{x}_1 x_1 + \ldots + \mathfrak{x}_\ell x_\ell + \mathfrak{z}_1 z_1 + \ldots + \mathfrak{z}_\ell z_\ell + \mathfrak{c} = 0$$

for some coefficients $(\mathfrak{x}_1, \ldots, \mathfrak{x}_\ell, \mathfrak{z}_1, \ldots, \mathfrak{z}_\ell, \mathfrak{c})$. For the right distributions we have

$$\mathfrak{x}_1 x_1 + \ldots + \mathfrak{x}_\ell x_\ell + \mathfrak{z}_1 x_1 r + \ldots + \mathfrak{z}_\ell x_\ell r + \mathfrak{c} = 0.$$

Since $(x_1, \ldots, x_\ell)$ and $(z_1, \ldots, z_\ell)$ are uniformly chosen, by Lemma 2 all coefficients in the left distribution must be 0 with all but negligible probability. The same holds for the right distribution, since r is uniformly sampled. It follows that a generic adversary cannot learn any non-trivial relation when it is given either the left or the right distribution. Therefore the left and right distributions look identical. □

Lemma 8. *Let* $q > 2^{\lambda}$. *For all generic group adversary* $\mathcal{A}$ *and* $b \in \{0,1\}$ *it holds that*

$$\left|\Pr\left[\mathsf{Hyb}_4^b = 1\right] - \Pr\left[\mathsf{Hyb}_5^b = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

Proof (of Lemma 8). The two experiments differ in the way the variables $(Y'_1, \ldots, Y'_\ell)$ and $(\sigma'_1, \ldots, \sigma'_\ell)$ are computed. Suppose that $\mathsf{H}(i\|m_i)$ and $\mathsf{H}(i\|m'_i)$ are programed to $G_2^{t_i^0}$ and $G_2^{t_i^1}$ respectively, where $(t_1^0, t_1^1, \ldots, t_\ell^0, t_\ell^1)$ is a randomly sampled vector in $\mathbb{Z}_q^{2\ell}$. The indistinguishability of the two hybrids reduces to arguing about the proximity of the following distributions

$$\begin{pmatrix} & G_1^{y_1}, & \ldots, G_1^{y_\ell}, \\ G_1, & G_1^{z_1}, & \ldots, G_1^{z_\ell}, \\ & G_1^{sy_1z_1}, & \ldots, G_1^{sy_\ell z_\ell}, \\ & G_2^{t_1^0 y_1}, & \ldots, G_2^{t_\ell^0 y_\ell}, \\ & G_2^{st_1^1 y_1}, & \ldots, G_2^{st_\ell^1 y_\ell}, \\ G_2, & G_2^{t_1^0}, & \ldots, G_2^{t_\ell^0}, \\ & G_2^{t_1^1}, & \ldots, G_2^{t_\ell^1} \end{pmatrix} \approx \begin{pmatrix} & G_1^{y_1}, & \ldots, G_1^{y_\ell}, \\ G_1, & G_1^{z_1}, & \ldots, G_1^{z_\ell}, \\ & G_1^{w_1z_1}, & \ldots, G_1^{w_\ell z_\ell}, \\ & G_2^{t_1^0 y_1}, & \ldots, G_2^{t_\ell^0 y_\ell}, \\ & G_2^{t_1^1 w_1}, & \ldots, G_2^{t_\ell^1 w_\ell}, \\ G_2, & G_2^{t_1^0}, & \ldots, G_2^{t_\ell^0}, \\ & G_2^{t_1^1}, & \ldots, G_2^{t_\ell^1} \end{pmatrix}$$

where the LHS corresponds to the distributions in Hyb_4^b and the RHS corresponds to the distributions in Hyb_5^b. Note that all non-trivial relations that the generic attacker can learn are restricted to certain polynomials of degree at most 2. For illustration, we can symbolically write the relations obtained from the RHS as

$$\begin{aligned} &\sum_{i\in[\ell],j\in[\ell]} \mathfrak{a}_{i,j}\left(t_i^1 w_i \cdot w_j z_j\right) + \sum_{i\in[\ell],j\in[\ell]} \mathfrak{b}_{i,j}\left(t_i^1 w_i \cdot y_j\right) \\ &+ \sum_{i\in[\ell],j\in[\ell]} \mathfrak{c}_{i,j}\left(t_i^0 y_i \cdot w_j z_j\right) + \sum_{i\in[\ell],j\in[\ell]} \mathfrak{d}_{i,j}\left(t_i^0 y_i \cdot y_j\right) \\ &+ \sum_{i\in[\ell],j\in[\ell],b\in\{0,1\}} \mathfrak{e}_{i,j}^b\left(t_i^b \cdot w_j z_j\right) + \sum_{i\in[\ell],j\in[\ell],b\in\{0,1\}} \mathfrak{f}_{i,j}^b\left(t_i^b \cdot y_j\right) \\ &+ \sum_{i\in[\ell],j\in[\ell]} \mathfrak{g}_{i,j}\left(t_i^1 w_i \cdot z_j\right) + \sum_{i\in[\ell]} \mathfrak{h}_i\left(t_i^1 w_i\right) \\ &+ \sum_{i\in[\ell],j\in[\ell]} \mathfrak{i}_{i,j}\left(t_1^0 y_i \cdot z_j\right) + \sum_{i\in[\ell]} \mathfrak{j}_i\left(t_1^0 y_i\right) \\ &+ \sum_{i\in[\ell],j\in[\ell],b\in\{0,1\}} \mathfrak{k}_{i,j}^b\left(t_i^b \cdot z_j\right) + \sum_{i\in[\ell],b\in\{0,1\}} \mathfrak{l}_i^b\left(t_i^b\right) \\ &+ \sum_{i\in[\ell]} \mathfrak{m}_i(y_i) + \sum_{i\in[\ell]} \mathfrak{n}_i(z_i) \\ &+ \sum_{i\in[\ell]} \mathfrak{o}_i(w_i z_i) + \mathfrak{p} \\ &= 0. \end{aligned}$$

whereas for the LHS the equation is identical except that all occurrences of w_i and w_j are replaced with $y_i \cdot s$ and $y_j \cdot s$, respectively. Since all variables are uniformly distributed, by Lemma 2 we have that the coefficient of each unique monomial must be 0 with all but negligible probability. It is left to argue that each non-trivial relation obtained on the RHS imply also a corresponding non-trivial relation on the LHS, and viceversa. By inspection we isolate the pairs

$$\left(\sum_{i\in[\ell],j\in[\ell],b\in\{0,1\}} \mathfrak{e}_{i,j}^b\left(t_i^b \cdot w_j z_j\right), \sum_{i\in[\ell],j\in[\ell]} \mathfrak{g}_{i,j}\left(t_i^1 w_i \cdot z_j\right)\right)$$

and

$$\left(\sum_{i\in[\ell],j\in[\ell],b\in\{0,1\}} \mathfrak{f}_{i,j}^b\left(t_i^b \cdot y_j\right), \sum_{i\in[\ell]} \mathfrak{j}_i(t_i^0 y_i)\right)$$

that have potentially common monomials. For the latter case it is enough to observe that the monomials are identical for both the LHS and the RHS distributions as they are independent of w_i and s for all $i \in [\ell]$. Therefore if

$$\sum_{i\in[\ell],j\in[\ell],b\in\{0,1\}} \mathfrak{f}_{i,j}(t_i^b \cdot y_j) + \sum_{i\in[\ell]} \mathfrak{j}_i(t_i^0 y_i) = 0$$

in the RHS then so it does in the LHS, and vice versa. For the former case we have that collisions occur only when $i = j$ and $b = 1$, as otherwise the monomials are distinct and therefore any non trivial set of coefficients will not cancel out (with very high probability). Setting $i = j$ and $b = 1$, for the RHS we have the following constraint

$$\sum_{i\in[\ell]} \mathfrak{e}_{i,i}^1 \left(t_i^1 w_i z_i\right) + \sum_{i\in[\ell]} \mathfrak{g}_{i,i} \left(t_i^1 w_i z_i\right) = 0$$

which implies that, with overwhelming probability, for all $i \in [\ell]$ it holds that $\mathfrak{e}_{i,i}^1 = -\mathfrak{g}_{i,i}$. Applying this constraint to the LHS we obtain a corresponding non-trivial relation

$$\sum_{i\in[\ell]} \mathfrak{e}_{i,i}^1 \left(s t_i^1 y_i z_i\right) + \sum_{i\in[\ell]} \mathfrak{g}_{i,i} \left(s t_i^1 y_i z_i\right) = 0.$$

The reverse direction holds with an identical argument. Since there is a bijection between the non-trivial relations on the LHS and those on the RHS, we can conclude that the view of $\mathcal{A}$ in the two cases are indistinguishable. □

Acknowledgments

This work is a result of the collaborative research project PROMISE (16KIS0763) by the German Federal Ministry of Education and Research (BMBF). FAU authors were also supported by the German research foundation (DFG) through the collaborative research center 1223, and by the state of Bavaria at the Nuremberg Campus of Technology (NCT). NCT is a research cooperation between the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and the Technische Hochschule Nürnberg Georg Simon Ohm (THN).

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.

BB04. Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238, 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.

BCD+17. Michael Till Beck, Jan Camenisch, David Derler, Stephan Krenn, Henrich C. Pöhls, Kai Samelin, and Daniel Slamanig. Practical strongly invisible and strongly accountable sanitizable signatures. In Josef Pieprzyk and Suriadi Suriadi, editors, *ACISP 17: 22nd Australasian Conference on Information Security and Privacy, Part I*, volume 10342 of *Lecture Notes in Computer Science*, pages 437–452, Auckland, New Zealand, July 3–5, 2017. 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.

BL17. Xavier Bultel and Pascal Lafourcade. Unlinkable and strongly accountable sanitizable signatures from verifiable ring signatures. In *Cryptology and Network Security*. Springer International Publishing, 2017.

BLS01. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532, Gold Coast, Australia, December 9–13, 2001. Springer, Heidelberg, Germany.

BPS13. Christina Brzuska, Henrich C. Pöhls, and Kai Samelin. Non-interactive public accountability for sanitizable signatures. In Sabrina De Capitani di Vimercati and Chris Mitchell, editors, *Public Key Infrastructures, Services and Applications*, pages 178–193, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

CDK+17. Jan Camenisch, David Derler, Stephan Krenn, Henrich C. Pöhls, Kai Samelin, and Daniel Slamanig. Chameleon-hashes with ephemeral trapdoors - and applications to invisible sanitizable signatures. In Serge Fehr, editor, *PKC 2017: 20th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 10175 of *Lecture Notes in Computer Science*, pages 152–182, Amsterdam, The Netherlands, March 28–31, 2017. 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.

DH76. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

DPSS16. David Derler, Henrich C. Pöhls, Kai Samelin, and Daniel Slamanig. A general framework for redactable signatures and new constructions. In Soonhak Kwon and Aaram Yun, editors, *ICISC 15: 18th International Conference on Information Security and Cryptology*, volume 9558 of *Lecture Notes in Computer Science*, pages 3–19, Seoul, Korea, November 25–27, 2016. Springer, Heidelberg, Germany.

ElG84. Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18, Santa Barbara, CA, USA, August 19–23, 1984. Springer, Heidelberg, Germany.

FH18. Marc Fischlin and Patrick Harasser. Invisible sanitizable signatures and public-key encryption are equivalent. Cryptology ePrint Archive, Report 2018/337, 2018. https://eprint.iacr.org/2018/337.

FHS18. Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. 02 2018.

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.

FO13. Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *Journal of Cryptology*, 26(1):80–101, January 2013.

FS07. Eiichiro Fujisaki and Koutarou Suzuki. Traceable ring signature. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007: 10th International Conference on Theory and Practice of Public Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 181–200, Beijing, China, April 16–20, 2007. Springer, Heidelberg, Germany.

FZ12. Matthew Franklin and Haibin Zhang. A framework for unique ring signatures. Cryptology ePrint Archive, Report 2012/577, 2012. http://eprint.iacr.org/2012/577.

HS14. Christian Hanser and Daniel Slamanig. Structure-preserving signatures on equivalence classes and their application to anonymous credentials. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 491–511, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. 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.

JWL12. Rob Johnson, Leif Walsh, and Michael Lamb. Homomorphic signatures for digital photographs. In George Danezis, editor, *FC 2011: 15th International Conference on Financial Cryptography and Data Security*, volume 7035 of *Lecture*

Notes in Computer Science, pages 141–157, Gros Islet, St. Lucia, February 28 – March 4, 2012. Springer, Heidelberg, Germany.

KSS16. Stephan Krenn, Kai Samelin, and Dieter Sommer. Stronger security for sanitizable signatures. In Joaquin Garcia-Alfaro, Guillermo Navarro-Arribas, Alessandro Aldini, Fabio Martinelli, and Neeraj Suri, editors, *Data Privacy Management, and Security Assurance*, pages 100–117, Cham, 2016. Springer International Publishing.

LWW04. Joseph K. Liu, Victor K. Wei, and Duncan S. Wong. Linkable spontaneous anonymous group signature for ad hoc groups (extended abstract). In Huaxiong Wang, Josef Pieprzyk, and Vijay Varadharajan, editors, *ACISP 04: 9th Australasian Conference on Information Security and Privacy*, volume 3108 of *Lecture Notes in Computer Science*, pages 325–335, Sydney, NSW, Australia, July 13–15, 2004. Springer, Heidelberg, Germany.

LX03. Jiqiang Lu and Wang Xinmei. Verifiable ring signature. 2003.

LZCS16. Russell W. F. Lai, Tao Zhang, Sherman S. M. Chow, and Dominique Schröder. Efficient sanitizable signatures without random oracles. In Ioannis G. Askoxylakis, Sotiris Ioannidis, Sokratis K. Katsikas, and Catherine A. Meadows, editors, *ESORICS 2016: 21st European Symposium on Research in Computer Security, Part I*, volume 9878 of *Lecture Notes in Computer Science*, pages 363–380, Heraklion, Greece, September 26–30, 2016. Springer, Heidelberg, Germany.

MHI08. Kunihiko Miyazaki, Goichiro Hanaoka, and Hideki Imai. Invisibly sanitizable digital signature scheme. In *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, 2008.

OTO99. Takeshi Okamoto, Mitsuru Tada, and Eiji Okamoto. Extended proxy signatures for smart cards. In Masahiro Mambo and Yuliang Zheng, editors, *ISW'99: 2nd International Workshop on Information Security*, volume 1729 of *Lecture Notes in Computer Science*, pages 247–258, Kuala Lumpur, Malaysia, November 1999. Springer, Heidelberg, Germany.

PS14. Henrich Christopher Pöhls and Kai Samelin. On updatable redactable signatures. In Ioana Boureanu, Philippe Owesarski, and Serge Vaudenay, editors, *ACNS 14: 12th International Conference on Applied Cryptography and Network Security*, volume 8479 of *Lecture Notes in Computer Science*, pages 457–475, Lausanne, Switzerland, June 10–13, 2014. Springer, Heidelberg, Germany.

Shi06. Kyung-Ah Shim. An identity-based proxy signature scheme from pairings. In Peng Ning, Sihan Qing, and Ninghui Li, editors, *ICICS 06: 8th International Conference on Information and Communication Security*, volume 4307 of *Lecture Notes in Computer Science*, pages 60–71, Raleigh, NC, USA, December 4–7, 2006. Springer, Heidelberg, Germany.

Sho97. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266, Konstanz, Germany, May 11–15, 1997. Springer, Heidelberg, Germany.

WP03. Huaxiong Wang and Josef Pieprzyk. Efficient one-time proxy signatures. In Chi-Sung Laih, editor, *Advances in Cryptology – ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 507–522, Taipei, Taiwan, November 30 – December 4, 2003. Springer, Heidelberg, Germany.

XY04. Shouhuai Xu and Moti Yung. Accountable ring signatures: A smart card approach. In *Smart Card Research and Advanced Applications VI*, pages 271–286. Springer, 2004.

A More Preliminaries

A.1 Public-Key Encryption

Here we recall the notion of public key encryption together with the standard definition of CCA-security.

Definition 14 (PKE). *A public-key encryption scheme (PKE) is a tuple of* PPT *algorithms* $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *defined as follows:*

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$*: The probabilistic key generation algorithm takes as input the security parameter* 1^λ *and returns a key pair* $(\mathsf{pk}, \mathsf{sk})$.

$c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$*: The probabilistic encryption algorithm takes as input the public key* pk *and a message* m *and returns a ciphertext* c.

$m \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$*: The decryption algorithm takes as input the secret key* sk *and a ciphertext* c *and returns a message* m.

Security of a public key encryption scheme is formalized as follows.

Definition 15 (CCA Security). *A* PKE *scheme is said to have indistinguishability against chosen-ciphertext attacks (IND-CCA) if for all PPT adversaries* $\mathcal{A}$*,*

$$\left|\mathsf{Pr}\left[\mathsf{ExpCCA}^0_{\mathcal{A},\mathsf{PKE}}(1^\lambda) = 1\right] - \mathsf{Pr}\left[\mathsf{ExpCCA}^1_{\mathcal{A},\mathsf{PKE}}(1^\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{ExpCCA}^b_{\mathcal{A},\mathsf{PKE}}$ *is defined in Figure 6.*

$\mathsf{ExpCCA}^{b}_{\mathcal{A},\mathsf{PKE}}(1^\lambda)$

$\mathcal{Q} := \emptyset$

$(\mathsf{pk}^\dagger, \mathsf{sk}^\dagger) \leftarrow \mathsf{KGen}_{\mathsf{PKE}}(1^\lambda)$

$b' \leftarrow \mathcal{A}^{\mathsf{Enc}\mathcal{O}_b, \mathsf{Dec}\mathcal{O}}(\mathsf{pk}^\dagger)$

return b'

$\mathsf{Enc}\mathcal{O}_b(m_0, m_1)$

if $|m_0| \neq |m_1|$ **then**
  return $\perp$
endif

$c \leftarrow \mathsf{Enc}(\mathsf{pk}^\dagger, m_b)$

$\mathcal{Q} := \mathcal{Q} \cup \{c\}$

return c

$\mathsf{Dec}\mathcal{O}(c)$

if $c \in \mathcal{Q}$ **then**
  return $\perp$
endif

$m \leftarrow \mathsf{Dec}(\mathsf{sk}^\dagger, c)$

return m

Fig. 6: IND-CCA Experiment

$\mathsf{Sign}\mathcal{O}(L, i, m)$

return $\mathsf{Sign}(\mathsf{sk}^\dagger_i, L, m)$

$\mathsf{Prove}\mathcal{O}(L, m, \sigma, i)$

if $\begin{cases} \mathsf{pk}^\dagger_i \in L \\ (m, \sigma) \notin \mathcal{Q} \end{cases}$ **then**
  return $\mathsf{Prove}(\mathsf{sk}^\dagger_i, L, m, \sigma)$
else
  return $\perp$
endif

$\mathsf{LoRSign}\mathcal{O}_b(L, m)$

if $\{\mathsf{pk}^\dagger_0, \mathsf{pk}^\dagger_1\} \subseteq L$ **then**
  $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}^\dagger_b, L, m)$
  $\mathcal{Q} := \mathcal{Q} \cup \{(m, \sigma)\}$
  return σ
else
  return $\perp$
endif

Fig. 7: Oracles for Verifiable Ring Signatures

A.2 Properties of Verifiable Ring Signatures

We define slightly simplified and strengthened security definitions for VRS. We note that the existing construction in [BL17] satisfies all properties defined below.

Strong Unforgeability. A VRS is strongly unforgeable when no adversary that has access to the signature oracle and the proof oracle is able to forge a fresh message-signature pair (m, σ), such that the corresponding ring contains public keys of honest users only.

Definition 16 (Strong Unforgeability). *A VRS scheme* VRS *is strongly unforgeable if for all* $n \in \mathsf{poly}(\lambda)$ *and for all PPT adversary* $\mathcal{A}$*,*

$$\Pr\left[\mathsf{ExpUnf}^{n}_{\mathsf{VRS},\mathcal{A}}(1^\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$$

where $\mathsf{ExpUnf}^{n}_{\mathsf{VRS},\mathcal{A}}(1^\lambda)$ *is defined in Figure 8.*

Anonymity. A VRS is anonymous when no adversary with access to the signature, proof, and "left-or-right signature" oracles is able to link a signature to the public key of its signer.

Definition 17 (Anonymity). *A VRS scheme* VRS *is anonymous if for all PPT* $\mathcal{A}$*,*

$$|\Pr\left[\mathsf{ExpAnon}^{0}_{\mathsf{VRS},\mathcal{A}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{ExpAnon}^{1}_{\mathsf{VRS},\mathcal{A}}(1^\lambda) = 1\right]| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{ExpAnon}^{b}_{\mathsf{VRS},\mathcal{A}}(1^\lambda)$ *is defined in Figure 8.*

$\mathsf{ExpUnf}^{n}_{\mathsf{VRS},\mathcal{A}}(1^\lambda)$

$\mathcal{Q} := \epsilon,\ \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$\forall i \in [n],\ (\mathsf{pk}_i^\dagger, \mathsf{sk}_i^\dagger) \leftarrow \mathsf{KGen}(\mathsf{pp})$
$\mathbb{O} := \{\mathsf{Sign}\mathcal{O}, \mathsf{Prove}\mathcal{O}\}$
$(L^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathbb{O}}(\mathsf{pp}, \{\mathsf{pk}_i^\dagger\}_{i\in[n]})$
$b_0 := \mathsf{Verify}(L^*, m^*, \sigma^*)$
$b_1 := L^* \subseteq \{\mathsf{pk}_i\}_{i\in[n]}$
$b_2 := (m^*, \sigma^*) \notin \mathcal{Q}$
return $b_0 \wedge b_1 \wedge b_2$

$\mathsf{ExpAnon}^{b}_{\mathsf{VRS},\mathcal{A}}(1^\lambda)$

$\mathcal{Q} := \epsilon,\ \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$(\mathsf{pk}_0^\dagger, \mathsf{sk}_0^\dagger) \leftarrow \mathsf{KGen}(\mathsf{pp})$
$(\mathsf{pk}_1^\dagger, \mathsf{sk}_1^\dagger) \leftarrow \mathsf{KGen}(\mathsf{pp})$
$\mathbb{O} := \{\mathsf{Sign}\mathcal{O}, \mathsf{Prove}\mathcal{O}, \mathsf{LoRSign}\mathcal{O}_b\}$
$b' \leftarrow \mathcal{A}^{\mathbb{O}}(\mathsf{pp}, \mathsf{pk}_0^\dagger, \mathsf{pk}_1^\dagger)$
return b'

$\mathsf{ExpAcc}^{n}_{\mathsf{VRS},\mathcal{A}}(1^\lambda)$

$\mathcal{Q} := \epsilon,\ \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$\forall i \in [n],\ (\mathsf{pk}_i^\dagger, \mathsf{sk}_i^\dagger) \leftarrow \mathsf{KGen}(\mathsf{pp})$
$\mathbb{O} := \{\mathsf{Sign}\mathcal{O}, \mathsf{Prove}\mathcal{O}\}$
$(L^*, m^*, \sigma^*, \mathsf{pk}^*, \pi^*) \leftarrow \mathcal{A}^{\mathbb{O}}(\mathsf{pp}, \{\mathsf{pk}_i^\dagger\}_{i\in[n]})$
$b_0 := \mathsf{Verify}(L^*, m^*, \sigma^*)$
$b_1 := (\mathsf{Judge}(L^*, m^*, \sigma^*, \mathsf{pk}^*, \pi^*) = 0)$
$b_2 := L^* \subseteq \{\mathsf{pk}_i\}_{i\in[n]} \cup \{\mathsf{pk}^*\}$
$b_3 := (m^*, \sigma^*) \notin \mathcal{Q}$
return $b_0 \wedge b_1 \wedge b_2 \wedge b_3$

$\mathsf{ExpNonSeiz}_{\mathsf{VRS},\mathcal{A}}(1^\lambda)$

$\mathcal{Q} := \epsilon,\ \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$(\mathsf{pk}^\dagger, \mathsf{sk}^\dagger) \leftarrow \mathsf{KGen}(\mathsf{pp})$
$\mathbb{O} := \{\mathsf{Sign}\mathcal{O}, \mathsf{Prove}\mathcal{O}\}$
$(L^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathbb{O}}(\mathsf{pp}, \mathsf{pk}^\dagger)$
$\pi^* \leftarrow \mathsf{Prove}(L^*, m^*, \mathsf{pk}^\dagger, \mathsf{sk}^\dagger)$
$b_0 := \mathsf{Verify}(L^*, m^*, \sigma^*)$
$b_1 := (\mathsf{Judge}(L^*, m^*, \sigma^*, \mathsf{pk}^\dagger, \pi^*) \neq 0)$
$b_2 := (m^*, \sigma^*) \notin \mathcal{Q}$
return $b_0 \wedge b_1 \wedge b_2$

Fig. 8: Experiments for Verifiable Ring Signatures.

Strong Accountability. A VRS is strongly accountable when no adversary that has access to the signature oracle and the proof oracle is able to forge a fresh message-signature pair (m, σ), together with a proof that it is not the signer of σ, such that the corresponding ring contains at most one public key of a non-honest user.

Definition 18 (Strong Accountability). *A VRS scheme* VRS *is strongly accountable if for all* $n \in \mathsf{poly}(\lambda)$ *and for all PPT adversary* $\mathcal{A}$*,*

$$\Pr\left[\mathsf{ExpAcc}^{n}_{\mathsf{VRS},\mathcal{A}}(1^\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$$

where $\mathsf{ExpAcc}^{n}_{\mathsf{VRS},\mathcal{A}}(1^\lambda)$ *is defined in Figure 8.*

It follows immediately that strong accountability implies strong unforgeability.

Strong Non-Seizability. A VRS is strongly non-seizable when no adversary that has access to the signature and the proof oracle is able to forge a fresh message-signature pair (m, σ), such that the proof algorithm ran by the honest user returns a proof that σ was computed by the honest user.

Definition 19 (Strong Non-Seizability). *A VRS scheme* VRS *is strongly non-seizable if for all PPT adversary* $\mathcal{A}$*,*

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

where $\mathsf{ExpNonSeiz}_{\mathsf{VRS},\mathcal{A}}(1^\lambda)$ *is defined in Figure 8.*

B Security Proofs for Construction II

B.1 Immutability

Proof (Immutability). Assume that there exists a polynomial time adversary $\mathcal{A}$ that breaks the immutability of Π_2. We show how to build an algorithm $\mathcal{B}$ that breaks the weak immutability of Π_1. The algorithm $\mathcal{B}$ receives $(\mathsf{pp}_{\mathsf{SS}}, \mathsf{spk}^{\ddagger}_{\mathsf{S}})$ as input, then it runs $\mathsf{pp}_{\mathsf{VRS}} \leftarrow \mathsf{VRS.Setup}(1^{\lambda})$, it sets $\mathsf{pp} := (\mathsf{pp}_{\mathsf{SS}}, \mathsf{pp}_{\mathsf{VRS}})$, it runs $(\mathsf{vpk}^{\dagger}_{\mathsf{S}}, \mathsf{vsk}^{\dagger}_{\mathsf{S}}) \leftarrow \mathsf{VRS.KGen}(\mathsf{pp}_{\mathsf{VRS}})$ and sets $\mathsf{pk}^{\dagger}_{\mathsf{S}} := (\mathsf{spk}^{\ddagger}_{\mathsf{S}}, \mathsf{vpk}^{\dagger}_{\mathsf{S}})$. It runs $(\mathsf{pk}^{*}_{\mathsf{Z}}, m^{*}, \sigma^{*}) \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{pk}^{\dagger}_{\mathsf{S}})$. While $\mathcal{A}$ is running, $\mathcal{B}$ simulates the oracles $\Pi_2.\mathsf{Sign}\mathcal{O}$ and $\Pi_2.\mathsf{Prove}\mathcal{O}$ as follows.

Answering Sign Oracle Queries. On input $(\mathsf{pk}_{\mathsf{Z}}, m, \alpha)$, it parses pk_{Z} as $(\mathsf{spk}_{\mathsf{Z}}, \mathsf{vpk}_{\mathsf{Z}})$, then it sets $\alpha' := 0 \| \alpha$ and $m' := \mathsf{pk}_{\mathsf{Z}} \| m$. $\mathcal{B}$ sends $(\mathsf{spk}_{\mathsf{Z}}, m', \alpha')$ to the sign oracle $\Pi_1.\mathsf{Sign}\mathcal{O}$ that answers σ_{SS}. It sets $t := \mathsf{pk}^{\dagger}_{\mathsf{S}} \| \mathsf{pk}_{\mathsf{Z}} \| m \| \sigma_{\mathsf{SS}}$ and runs $\sigma_{\mathsf{VRS}} \leftarrow \mathsf{VRS.Sign}(\mathsf{vsk}^{\dagger}_{\mathsf{S}}, \{\mathsf{vpk}^{\dagger}_{\mathsf{S}}, \mathsf{vpk}_{\mathsf{Z}}\}, t)$ and returns $\sigma = (\sigma_{\mathsf{SS}}, \sigma_{\mathsf{VRS}})$ to $\mathcal{A}$.

Answering Prove Oracle Queries. On input $(\mathsf{pk}_{\mathsf{Z}}, m, \sigma)$, it parses σ as $(\sigma_{\mathsf{SS}}, \sigma_{\mathsf{VRS}})$ and pk_{Z} as $(\mathsf{spk}_{\mathsf{Z}}, \mathsf{vpk}_{\mathsf{Z}})$, then it sets $t := \mathsf{pk}^{\dagger}_{\mathsf{S}} \| \mathsf{pk}_{\mathsf{Z}} \| m \| \sigma_{\mathsf{SS}}$. It runs $\pi \leftarrow \mathsf{VRS.Prove}(\mathsf{vsk}^{\dagger}_{\mathsf{S}}, \{\mathsf{vpk}^{\dagger}_{\mathsf{S}}, \mathsf{vpk}_{\mathsf{Z}}\}, t, \sigma_{\mathsf{VRS}})$ and returns π.

$\mathcal{B}$ parses σ^{*} as $(\sigma^{*}_{\mathsf{SS}}, \sigma^{*}_{\mathsf{VRS}})$ and $\mathsf{pk}^{*}_{\mathsf{Z}}$ as $(\mathsf{spk}^{*}_{\mathsf{Z}}, \mathsf{vpk}^{*}_{\mathsf{Z}})$. It returns $(\mathsf{spk}^{*}_{\mathsf{Z}}, \mathsf{pk}^{*}_{\mathsf{Z}} \| m^{*}, \sigma^{*}_{\mathsf{SS}})$. Parse L as

$$\{(\mathsf{pk}_{\mathsf{S},i}, \mathsf{pk}_{\mathsf{Z},i}, m_i, \alpha_i, \sigma_i)\}_{i=1}^{[|L|]}.$$

Assume that $\mathsf{ExpImmutability}_{\mathcal{A},\Pi_2}(1^{\lambda})$ returns 1, then the following holds:

- $\Pi_2.\mathsf{Verify}(\mathsf{pk}^{\dagger}_{\mathsf{S}}, \mathsf{pk}^{*}_{\mathsf{Z}}, m^{*}, \sigma^{*}) = 1$
- For all $i \in [|L|]$, $\mathsf{pk}^{*}_{\mathsf{Z}} \neq \mathsf{pk}_{\mathsf{Z},i} \vee m^{*} \notin \{\delta(m_i) \mid \delta \text{ with } \alpha_i(\delta) = 1\}$.

The first condition implies that $\Pi_1.\mathsf{Verify}(\mathsf{spk}^{\dagger}_{\mathsf{S}}, \mathsf{spk}^{*}_{\mathsf{Z}}, \mathsf{pk}^{*}_{\mathsf{Z}} \| m^{*}, \sigma^{*}_{\mathsf{SS}}) = 1$.

For the second condition, recall that given any δ_i, δ'_i is defined so that $\delta'_i(\mathsf{pk}_{\mathsf{Z},i} \| m_i) := \mathsf{pk}_{\mathsf{Z},i} \| \delta_i(m_i)$. Therefore if $\mathsf{pk}^{*}_{\mathsf{Z}} \neq \mathsf{pk}_{\mathsf{Z},i}$ or

$$m^{*} \notin \{\delta(m_i) \mid \delta \text{ with } \alpha_i(\delta) = 1\},$$

then

$$\mathsf{pk}^{*}_{\mathsf{Z}} \| m^{*} \notin \left\{\delta'(\mathsf{pk}_{\mathsf{Z},i} \| m_i) \mid \delta' \text{ with } \alpha'_i(\delta') = 1\right\}.$$

We therefore conclude that $\mathsf{ExpImmutability}_{\mathcal{B},\Pi_1}(1^{\lambda}) = 1$. To summarize, if $\mathsf{ExpImmutability}_{\mathcal{A},\Pi_2}(1^{\lambda})$ returns 1 with non-negligible probability, then $\mathsf{wExpImmutability}_{\mathcal{B},\Pi_1}(1^{\lambda})$ also returns 1 with non-negligible probability, which contradicts that Π_1 is weakly immutable. □

B.2 Unlinkability

Proof (Unlinkability). We define the following sequence of hybrid experiments:
Hyb^{b}_{0}: is identical to $\mathsf{ExpUnlink}^{b}_{\mathcal{A},\Pi_2}(1^{\lambda})$.
Hyb^{b}_{1}: is identical to Hyb^{b}_{0} except for the following change. Let $(m_0, \delta_0, \sigma_0, m_1, \delta_1, \sigma_1)$ be a query from $\mathcal{A}$ to $\mathsf{LoRSanit}\mathcal{O}_b$. Suppose that $\Pi_2.\mathsf{Vf}(\mathsf{pk}^{\dagger}_{\mathsf{S}}, \mathsf{pk}^{\dagger}_{\mathsf{Z}}, m_\beta, \sigma_\beta) = 1$ for all $\beta \in \{0,1\}$. Then if for some $\beta \in \{0,1\}$, there is no α_β which satisfies $(\mathsf{pk}^{\dagger}_{\mathsf{S}}, \mathsf{pk}^{\dagger}_{\mathsf{Z}}, m_\beta, \alpha_\beta, \sigma_{\mathsf{SS},\beta}) \in L$, the challenger aborts.

Lemma 9. *If* VRS *is strongly unforgeable, then the probability of the challenger aborting is negligible, which implies*

$$\left|\mathsf{Pr}\left[\mathsf{Hyb}^{b}_{0} = 1\right] - \mathsf{Pr}\left[\mathsf{Hyb}^{b}_{1} = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

Proof. Suppose that there exists a polynomial time adversary $\mathcal{A}$ that forces the challenger to abort in Hyb^{b}_{1} with non negligible probability, we show how to build a polynomial time adversary $\mathcal{B}$ that breaks the strong unforgeability of the VRS scheme with non-negligible probability. $\mathcal{B}$ receives the set of public keys $\{\mathsf{vpk}^{\ddagger}_{\mathsf{S}}, \mathsf{vsk}^{\ddagger}_{\mathsf{S}}\}$. It runs $\mathsf{pp}_{\mathsf{SS}} \leftarrow \Pi_1.\mathsf{Setup}(1^{\lambda}, 1^{\ell})$, $(\mathsf{spk}^{\dagger}_{\mathsf{S}}, \mathsf{ssk}^{\dagger}_{\mathsf{S}}) \leftarrow \Pi_1.\mathsf{KGen}_{\mathsf{S}}(\mathsf{pp}_{\mathsf{SS}})$ and $(\mathsf{spk}^{\dagger}_{\mathsf{Z}}, \mathsf{ssk}^{\dagger}_{\mathsf{Z}}) \leftarrow \Pi_1.\mathsf{KGen}_{\mathsf{Z}}(\mathsf{pp}_{\mathsf{SS}})$; and it sets $\mathsf{pk}^{\dagger}_{\mathsf{S}} := (\mathsf{spk}^{\dagger}_{\mathsf{S}}, \mathsf{vpk}^{\ddagger}_{\mathsf{S}})$ and $\mathsf{pk}^{\dagger}_{\mathsf{Z}} := (\mathsf{spk}^{\dagger}_{\mathsf{Z}}, \mathsf{vpk}^{\ddagger}_{\mathsf{Z}})$. It then runs $\mathcal{A}((\mathsf{pp}_{\mathsf{SS}}, \mathsf{pp}_{\mathsf{VRS}}), \mathsf{pk}^{\dagger}_{\mathsf{S}}, \mathsf{pk}^{\dagger}_{\mathsf{Z}})$. While $\mathcal{A}$ is running, $\mathcal{B}$ simulates the oracles $\Pi_2.\mathsf{Sign}\mathcal{O}$, $\Pi_2.\mathsf{San}\mathcal{O}$, $\Pi_2.\mathsf{Prove}\mathcal{O}$ and $\Pi_2.\mathsf{LoRSanit}\mathcal{O}_b$ as follows:

Answering $\mathsf{San}\mathcal{O}$ *Queries.* $\mathcal{B}$ simulates this oracle honestly except that it generates σ_{VRS} by using the oracle $\mathsf{VRS.Sign}\mathcal{O}$ on the input $(\{\mathsf{vpk_S}, \mathsf{vpk_Z}\}, 1, \mathsf{pk_S}\|\mathsf{pk_Z}\|m\|\sigma_{\mathsf{SS}})$.

Answering $\mathsf{San}\mathcal{O}$ *Queries.* $\mathcal{B}$ simulates this oracle honestly except that it generates σ_{VRS} by calling $\mathsf{VRS.Sign}\mathcal{O}$ on $(\{\mathsf{vpk_S}, \mathsf{vpk_Z}\}, 2, \mathsf{pk_S}\|\mathsf{pk_Z}\|\delta(m)\|\sigma_{\mathsf{SS}})$.

Answering $\mathsf{Prove}\mathcal{O}$ *Queries.* $\mathcal{B}$ simulates this oracle honestly except that it generates the proof π by calling $\mathsf{VRS.Prove}\mathcal{O}$ on $(\{\mathsf{vpk_S}, \mathsf{vpk_Z}\}, 1, \mathsf{pk_S}\|\mathsf{pk_Z}\|m\|\sigma_{\mathsf{SS}}, \sigma_{\mathsf{VRS}})$.

Answering $\mathsf{LoRSanit}\mathcal{O}_b$ *Queries.* Let $(m_0, \delta_0, \sigma_0, m_1, \delta_1, \sigma_1)$ be a query from $\mathcal{A}$. Suppose that $\Pi_2.\mathsf{Vf}(\mathsf{pk_S^\dagger}, \mathsf{pk_Z^\dagger}, m_\beta, \sigma_\beta) = 1$ for all $\beta \in \{0,1\}$ (otherwise the oracle outputs $\perp$). This implies that

$$\mathsf{VRS.Verify}(\{\mathsf{vpk_Z}, \mathsf{vpk_S}\}, \mathsf{pk_S^\dagger}\|\mathsf{pk_Z^\dagger}\|m_\beta\|\sigma_{\mathsf{SS},\beta}, \sigma_{\mathsf{VRS},\beta}) = 1.$$

Suppose further that for some $\beta \in \{0,1\}$, there is no α_β which satisfies $(\mathsf{pk_S^\dagger}, \mathsf{pk_Z^\dagger}, m_\beta, \alpha_\beta, \sigma_{\mathsf{SS},\beta}) \in L$. Then $\mathcal{B}$ aborts the simulation and outputs $(\{\mathsf{vpk_Z}, \mathsf{vpk_S}\}, \mathsf{pk_S^\dagger}\|\mathsf{pk_Z^\dagger}\|m_\beta\|\sigma_{\mathsf{SS},\beta}, \sigma_{\mathsf{VRS},\beta})$ as a forgery. Otherwise, $\mathcal{B}$ simulates this oracle honestly except that it generates σ_{VRS} by calling $\mathsf{VRS.Sign}\mathcal{O}$ on $(\{\mathsf{vpk_S}, \mathsf{vpk_Z}\}, \mathsf{pk_S^\dagger}\|\mathsf{pk_Z^\dagger}\|\delta_b(m_b)\|\sigma_{\mathsf{SS},b})$, where $\sigma_b = (\sigma_{\mathsf{SS},b}, \sigma_{\mathsf{VRS},b})$.

Clearly, if $\mathcal{B}$ aborts and returns a forgery, then it breaks the strong unforgeability of VRS. We can therefore assume that $\mathcal{B}$ does not abort with overwhelming probability, which implies our claim. □

Lemma 10. *If Π_1 is weakly unlinkable, then*

$$|\mathsf{Pr}\left[\mathsf{Hyb}_1^0 = 1\right] - \mathsf{Pr}\left[\mathsf{Hyb}_1^1 = 1\right]| \leq \mathsf{negl}(\lambda)\,.$$

Proof. Assume that there exists a polynomial time adversary $\mathcal{A}$ distinguishes between the above hybrids with non-negligible probability. We show how to build an algorithm $\mathcal{B}$ that breaks the weak unlinkability of Π_1. The algorithm $\mathcal{B}$ receives $(\mathsf{pp_{SS}}, \mathsf{spk_S^\ddagger}, \mathsf{spk_Z^\ddagger})$ as input, then it runs $\mathsf{pp_{VRS}} \leftarrow \mathsf{VRS.Setup}(1^\lambda)$, it sets $\mathsf{pp} := (\mathsf{pp_{SS}}, \mathsf{pp_{VRS}})$, it runs $(\mathsf{vpk_S^\dagger}, \mathsf{vsk_S^\dagger}) \leftarrow \mathsf{VRS.KGen}(\mathsf{pp_{VRS}})$ and $(\mathsf{vpk_Z^\dagger}, \mathsf{vsk_Z^\dagger}) \leftarrow \mathsf{VRS.KGen}(\mathsf{pp_{VRS}})$, and it sets $\mathsf{pk_S^\dagger} := \{\mathsf{spk_S^\ddagger}, \mathsf{vpk_S^\dagger}\}$ and $\mathsf{pk_Z} := \{\mathsf{spk_Z^\ddagger}, \mathsf{vpk_Z^\dagger}\}$. It runs $b_* \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{pk_S^\dagger}, \mathsf{pk_Z^\dagger})$. While $\mathcal{A}$ is running, $\mathcal{B}$ answers to the sign and proof oracle queries as in proof of immutability, and it simulates the other oracles $\Pi_2.\mathsf{San}\mathcal{O}$ and $\Pi_2.\mathsf{LoRSanit}\mathcal{O}_b$ to $\mathcal{A}$ as follows.

Answering $\mathsf{San}\mathcal{O}$ *Queries.* On input $(\mathsf{pk_S}, m, \delta, \sigma)$, it parses $\mathsf{pk_S}$ as $(\mathsf{spk_S}, \mathsf{vpk_S})$ and σ as $(\sigma_{\mathsf{SS}}, \sigma_{\mathsf{VRS}})$ and sets $m' := \mathsf{pk_Z^\dagger}\|m$ and δ' such that $\delta'(m') = \mathsf{pk_Z^\dagger}\|\delta(m)$. $\mathcal{B}$ generates σ'_{SS} by calling $\Pi_1.\mathsf{San}\mathcal{O}$ on $(\mathsf{spk_S}, \mathsf{pk_Z^\dagger}\|m', \delta', \sigma_{\mathsf{SS}})$. It generates the rest of the signature honestly.

Answering $\mathsf{LoRSanit}\mathcal{O}_b$ *Queries.* Let $((m_0, \delta_0, \sigma_0), (m_1, \delta_1, \sigma_1))$ be a query from $\mathcal{A}$. Note that the abort conditions in Hyb_1^0 and Hyb_1^1 are identical. Therefore if $\mathcal{B}$ aborts, then $\mathsf{Hyb}_1^0 = \mathsf{Hyb}_1^1$. Suppose that $\mathcal{B}$ does not abort. Then it must be the case that for all $\beta \in \{0,1\}$, there exists i_β, α_β which satisfies $L[i_\beta] = (\mathsf{pk_S^\dagger}, \mathsf{pk_Z^\dagger}, m_\beta, \alpha_\beta, \sigma_\beta)$. $\mathcal{B}$ can thus generate σ'_b by calling $\Pi_1.\mathsf{wLoRSanit}\mathcal{O}_b$ on $(i_0, \delta_0, i_1, \delta_1)$.

Eventually, $\mathcal{A}$ returns $b*$ which is also returned by $\mathcal{B}$. Clearly, the game is perfectly simulated for $\mathcal{A}$, therefore

$$\begin{aligned}
&\left|\mathsf{Pr}\left[\mathsf{Hyb}_1^0 = 1\right] - \mathsf{Pr}\left[\mathsf{Hyb}_1^1 = 1\right]\right| \\
=&\left|\mathsf{Pr}\left[\mathsf{wExpUnlink}^0_{\mathcal{B},\Pi_1}(1^\lambda) = 1\right] - \mathsf{Pr}\left[\mathsf{wExpUnlink}^0_{\mathcal{B},\Pi_1}(1^\lambda) = 1\right]\right| \\
\leq&\mathsf{negl}(\lambda)
\end{aligned}$$

where the inequality is due to the weak unlinkability of Π_1. □

Combining the two lemmas, we conclude that Π_2 is unlinkable. □

B.3 Strong Invisibility

Proof (Strong Invisibility). Assume that there exists a polynomial time adversary $\mathcal{A}$ that breaks the strong invisibility of Π_2. We show how to build an algorithm $\mathcal{B}$ that breaks the strong invisibility of Π_1. The algorithm $\mathcal{B}$ receives $(\mathsf{pp}_{\mathsf{SS}}, \mathsf{spk}^{\ddagger}_{\mathsf{S}}, \mathsf{spk}^{\ddagger}_{\mathsf{Z}})$ as input, then it runs $\mathsf{pp}_{\mathsf{VRS}} \leftarrow \mathsf{VRS.Setup}(1^\lambda)$, it sets $\mathsf{pp} := (\mathsf{pp}_{\mathsf{SS}}, \mathsf{pp}_{\mathsf{VRS}})$, it runs $(\mathsf{vpk}^{\dagger}_{\mathsf{S}}, \mathsf{vsk}^{\dagger}_{\mathsf{S}}) \leftarrow \mathsf{VRS.KGen}(\mathsf{pp}_{\mathsf{VRS}})$ and $(\mathsf{vpk}^{\dagger}_{\mathsf{Z}}, \mathsf{vsk}^{\dagger}_{\mathsf{Z}}) \leftarrow \mathsf{VRS.KGen}(\mathsf{pp}_{\mathsf{VRS}})$, and it sets $\mathsf{pk}^{\dagger}_{\mathsf{S}} := (\mathsf{spk}^{\ddagger}_{\mathsf{S}}, \mathsf{vpk}^{\dagger}_{\mathsf{S}})$ and $\mathsf{pk}^{\dagger}_{\mathsf{Z}} := (\mathsf{spk}^{\ddagger}_{\mathsf{Z}}, \mathsf{vpk}^{\dagger}_{\mathsf{Z}})$. It initializes $\mathcal{Q} := \emptyset$, then it runs $b' \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{pk}^{\dagger}_{\mathsf{S}}, \mathsf{pk}^{\dagger}_{\mathsf{Z}})$. While $\mathcal{A}$ is running, $\mathcal{B}$ answers to the proof oracle queries as in proof of immutability, and it simulates the other oracles $\Pi_2.\mathsf{San}\mathcal{O}'$ and $\Pi_2.\mathsf{LoRAdm}\mathcal{O}_b$ to $\mathcal{A}$ as follows.

Answering $\mathsf{San}\mathcal{O}'$ *Queries.* On input a tuple $(\mathsf{pk}_{\mathsf{S},i}, m_i, \delta_i, \sigma_i)$, the oracle returns $\perp$ if $\mathsf{pk}_{\mathsf{S},i} = \mathsf{pk}^{\dagger}_{\mathsf{S}} \wedge (\nexists (m_i, \sigma_i, \alpha) \in \mathcal{R}\ s.t.\ \forall j \notin \alpha,\ \delta_i(m_i)_j = (m_i)_j)$, otherwise it parses σ_i as $(\sigma_{\mathsf{SS},i}, \sigma_{\mathsf{VRS},i})$ and $\mathsf{pk}_{\mathsf{S},i}$ as $\{\mathsf{spk}_{\mathsf{S},i}, \mathsf{vpk}_{\mathsf{S},i}\}$, sets $m'_i := \mathsf{pk}^{\dagger}_{\mathsf{Z}} \| m_i$ and δ' such that $\delta'_i(m'_i) = \mathsf{pk}^{\dagger}_{\mathsf{Z}} \| \delta_i(m_i)$, and sends $(\mathsf{spk}_{\mathsf{S},i}, m'_i, \delta'_i, \sigma_{\mathsf{SS},i})$ to the oracle $\Pi_1.\mathsf{San}'\mathcal{O}$ and receives $\sigma'_{\mathsf{SS},i}$. It then sets $t := \mathsf{pk}_{\mathsf{S},i} \| \mathsf{pk}^{\dagger}_{\mathsf{Z}} \| \delta_i(m_i) \| \sigma'_{\mathsf{SS},i}$ and runs $\sigma'_{\mathsf{VRS},i} \leftarrow \mathsf{VRS.Sign}(\mathsf{vsk}^{\dagger}_{\mathsf{Z}}, \{\mathsf{vpk}_{\mathsf{S},i}, \mathsf{vpk}^{\dagger}_{\mathsf{Z}}\}, t)$ and sets $\sigma_i := (\sigma'_{\mathsf{SS},i}, \sigma'_{\mathsf{VRS},i})$. If $\mathsf{pk}_{\mathsf{S},i} = \mathsf{pk}^{\dagger}_{\mathsf{S}} \wedge (\exists (m_i, \sigma_i, \alpha') \in \mathcal{R}\ s.t.\ \forall j \notin \alpha',\ \delta_i(m_i)_j = (m_i)_j)$, then the oracle sets $\mathcal{R} \leftarrow \mathcal{R} \| \{(\delta_i(m_i), \sigma'_i, \alpha')\}$.

Answering $\mathsf{LoRAdm}\mathcal{O}_b$ *Queries.* On input a tuple $(\mathsf{pk}_{\mathsf{Z}}, m, \alpha_0, \alpha_1)$, it returns $\perp$ if $\neg(|\alpha_0| = |\alpha_1| = |m|)$ or $\mathsf{pk}^{\dagger}_{\mathsf{Z}} \neq \mathsf{pk}_{\mathsf{Z}} \wedge \alpha_0 \neq \alpha_1$, otherwise it parses pk_{Z} as $\{\mathsf{spk}_{\mathsf{Z}}, \mathsf{vpk}_{\mathsf{Z}}\}$ and sets for all $i \in \{0, 1\}$, $\alpha'_i := 0 \| \alpha_i$ and $m' := \mathsf{pk}_{\mathsf{Z}} \| m$. $\mathcal{B}$ sends $(\mathsf{spk}_{\mathsf{Z}}, m', \alpha'_0, \alpha'_1)$ to the oracle $\Pi_1.\mathsf{LoRAdm}\mathcal{O}_b$ that answers $\sigma_{\mathsf{SS},b}$. It sets $t := \mathsf{pk}^{\dagger}_{\mathsf{S}} \| \mathsf{pk}_{\mathsf{Z}} \| m \| \sigma_{\mathsf{SS},b}$ and runs $\sigma_{\mathsf{VRS},b} \leftarrow \mathsf{VRS.Sign}(\mathsf{vsk}^{\dagger}_{\mathsf{S}}, \{\mathsf{vpk}^{\dagger}_{\mathsf{S}}, \mathsf{vpk}_{\mathsf{Z}}\}, t)$ and returns $\sigma_b := (\sigma_{\mathsf{SS},b}, \sigma_{\mathsf{VRS},b})$ to $\mathcal{A}$. It sets $\mathcal{R} \leftarrow \mathcal{R} \| \{(m, \sigma_b, \alpha_0 \circ \alpha_1)\}$ if $\mathsf{pk}^{\dagger}_{\mathsf{Z}} = \mathsf{pk}_{\mathsf{Z}}$.

At the end of the experiment, $\mathcal{B}$ returns b'. Clearly, the experiment is perfectly simulated for $\mathcal{A}$, so $\mathsf{ExpInvisibility}^{b}_{\mathcal{B},\Pi_1}(1^\lambda)$ returns 1 with at least the probability that $\mathsf{ExpInvisibility}^{b}_{\mathcal{A},\Pi_2}(1^\lambda)$ returns 1, which is a contradiction. This concludes the proof. □

B.4 Strong Accountability

Proof (Strong Signer accountability). Assume that there exists a polynomial time adversary $\mathcal{A}$ that breaks the strong signer accountability of Π_2. We show how to build an algorithm $\mathcal{B}$ that breaks the strong accountability of VRS for $n = 1$. The algorithm $\mathcal{B}$ receives $(\mathsf{pp}_{\mathsf{VRS}}, \mathsf{vpk}^{\ddagger}_{\mathsf{Z}})$ as input, then it runs $\mathsf{pp}_{\mathsf{SS}} \leftarrow \Pi_1.\mathsf{Setup}(1^\lambda)$ and $(\mathsf{spk}^{\dagger}_{\mathsf{Z}}, \mathsf{ssk}^{\dagger}_{\mathsf{Z}}) \leftarrow \Pi_1.\mathsf{KGen}_{\mathsf{Z}}(\mathsf{pp}_{\mathsf{SS}})$. It sets $\mathsf{pp} := (\mathsf{pp}_{\mathsf{SS}}, \mathsf{pp}_{\mathsf{VRS}})$ and and $\mathsf{pk}^{\dagger}_{\mathsf{Z}} := (\mathsf{spk}^{\dagger}_{\mathsf{Z}}, \mathsf{vpk}^{\ddagger}_{\mathsf{Z}})$. It runs $(\mathsf{pk}^{*}_{\mathsf{S}}, m^*, \sigma^*, \pi^*) \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{pk}^{\dagger}_{\mathsf{Z}})$. While $\mathcal{A}$ is running, $\mathcal{B}$ runs the sanitize oracle $\Pi_2.\mathsf{San}\mathcal{O}$ as follows.

Answering Sanitize Oracle Queries. On input a tuple $(\mathsf{pk}_{\mathsf{S},i}, m_i, \delta_i, \sigma_i)$, the oracle parses σ_i as $(\sigma_{\mathsf{SS},i}, \sigma_{\mathsf{VRS},i})$ and $\mathsf{pk}_{\mathsf{S},i}$ as $\{\mathsf{spk}_{\mathsf{S},i}, \mathsf{vpk}_{\mathsf{S},i}\}$, and sets $m'_i = \mathsf{pk}^{\dagger}_{\mathsf{Z}} \| m_i$ and δ'_i such that $\delta'_i(m'_i) = \mathsf{pk}^{\dagger}_{\mathsf{Z}} \| \delta_i(m_i)$. The oracle computes $\sigma'_{\mathsf{SS},i} \leftarrow \Pi_1.\mathsf{San}(\mathsf{ssk}^{\dagger}_{\mathsf{Z}}, \mathsf{spk}_{\mathsf{S},i}, m'_i, \delta'_i, \sigma_{\mathsf{SS},i})$ and sends $(\{\mathsf{vpk}_{\mathsf{S},i}, \mathsf{vpk}^{\ddagger}_{\mathsf{Z}}\}, 1, \mathsf{pk}_{\mathsf{S},i} \| \mathsf{pk}^{\dagger}_{\mathsf{Z}} \| \delta_i(m_i) \| \ \sigma'_{\mathsf{SS},i})$ to the oracle $\mathsf{VRS.Sign}\mathcal{O}$ and receives $\sigma'_{\mathsf{VRS},i}$. It sets $\sigma'_i := (\sigma'_{\mathsf{SS},i}, \sigma'_{\mathsf{VRS},i})$ and returns it to $\mathcal{A}$.

$\mathcal{B}$ parses σ^* as $(\sigma^*_{\mathsf{SS}}, \sigma^*_{\mathsf{VRS}})$ and $\mathsf{pk}^*_{\mathsf{S}}$ as $\{\mathsf{spk}^*_{\mathsf{S}}, \mathsf{vpk}_{\mathsf{S}}{}^*\}$. It sets $L^{**} := \{\mathsf{vpk}_{\mathsf{S}}{}^*, \mathsf{vpk}^{\ddagger}_{\mathsf{Z}}\}$, $m^{**} := \sigma^*_{\mathsf{SS}}$, $\sigma^{**} := \sigma^*_{\mathsf{VRS}}$, $\mathsf{pk}^{**} := \mathsf{vpk}_{\mathsf{S}}{}^*$ and $\pi^{**} := \pi^*$. Finally, $\mathcal{B}$ returns $(L^{**}, m^{**}, \sigma^{**}, \mathsf{pk}^{**}, \pi^{**})$.

Clearly, the experiment is perfectly simulated for $\mathcal{A}$. Assume that $\mathcal{A}$ wins its experiment, then for all i the following holds:

- $(\mathsf{pk}^*_{\mathsf{S}}, m^*, \sigma^*) \neq (\mathsf{pk}_{\mathsf{S},i}, \delta_i(m_i), \sigma'_i)$. Parse σ^* as $(\sigma^*_{\mathsf{SS}}, \sigma^*_{\mathsf{VRS}})$. By rearranging the terms, we have $((\mathsf{pk}^*_{\mathsf{S}}, \mathsf{pk}^{\dagger}_{\mathsf{Z}}, m^*, \sigma^*_{\mathsf{SS}}), \sigma^*_{\mathsf{VRS}}) \neq ((\mathsf{pk}_{\mathsf{S},i}, \mathsf{pk}^{\dagger}_{\mathsf{Z}}, \delta_i(m_i), \sigma'_{\mathsf{SS},i}), \sigma'_{\mathsf{VRS},i})$.
- $\Pi_2.\mathsf{Verify}(\mathsf{pk}^*_{\mathsf{S}}, \mathsf{pk}^{\dagger}_{\mathsf{Z}}, m^*, \sigma^*) = 1$. It implies that $\mathsf{VRS.Verify}(L^{**}, \sigma^{**}, m^{**}) = 1$.
- $\Pi_2.\mathsf{Judge}(\mathsf{pk}^*_{\mathsf{S}}, \mathsf{pk}^{\dagger}_{\mathsf{Z}}, m^*, \sigma^*, \pi^*) \neq \mathsf{S}$. So $\mathsf{VRS.Judge}(L^{**}, m^{**}, \sigma^{**}, \mathsf{pk}^{**}, \pi^{**}) = 0$.

Finally, note that $L^{**} \subseteq (\{\mathsf{vpk}^{\ddagger}_{\mathsf{Z}}\} \cup \{\mathsf{pk}^{**}\})$. Therefore, $\mathsf{ExpAcc}_{\mathsf{VRS},\mathcal{B}}(1^\lambda)$ returns 1 with at least the probability that $\mathsf{ExpSigAcc}_{\mathcal{A},\Pi_2}(1^\lambda)$ returns 1, which is a contradiction. This concludes the proof. □

Proof (Sanitizer accountability). Assume that there exists a polynomial time adversary $\mathcal{A}$ that breaks the strong sanitizer accountability of Π_2. We show how to build an algorithm $\mathcal{B}$ that breaks the strong non-seizability of VRS. The algorithm $\mathcal{B}$ receives $(\mathsf{pp}_{\mathsf{VRS}}, \mathsf{vpk}_{\mathsf{S}}^{\ddagger})$ as input, then it runs $\mathsf{pp}_{\mathsf{SS}} \leftarrow \Pi_1.\mathsf{Setup}(1^\lambda)$ and $(\mathsf{spk}_{\mathsf{S}}^{\dagger}, \mathsf{ssk}_{\mathsf{S}}^{\dagger}) \leftarrow \mathsf{VRS.KGen}(\mathsf{pp}_{\mathsf{SS}})$. It sets $\mathsf{pp} := (\mathsf{pp}_{\mathsf{SS}}, \mathsf{pp}_{\mathsf{VRS}})$ and $\mathsf{pk}_{\mathsf{S}}^{\dagger} := (\mathsf{spk}_{\mathsf{S}}^{\dagger}, \mathsf{vpk}_{\mathsf{S}}^{\ddagger})$. It runs $(\mathsf{pk}_{\mathsf{Z}}^{*}, m^*, \sigma^*) \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{pk}_{\mathsf{S}}^{\dagger})$. While $\mathcal{A}$ is running, $\mathcal{B}$ simulates the oracles $\Pi_2.\mathsf{Sign}\mathcal{O}$ and $\Pi_2.\mathsf{Prove}\mathcal{O}$ as follows.

Answering $\mathsf{Sign}\mathcal{O}$ *Oracle Queries.* On input a tuple $(\mathsf{pk}_{\mathsf{Z},i}, m_i, \alpha_i)$, the oracle parses $\mathsf{pk}_{\mathsf{Z},i}$ as $\{\mathsf{spk}_{\mathsf{Z},i}, \mathsf{vpk}_{\mathsf{Z},i}\}$. The oracle computes $\sigma_{\mathsf{SS},i} \leftarrow \Pi_1.\mathsf{Sign}(\mathsf{ssk}_{\mathsf{S}}^{\dagger}, \mathsf{spk}_{\mathsf{Z},i}, \mathsf{pk}_{\mathsf{Z},i} \| m_i, 0 \| \alpha_i)$ and sends $(\{\mathsf{vpk}_{\mathsf{S}}^{\ddagger}, \mathsf{vpk}_{\mathsf{Z},i}\}, 1, {\mathsf{pk}_{\mathsf{S}}}^{\dagger} \| \mathsf{pk}_{\mathsf{Z},i} \| m_i \| \sigma_{\mathsf{SS},i})$ to the oracle $\mathsf{VRS.Sign}\mathcal{O}$ and receives $\sigma_{\mathsf{VRS},i}$. It sets $\sigma_i := (\sigma_{\mathsf{SS},i}, \sigma_{\mathsf{VRS},i})$ and returns it to $\mathcal{A}$.

Answering $\mathsf{Prove}\mathcal{O}$ *Oracle Queries.* On input a tuple $(\tilde{\mathsf{pk}}_{\mathsf{Z},i}, \tilde{m}_i, \tilde{\sigma}_i)$, the oracle parses $\tilde{\mathsf{pk}}_{\mathsf{Z},i}$ as $\{\tilde{\mathsf{spk}}_{\mathsf{Z},i}, \tilde{\mathsf{vpk}}_{\mathsf{Z},i}\}$ and $\tilde{\sigma}_i$ as $(\tilde{\sigma}_{\mathsf{SS},i}, \tilde{\sigma}_{\mathsf{VRS},i})$. The oracle sends $(\{\mathsf{vpk}_{\mathsf{S}}^{\ddagger}, \tilde{\mathsf{vpk}}_{\mathsf{Z},i}\}, 1, {\mathsf{pk}_{\mathsf{S}}}^{\dagger} \| \tilde{\mathsf{pk}}_{\mathsf{Z},i} \| \tilde{m}_i \| \tilde{\sigma}_{\mathsf{SS},i})$ to the oracle $\mathsf{VRS.Prove}\mathcal{O}$ and receives π_i, then it returns it to $\mathcal{A}$.

$\mathcal{B}$ parses σ^* as $(\sigma_{\mathsf{SS}}^*, \sigma_{\mathsf{VRS}}^*)$ and $\mathsf{pk}_{\mathsf{Z}}^*$ as $\{\mathsf{spk}_{\mathsf{Z}}^*, {\mathsf{vpk}_{\mathsf{Z}}}^*\}$. It sets $L^{**} := \{\mathsf{vpk}_{\mathsf{S}}^{\ddagger}, {\mathsf{vpk}_{\mathsf{Z}}}^*\}$, $m^{**} := \sigma_{\mathsf{SS}}^*$, $\sigma^{**} := \sigma_{\mathsf{VRS}}^*$. Finally, $\mathcal{B}$ returns $(L^{**}, m^{**}, \sigma^{**})$.

Clearly, the experiment is perfectly simulated for $\mathcal{A}$. Assume that $\mathcal{A}$ wins its experiment, then for all i, and for any $\pi \leftarrow \Pi_2.\mathsf{Prove}(\mathsf{sk}_{\mathsf{S}}^{\dagger}, \mathsf{pk}_{\mathsf{Z}}^*, m^*, \sigma^*)$, the following holds:

- $(\mathsf{pk}_{\mathsf{Z}}^*, m^*, \sigma^*) \neq (\mathsf{pk}_{\mathsf{Z},i}, m_i, \sigma_i)$. Parse σ^* as $(\sigma_{\mathsf{SS}}^*, \sigma_{\mathsf{VRS}}^*)$. By rearranging the terms, we have $((\mathsf{pk}_{\mathsf{S}}^{\dagger}, \mathsf{pk}_{\mathsf{Z}}^*, m^*, \sigma_{\mathsf{SS}}^*), \sigma_{\mathsf{VRS}}^*) \neq ((\mathsf{pk}_{\mathsf{S}}^{\dagger}, \mathsf{pk}_{\mathsf{Z},i}, m_i, \sigma_{\mathsf{SS},i}), \sigma_{\mathsf{VRS},i})$.
- $\mathsf{Verify}(\mathsf{pk}_{\mathsf{S}}^{\dagger}, \mathsf{pk}_{\mathsf{Z}}^*, m^*, \sigma^*) = 1$. It implies that $\mathsf{VRS.Verify}(L^{**}, \sigma^{**}, m^{**}) = 1$.
- $\mathsf{Judge}(\mathsf{pk}_{\mathsf{S}}^{\dagger}, \mathsf{pk}_{\mathsf{Z}}^*, m^*, \sigma^*, \pi^*) \neq \mathsf{Z}$. So $\mathsf{VRS.Judge}(L^{**}, m^{**}, \sigma^{**}, \mathsf{pk}^{**}, \pi^{**}) \neq 0$.

Therefore, $\mathsf{ExpNonSeiz}_{\mathsf{VRS},\mathcal{B}}(1^\lambda)$ returns 1 with at least the probability that $\mathsf{ExpSanAcc}_{\mathcal{A},\Pi_2}(1^\lambda)$ returns 1, which is a contradiction. This concludes the proof. □

B.5 Strong Proof-Restricted Transparency

Proof (Strong Proof-Restricted Transparency). We prove by hybrid argument. For any polynomial time algorithm $\mathcal{A}$, we define the hybrid experiment $\mathsf{ExpHTransparency}_{\mathcal{A},\Pi_2}(1^\lambda)$ as the same experiment as $\mathsf{ExpTransparency}^0_{\mathcal{A},\Pi_2}(1^\lambda)$ except that, on input $(m_i, \delta_i, \alpha_i)$ the oracle $\Pi_2.\mathsf{Sign}/\mathsf{San}\mathcal{O}_b$ returns $\sigma_i = (\sigma_{\mathsf{SS},i}, \sigma_{\mathsf{VRS},i})$ where $\sigma_{\mathsf{VRS},i}$ is a signature generated by the signer instead of the sanitizer. Concretely, $\sigma_{\mathsf{VRS},i}$ is generated as $\sigma_{\mathsf{VRS},i} \leftarrow \mathsf{VRS.Sign}(\mathsf{vsk}_{\mathsf{S}}, \{\mathsf{vpk}_{\mathsf{S}}, \mathsf{vpk}_{\mathsf{Z}}\}, \mathsf{pk}_{\mathsf{S}} \| \mathsf{pk}_{\mathsf{Z}} \| \delta_i(m_i) \| \sigma_{\mathsf{SS},i})$.

We first argue that for any PPT adversary $\mathcal{A}$, the probabilities that the experiments $\mathsf{ExpHTransparency}_{\mathcal{A},\Pi_2}(1^\lambda)$ and $\mathsf{ExpTransparency}^0_{\mathcal{A},\Pi_2}(1^\lambda)$ output 1 are negligibly close if VRS is anonymous. Suppose not, we construct a PPT adversary $\mathcal{B}$ against the anonymity of VRS.

$\mathcal{B}$ receives $(\mathsf{pp}_{\mathsf{VRS}}, \{\mathsf{vpk}_{\mathsf{Z}}^{\ddagger}, \mathsf{vpk}_{\mathsf{S}}^{\ddagger}\})$ as input, then it runs $\mathsf{pp}_{\mathsf{SS}} \leftarrow \mathsf{Setup}_{\mathsf{SS}}(1^\lambda)$, $(\mathsf{spk}_{\mathsf{S}}^{\dagger}, \mathsf{ssk}_{\mathsf{S}}^{\dagger}) \leftarrow \Pi_1.\mathsf{KGen}_{\mathsf{S}}(\mathsf{pp}_{\mathsf{SS}})$ and $(\mathsf{spk}_{\mathsf{Z}}^{\dagger}, \mathsf{ssk}_{\mathsf{Z}}^{\dagger}) \leftarrow \Pi_1.\mathsf{KGen}_{\mathsf{Z}}(\mathsf{pp}_{\mathsf{SS}})$, and it sets $\mathsf{pp} := (\mathsf{pp}_{\mathsf{SS}}, \mathsf{pp}_{\mathsf{VRS}})$, $\mathsf{pk}_{\mathsf{S}}^{\dagger} := (\mathsf{spk}_{\mathsf{S}}^{\dagger}, \mathsf{vpk}_{\mathsf{S}}^{\ddagger})$ and $\mathsf{pk}_{\mathsf{Z}}^{\dagger} := (\mathsf{spk}_{\mathsf{Z}}^{\dagger}, \mathsf{vpk}_{\mathsf{Z}}^{\ddagger})$. $\mathcal{B}$ then simulates the $\Pi_2.\mathsf{Sign}\mathcal{O}$, $\Pi_2.\mathsf{San}\mathcal{O}$, $\Pi_2.\mathsf{Prove}\mathcal{O}$, and $\Pi_2.\mathsf{Sign}/\mathsf{San}\mathcal{O}_b$ oracles of Π_2 for $\mathcal{A}$ honestly, except that whenever the $\mathsf{VRS.Sign}\mathcal{O}$ and $\mathsf{VRS.Prove}\mathcal{O}$ algorithms are to be called, it queries the corresponding oracles provided by the anonymity challenger of VRS instead.

More specifically, we pay special attention to how $\mathcal{B}$ answers $\Pi_2.\mathsf{Sign}/\mathsf{San}\mathcal{O}_b$ and $\Pi_2.\mathsf{Prove}\mathcal{O}$ oracle queries:

- When $\mathcal{A}$ queries $\Pi_2.\mathsf{Sign}/\mathsf{San}\mathcal{O}_b$ on input $(m_i, \delta_i, \alpha_i)$, $\mathcal{B}$ computes $\sigma_{\mathsf{SS},i}$ as specified in $\mathsf{ExpTransparency}^0_{\mathcal{A},\Pi_2}(1^\lambda)$, and queries the $\mathsf{LoRSign}\mathcal{O}_b$ oracle on $(\{\mathsf{vpk}_{\mathsf{S}}^{\ddagger}, \mathsf{vpk}_{\mathsf{Z}}^{\ddagger}\}, \mathsf{pk}_{\mathsf{S}}^{\dagger} \| \mathsf{pk}_{\mathsf{Z}}^{\dagger} \| \delta_i(m_i) \| \sigma_{\mathsf{SS},i})$ and obtains $\sigma_{\mathsf{VRS},i}$. $\mathcal{B}$ then outputs $\sigma_i := (\sigma_{\mathsf{SS},i}, \sigma_{\mathsf{VRS},i})$.
- When $\mathcal{A}$ queries $\Pi_2.\mathsf{Prove}\mathcal{O}$ on input $(\mathsf{pk}'_{\mathsf{Z},j}, m'_j, \sigma'_j)$, $\mathcal{B}$ parses σ'_j as $(\sigma'_{\mathsf{SS},j}, \sigma'_{\mathsf{VRS},j})$ and queries the $\mathsf{VRS.Prove}\mathcal{O}$ oracle on $(\{\mathsf{vpk}_{\mathsf{S}}^{\ddagger}, \mathsf{vpk}_{\mathsf{Z}}^{\ddagger}\}, \mathsf{pk}_{\mathsf{S}}^{\dagger} \| \mathsf{pk}_{\mathsf{Z}}^{\dagger} \| m'_j \| \sigma'_{\mathsf{SS},j}, \sigma'_{\mathsf{VRS},j}, 1)$, where 1 specifies the signer. In return, $\mathcal{B}$ receives a bit d and outputs Z if $d = 0$ and S otherwise.

Clearly, depending on the choice of the anonymity challenger of VRS, $\mathcal{B}$ simulates either $\mathsf{ExpHTransparency}_{\mathcal{A},\Pi_2}(1^\lambda)$ or $\mathsf{ExpTransparency}^0_{\mathcal{A},\Pi_2}(1^\lambda)$ perfectly. Therefore, if the probabilities that the experiments $\mathsf{ExpHTransparency}_{\mathcal{A},\Pi_2}(1^\lambda)$ and $\mathsf{ExpTransparency}^b_{\mathcal{A},\Pi_2}(1^\lambda)$ output 1 differ in non-negligible probability, then the probabilities that $\mathsf{ExpAnon}^0_{\mathsf{VRS},\mathcal{B}}(1^\lambda)$ and $\mathsf{ExpAnon}^1_{\mathsf{VRS},\mathcal{B}}(1^\lambda)$ output 1 differ by the same non-negligible probability, which contracts the assumption that VRS is anonymous.

Next, we argue that for any PPT adversary $\mathcal{A}$, the probabilities that the experiments $\mathsf{ExpHTransparency}_{\mathcal{A},\Pi_2}(1^\lambda)$ and $\mathsf{ExpTransparency}^1_{\mathcal{A},\Pi_2}(1^\lambda)$ output 1 are negligibly close if Π_1 is strongly proof-restricted transparent. Suppose not, we construct a PPT adversary $\mathcal{B}$ against the strongly proof-restricted transparency of Π_1.

$\mathcal{B}$ receives $(\mathsf{pp}_{\mathsf{SS}}, \mathsf{spk}^\ddagger_{\mathsf{Z}}, \mathsf{spk}^\ddagger_{\mathsf{S}})$, then it runs $\mathsf{pp}_{\mathsf{VRS}} \leftarrow \mathsf{VRS.Setup}(1^\lambda)$, $(\mathsf{vpk}^\dagger_{\mathsf{Z}}, \mathsf{vsk}^\dagger_{\mathsf{Z}}) \leftarrow \mathsf{VRS.KGen}(\mathsf{pp}_{\mathsf{VRS}})$ and $(\mathsf{vpk}^\dagger_{\mathsf{S}}, \mathsf{vsk}^\dagger_{\mathsf{S}}) \leftarrow \mathsf{VRS.KGen}(\mathsf{pp}_{\mathsf{VRS}})$, and it sets $\mathsf{pp} := (\mathsf{pp}_{\mathsf{SS}}, \mathsf{pp}_{\mathsf{VRS}})$, $\mathsf{pk}^\dagger_{\mathsf{S}} := (\mathsf{spk}^\ddagger_{\mathsf{S}}, \mathsf{vpk}^\dagger_{\mathsf{S}})$ and $\mathsf{pk}^\dagger_{\mathsf{Z}} := (\mathsf{spk}^\ddagger_{\mathsf{Z}}, \mathsf{vpk}^\dagger_{\mathsf{Z}})$. $\mathcal{B}$ then simulates the $\Pi_2.\mathsf{Sign}\mathcal{O}$, $\Pi_2.\mathsf{San}\mathcal{O}$, $\Pi_2.\mathsf{Prove}\mathcal{O}$, and $\Pi_2.\mathsf{Sign/San}\mathcal{O}_b$ oracles of Π_2 for $\mathcal{A}$ by querying the corresponding oracles of Π_1 and, when appropriate, signing their outputs using the honestly generated VRS secret keys.

Clearly, depending on the choice of the transparency challenger of Π_1, $\mathcal{B}$ simulates either the experiment $\mathsf{ExpHTransparency}_{\mathcal{A},\Pi_2}(1^\lambda)$ or $\mathsf{ExpTransparency}^1_{\mathcal{A},\Pi_2}(1^\lambda)$ perfectly. Therefore, if the probabilities that the experiments $\mathsf{ExpHTransparency}_{\mathcal{A},\Pi_2}(1^\lambda)$ and $\mathsf{ExpTransparency}^1_{\mathcal{A},\Pi_2}(1^\lambda)$ output 1 differ in non-negligible probability, then the probabilities that $\mathsf{ExpTransparency}^0_{\mathcal{B},\Pi_1}(1^\lambda)$ and $\mathsf{ExpTransparency}^1_{\mathcal{B},\Pi_1}(1^\lambda)$ output 1 differ by the same non-negligible probability, which contracts the assumption that Π_1 is strongly proof-restricted transparent. □