Fully Invisible Protean Signatures Schemes

Stephan Krenn[1], Henrich C. Pöhls[2], Kai Samelin[3], and Daniel Slamanig[1]

[1] AIT Austrian Institute of Technology, Vienna, Austria
{stephan.krenn,daniel.slamanig}@ait.ac.at
[2] ISL & Chair of IT-Security, University of Passau, Passau, Germany
hp@sec.uni-passau.de
[3] TÜV Rheinland i-sec GmbH, Hallbergmoos, Germany
kaispapers@gmail.com

Abstract. Protean Signatures (PS), recently introduced by Krenn et al. (CANS'18), allow a semi-trusted third party, named the sanitizer, to modify a signed message in a controlled way. The sanitizer can edit signer-chosen parts to arbitrary bitstrings, while the sanitizer can also redact admissible parts, which are also chosen by the signer. Thus, PSs generalize both redactable signature (RS) and sanitizable signature (SS) into a single notion. However, the current definition of invisibility does not prohibit that an outsider can decide which parts of a message are *redactable* — only which parts can be edited are hidden. This negatively impacts on the privacy guarantees provided by the state-of-the-art definition.
We extend PSs to be *fully* invisible. This strengthened notion guarantees that an outsider can neither decide which parts of a message can be edited nor which parts can be redacted. To achieve our goal, we introduce the new notions of Invisible RSs and Invisible Non-Accountable SSs (SS′), along with a consolidated framework for aggregate signatures. Using those building blocks, our resulting construction is significantly more efficient than the original scheme by Krenn et al., which we demonstrate in a prototypical implementation.

1 Introduction

In their standard definition, digital signatures prohibit any kind of modification of signed messages. This means, that only the holder of the secret signing key sk_Σ can sign messages [31]. Still, as time showed [6], there are a plethora of application scenarios where a later modification of signed messages by a (semi-trusted) third party has its merits. Consider the following use-case, based on the handling of patient data [2,40,53,57]:

Assume that a M.D. always signs the complete record of each patient. Further assume that each of those records consists of the patient's name, its insurance number and the treatments given. After the patient is released from the hospital, the responsible accountant receives the complete signed record corresponding to the to-be-released patient to be able to write a bill to the patient's insurance company.

It is not hard to see that this process is not very privacy-friendly, especially from the patient's point of view, as the accountant receives all information related to patient. However, most of the information is not relevant for writing the bill, e.g., the patient's name. Thus, a solution is to only give the treatments and the insurance number to the accountant, anonymizing the paperwork. The major obstacle is that standard digital signatures prohibit any alterations, and thus the M.D. either needs to re-sign the document, or an additional trusted entity does need to sign on behalf of the M.D. Still, both solutions are not very satisfactory, as both induce additional overhead, one might even be impossible, e.g., if the M.D. is no longer employed. We conclude that modifying signed messages in a controlled way does have its merits.

1.1 Motivation and Contribution

Strictly speaking, the above application scenario only requires that parts of a signed message can be redacted without invalidating the signature, which is achieved by redactable signatures (RS) [36,54]. However, as already shown by Bilzhause et al., editing, but not redacting, parts of a message also has many use-cases, including secure routing, document sanitization, and outsourcing of computation [6]. This is achieved by

sanitizable signatures (SS) [2]. Thus, a SS only allows for editing parts of a message, while a RS only allows for redacting parts.

Bilzhause et al. [6] then asked the next logical question: Can SSs and RSs be combined, enabling signatures which allow for editing and redacting blocks at the same time? This question was answered by Krenn et al. by introducing "Protean Signatures" (PS) [40], which allow subsequent editing and redacting simultaneously. They provide a formal security model, a provably secure construction, and present first implementation results.

However, their definition of invisibility only guarantees that an outsider cannot decide which parts of a message are editable, but not which are redactable. This negatively impacts on the privacy guarantees their construction provides. Moreover, their corresponding implementation shows that their construction cannot be considered *practically* efficient.

We extend their work in the following areas: (1) As our main contribution we introduce a stronger invisibility definition for PSs. In more detail, in our definition, an outsider can neither decide which parts of a message are redactable nor which parts are editable. (2) To show that our strengthened definition is actually achievable, we provide an altered (black-box) construction of a PS, derived from the original one by Krenn et al. [40]. (3) We provide a new framework for aggregate signatures [8], which may be useful in other contexts as well. Namely, we introduce the new notions of strict aggregate uniqueness, a black-box "no-extraction" notion, and explicitly allow for de-aggregation. (4) Our construction makes use of two new primitives which we also introduce: An invisible designated redactor RS, and a non-accountable invisible SS′. In an invisible designated redactor RS, only a signer-chosen semi-trusted third party can redact, while an outsider cannot decide which parts are redactable. Likewise, non-accountable invisible SS′s behave as standard invisible SSs, but do not offer any form of accountability. This allows for a far more efficient instantiation. For both new notions, we provide formal frameworks, formal security models, and provably secure instantiations. (5) We have implemented our scheme in a prototypical way. Using our new primitives, the resulting construction is an order of magnitude more efficient than the one given by Krenn et al. [40], and can be considered really practical. (6) Last, but not least, we provide a stronger definition of invisibility for standard SSs, where the adversary is now able to query *arbitrary* messages to the sanitization oracle.

1.2 Related Work

Signatures allowing for subsequent alterations received a lot of attention in the recent past, as it became apparent that there are many application scenarios where signed messages need to be modified in a controlled way [1,6,23,29]. This weakens the standard unforgeability definition, where the messages protected by signatures cannot be altered at all, which is clearly not avoidable, if one wants to allow for modifications or derivations.

From our perspective, existing work can be grouped into three, not always distinct, directions. The first direction are homomorphic signatures [1,7,36,55], and some other closely related concepts [9,56]. Homomorphic signatures take several (signed) messages as input and can be used to compute functions on authenticated data-sets. In such schemes, an entity not holding any secrets can derive a new (valid) signature σ' on $f(m)$, where the function f is public.

Related are RSs, where anyone (i.e., no secrets are required) can publish a subset of signed data, along with a new signature σ'. To illustrate this, let m = (I, do, not, like, fish) along with a valid redactable signature σ. Anyone can then derive a signature σ' on m' = (I, like, fish), i.e., redact the second and third block m^2 = do and m^3 = not, if both blocks are marked as redactable. The original ideas of RSs [36,54] were later formalized [10,43], including adding new values after signure generation [51]. Then, RSs have been extended to allow for additional use-cases, including adding accountability [48], discussing their relation to SSs [22], allowing for redactable structure [52], prohibiting additional redactions [33,34,35,46], yet also defining dependencies between different parts of a message [53]. Moreover, there are also some real-world implementations

of this primitive proving that they are practical [49,57]. All these approaches (but accountability) have later been unified into a generalized framework by Derler et al. [25]. We stress that the work by Izu et al. [34] addresses the case of "sanitizable and deletable signatures". However, they actually address the case of RSs and not SSs. In particular, in their scheme, a third party can decide whether a redaction is visible or not, but does not allow for any other alterations. We follow the nomenclature clarified by Bilzhause et al. [6], and thus classify the work by Izu et al. [34] as an RS.

In contrast, SSs allow editing of signer-chosen blocks of signed messages by a semi-trusted entity named the sanitizer [2]. In particular, the sanitizer holds its own secret key and can derive new messages, along with the corresponding signatures, but cannot completely redact blocks. For example, if $m = (\text{I, do, not, like, fish})$ (and m^5 is admissible, i.e., modifiable), then the sanitizer can, e.g., derive a new signature σ' on the message $m' = (\text{I, do, not, like, meat})$. Even though this seems to be off the limits, it turned out that this primitive has many real-life application scenarios, see, e.g., Bilzhause et al. [6]. After the initial ideas by Ateniese et al. [2], SSs also received a lot of attention in the recent past. Namely, the first thorough security model was given by Brzuska et al. [11] (later slightly modified by Gong et al. [32]), which was later extended for multiple signers/sanitizers [12,18], unlinkability (which means a derived signatures cannot be linked to its original) [13,15,28,44], trapdoor SSs (where a signer can choose additional sanitizers after signature generation) [19,58], non-interactive public-accountability (an outsider can determine which party is accountable for a given valid message/signature pair) [14], limiting the sanitizer to signer-chosen values [17,26,37], invisibility (meaning that an outsider cannot derive which parts of a message are sanitizable) [3,16,27] and the case of strongly unforgeable signatures [41]. All these extensions allow for additional use-cases of this primitive [6].

Additional related work is given in some recent surveys [6,23,30]. We stress that a slightly altered SS can be used to "mimic" an RS by defining a special symbol to which the specific block is sanitized to, which then marks the block as "redacted". However, as shown by de Meer et al. [22], this has a negative impact on the privacy guarantees of the resulting scheme because the special symbol remains visible. For example, $m' = (\text{I, like, fish})$ is clearly different from $m' = (\text{I, } \perp, \perp, \text{ like, fish})$. We stress that our scheme supports both possibilities, i.e., visible and non-visible (transparent) redactions, adding additional freedom.

2 Preliminaries and Notation

We now give our notation and the required preliminaries.

2.1 Notation

The main security parameter is denoted by $\lambda \in \mathbb{N}$. All algorithms implicitly take 1^λ as an additional input. We write $a \leftarrow A(x)$ if a is assigned to the output of the deterministic algorithm A with input x. If an algorithm A is probabilistic, we write $a \leftarrow_r A(x)$. If we want to make the random coins r used explicit, we write $a \leftarrow_r A(x; r)$. Otherwise, we assume that they are drawn internally. An algorithm is efficient if it runs in probabilistic polynomial time (PPT) in the length of its input. For the remainder of this paper, all algorithms are PPT if not explicitly mentioned otherwise. Most algorithms may return a special error symbol $\perp \notin \{0,1\}^*$, denoting an exception. If S is a set, we write $a \leftarrow_r S$ to denote that a is chosen uniformly at random from S. For a message $m = (m^1, m^2, \ldots, m^{\ell_m})$, m^i is called a block and $\ell_m \in \mathbb{N}$ denotes the number of blocks in m. If m is clear from the context, it is dropped from ℓ_m. To shorten notation, we use $[a..b]$ (both a and b, $b \geq a$, are always positive natural numbers) for the set $\{a, a+1, \ldots, b\}$. A function $\nu : \mathbb{N} \to \mathbb{R}_{\geq 0}$ is negligible, if it vanishes faster than every inverse polynomial, i.e., $\forall k \in \mathbb{N}$, $\exists n_0 \in \mathbb{N}$ such that $\nu(n) \leq n^{-k}$, $\forall n > n_0$.

2.2 Building Blocks

We now present our required building blocks. These include labeled IND-CCA2 secure public-key encryption schemes (Π), sanitizable signature (SS), PRFs, and aggregate signatures (Σ).

$\mathbf{Exp}^{\mathsf{IND\text{-}CCA2}}_{\mathcal{A},\Pi}(\lambda)$:
$\mathsf{pp}_\Pi \leftarrow_r \mathsf{PPGen}^\Pi(1^\lambda)$
$(\mathsf{sk}_\Pi, \mathsf{pk}_\Pi) \leftarrow_r \mathsf{KGen}^\Pi(\mathsf{pp}_\Pi)$
$b \leftarrow_r \{0,1\}$
$(m_0^*, m_1^*, \vartheta^*, state_\mathcal{A}) \leftarrow_r \mathcal{A}^{\mathsf{Dec}^\Pi(\mathsf{sk}_\Pi,\cdot,\cdot)}(\mathsf{pk}_\Pi)$
If $|m_0^*| \neq |m_1^*| \vee m_0^* \notin \mathcal{M} \vee m_1^* \notin \mathcal{M}$:
  $c^* \leftarrow \perp$
Else:
  $c^* \leftarrow_r \mathsf{Enc}^\Pi(\mathsf{pk}_\Pi, m_b^*, \vartheta^*)$
$a \leftarrow_r \mathcal{A}^{\mathsf{Dec}^{\Pi\prime}(\mathsf{sk}_\Pi,\cdot,\cdot)}(state_\mathcal{A}, c^*)$
  where $\mathsf{Dec}^{\Pi\prime}(\mathsf{sk}_\Pi,\cdot,\cdot)$ behaves as $\mathsf{Dec}^\Pi(\mathsf{sk}_\Pi,\cdot,\cdot)$,
    but returns $\perp$, if (c^*, ϑ^*) is queried.
return 1, if $a = b$
return 0

Fig. 1: Π IND-CCA2-Security

Labeled Public-Key Encryption Schemes A labeled public-key encryption scheme Π allows to encrypt a message m using a given public key pk_Π and label $\vartheta \in \{0,1\}^*$. In a nutshell, the given ciphertext leaks no information about the contained message, except its length, if the corresponding secret key sk_Π is not known:

Definition 1 (Labeled Public-Key Encryption). *A labeled public-key encryption scheme Π consists of four algorithms $\{\mathsf{PPGen}^\Pi, \mathsf{KGen}^\Pi, \mathsf{Enc}^\Pi, \mathsf{Dec}^\Pi\}$, such that:*

PPGen^Π**.** *The algorithm PPGen^Π outputs the public parameters of the scheme:*

$$\mathsf{pp}_\Pi \leftarrow_r \mathsf{PPGen}^\Pi(1^\lambda)$$

It is assumed that pp_Π is implicit input to all other algorithms.

KGen^Π**.** *The algorithm KGen^Π outputs the key pair, on input pp_Π:*

$$(\mathsf{sk}_\Pi, \mathsf{pk}_\Pi) \leftarrow_r \mathsf{KGen}^\Pi(\mathsf{pp}_\Pi)$$

Enc^Π**.** *The algorithm Enc^Π gets as input the public key pk_Π, a message m, and a label $\vartheta \in \{0,1\}^*$ to encrypt. It outputs a ciphertext c:*

$$c \leftarrow_r \mathsf{Enc}^\Pi(\mathsf{pk}_\Pi, m, \vartheta)$$

Dec^Π**.** *The deterministic algorithm Dec^Π outputs a message m (or $\perp$, if the ciphertext is invalid) on input sk_Π, ϑ and a ciphertext c:*

$$m \leftarrow \mathsf{Dec}^\Pi(\mathsf{sk}_\Pi, c, \vartheta)$$

Π IND-CCA2-Security We need IND-CCA2-security (and perfect correctness) for our construction to work. Note, $\mathcal{M}$ is some message space implicitly defined by pk_Π and pp_Π.

Definition 2 (IND-CCA2-Security). *A labeled encryption scheme Π is IND-CCA2-secure, if for any PPT adversary $\mathcal{A}$ there exists a negligible function ν such that:*

$$\Pr\left[\mathbf{Exp}^{\mathsf{IND\text{-}CCA2}}_{\mathcal{A},\Pi}(\lambda) = 1\right] \leq \nu(\lambda)$$

The corresponding experiment is depicted in Figure 1.

A suitable instantiation is CS-encryption with labels [21].

Aggregate Signatures Standard digital signatures allow the holder of a secret key $\mathsf{sk}_{\mathsf{sig}}$ to sign a message m, while with knowledge of the corresponding public key $\mathsf{pk}_{\mathsf{sig}}$ everyone can verify whether a given signature was actually endorsed by the signer, i.e., the holder of $\mathsf{pk}_{\mathsf{sig}}$ [50]. An aggregate signature scheme (Σ) allows to aggregate multiple signatures into a single (short) value [8].

As a side contribution, we introduce a new framework for aggregate signatures, where one can also de-aggregate signatures, and a novel *aggregate* uniqueness definition. This was, to the best of our knowledge, only considered in a white-box fashion in the context of BGLS-signatures [8], but was never formally defined in a black-box way, or only in some ad-hoc fashion [47].

Definition 3 (Aggregate Signatures). *An aggregate signature scheme Σ with explicit de-aggregation consists of six algorithms $\{\mathsf{PPGen}_\Sigma, \mathsf{KGen}_\Sigma, \mathsf{Sign}_\Sigma, \mathsf{AVerf}_\Sigma, \mathsf{Agg}_\Sigma, \mathsf{DAgg}_\Sigma\}$ such that:*

PPGen_Σ**.** *The algorithm PPGen_Σ outputs the public parameters*

$$\mathsf{pp}_\Sigma \leftarrow_r \mathsf{PPGen}_\Sigma(1^\lambda)$$

and we assume that pp_Σ contains 1^λ and is implicit input to all other algorithms.

KGen_Σ**.** *The algorithm KGen_Σ outputs the key pair of a signer:*

$$(\mathsf{sk}_\Sigma, \mathsf{pk}_\Sigma) \leftarrow_r \mathsf{KGen}_\Sigma(\mathsf{pp}_\Sigma)$$

Sign_Σ**.** *The algorithm Sign_Σ gets as input the secret key sk_Σ, and the message m to sign. It outputs a signature:*

$$\sigma \leftarrow \mathsf{Sign}_\Sigma(\mathsf{sk}_\Sigma, m)$$

AVerf_Σ**.** *The algorithm AVerf_Σ outputs a decision bit $d \in \{0,1\}$, indicating if an aggregate signature σ is valid, w.r.t. to a set $\{(\mathsf{pk}_{\Sigma,i}, m_i)\}$ of public keys/messages:*

$$d \leftarrow \mathsf{AVerf}_\Sigma(\{(\mathsf{pk}_{\Sigma,i}, m_i)\}, \sigma)$$

Agg_Σ**.** *The algorithm Agg_Σ receives a set of key/message tuples with corresponding aggregated signatures $\mathcal{S}_{\mathsf{agg}} = \{\{(\mathsf{pk}_{\Sigma,i,j}, m_{i,j})\}, \sigma_i\}$ to aggregate, where the sets $\{(\mathsf{pk}_{\Sigma,i,j}, m_{i,j})\}$ must be pair-wise disjoint, and each σ_i protects $\{(\mathsf{pk}_{\Sigma,i,j}, m_{i,j})\}$. In case $\mathcal{S}_{\mathsf{agg}} = \emptyset$, it returns $\perp$. Otherwise it outputs a new aggregate signature σ', protecting $\bigcup_i \{(\mathsf{pk}_{\Sigma,i,j}, m_{i,j})\}$.*[4]

$$\sigma'' \leftarrow \mathsf{Agg}_\Sigma(\{\{(\mathsf{pk}_{\Sigma,i,j}, m_{i,j})\}, \sigma_i\})$$

DAgg_Σ**.** *The algorithm DAgg_Σ gets a set $\{\{(\mathsf{pk}_{\Sigma,i,j}, m_{i,j})\}, \sigma_i\}$ of public key/message tuples, along with an aggregate signature σ_i (protecting $\{(\mathsf{pk}_{\Sigma,i,j}, m_{i,j})\}$), an additional set $\{(\mathsf{pk}_{\Sigma,k}, m_k)\}$, protected by an aggregate signature σ_k, and outputs an a new aggregate signature σ', which protects $\{(\mathsf{pk}_{\Sigma,k}, m_k)\} \setminus \bigcup_i \{(\mathsf{pk}_{\Sigma,i,j}, m_{i,j})\}$:*

$$\sigma' \leftarrow \mathsf{DAgg}_\Sigma(\{\{(\mathsf{pk}_{\Sigma,i,j}, m_{i,j})\}, \sigma_i\}, \{(\mathsf{pk}_{\Sigma,k}, m_k)\}, \sigma_k)$$

Clearly, an aggregate signature may also protect a single message, i.e., degenerate to a "normal" signature.

Moreover, we require the usual correctness properties to hold. Namely, honestly generated signatures verify, which must also be true for honestly generated aggregates. Likewise, honestly generated signatures stemming from de-aggregation must also verify.

From a security perspective, we require existential unforgeability under chosen-message attacks (eUNF-CMA), a strict form of uniqueness, correctness, and that a third party cannot remove signatures from an aggregate, if it does not know the signatures for the messages to be removed. We now formally define each of those properties.

Unforgeability requires that an adversary $\mathcal{A}$ cannot (except with negligible probability) come up with a signature for a message m^* for which the adversary did not see any signature before. As usual, the adversary $\mathcal{A}$ can adaptively query for signatures on messages of its own choice.

[4] We note that Boneh et al. require that each *message* appears at most once [8]. However, there are also mitigation strategies [4,8].

$\mathbf{Exp}^{\mathsf{eUNF\text{-}CMA}}_{\mathcal{A},\Sigma}(\lambda)$
  $\mathsf{pp}_\Sigma \leftarrow_r \mathsf{PPGen}_\Sigma(1^\lambda)$
  $(\mathsf{sk}_\Sigma, \mathsf{pk}_\Sigma) \leftarrow_r \mathsf{KGen}_\Sigma(\mathsf{pp}_\Sigma)$
  $\mathcal{Q} \leftarrow \emptyset$
  $(\{(\mathsf{pk}_i^*, m_i^*)\}, m^*, \sigma^*) \leftarrow_r \mathcal{A}^{\mathsf{Sign}'_\Sigma(\mathsf{sk}_\Sigma,\cdot)}(\mathsf{pk}_{\mathsf{sig}})$
    where $\mathsf{Sign}'_\Sigma(\mathsf{sk}_\Sigma, m)$:
      $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{m\}$
      return $\sigma \leftarrow \mathsf{Sign}_\Sigma(\mathsf{sk}_\Sigma, m)$
  return 1, if $\mathsf{AVerf}_\Sigma(\{(\mathsf{pk}_i^*, m_i^*)\} \cup \{(\mathsf{pk}_\Sigma, m^*)\}, \sigma^*) = 1 \ \wedge \ m^* \notin \mathcal{Q}$
  return 0

Fig. 2: Σ Unforgeability

Definition 4 (Σ Unforgeability). *We say a Σ scheme is unforgeable, if for every PPT adversary $\mathcal{A}$, there exists a negligible function ν such that:*

$$\Pr\left[\mathbf{Exp}^{\mathsf{eUNF\text{-}CMA}}_{\mathcal{A},\Sigma}(\lambda) = 1\right] \leq \nu(\lambda)$$

The corresponding experiment is depicted in Figure 2.

$\mathbf{Exp}^{\mathsf{Uniqueness}}_{\mathcal{A},\Sigma}(\lambda)$
  $\mathsf{pp}_\Sigma \leftarrow_r \mathsf{PPGen}_\Sigma(1^\lambda)$
  $(\{(\mathsf{pk}_i^*, m_i^*)\}, \sigma^*, \sigma'^*) \leftarrow_r \mathcal{A}(\mathsf{pp}_\Sigma)$
  return 1, if $\mathsf{AVerf}_\Sigma(\{(\mathsf{pk}_i^*, m_i^*)\}, \sigma^*) = 1 \wedge$
    $\mathsf{AVerf}_\Sigma(\{(\mathsf{pk}_i^*, m_i^*)\}, \sigma'^*) = 1 \ \wedge \ \sigma^* \neq \sigma'^*$
  return 0

Fig. 3: Σ Uniqueness

Uniqueness for aggregate signatures requires that for each set $\{(\mathsf{pk}_i, m_i)\}$ at most one signature can be found, even if all values can be adversarially generated. In contrast to Kuchta and Manulis [42], we require a slightly different uniqueness notion, i.e., the *complete* signature must be unique, and not only some part of it. Additionally, all values, but the public parameters, are explicitly generated by the adversary.

Definition 5 (Σ Uniqueness). *We say a Σ scheme is unique, if for every PPT adversary $\mathcal{A}$, there exists a negligible function ν such that:*

$$\Pr\left[\mathbf{Exp}^{\mathsf{Uniqueness}}_{\mathcal{A},\Sigma}(\lambda) = 1\right] \leq \nu(\lambda)$$

The corresponding experiment is depicted in Figure 3.

Remark 1. It should be obvious that uniqueness for aggregate signatures implies uniqueness for normal signatures. However, the converse is not true by the following argument: Append a random bit for every generated *aggregate*, and remove it before proceeding with the other algorithms, appending another random bit once they are finished, if the resulting signature is an aggregate.

Our final definition requires that an adversary cannot de-aggregate signatures from an aggregate signature, if the adversary never saw a signature (or aggregate) for the messages for which it tries to remove the signature from the aggregate. This resembles the k-element extraction assumption by Boneh et al. [8,20], but

for general aggregate signatures. To find out whether the adversary actually wins, we need to explicitly disregard aggregates which the adversary could find using non-avoidable transitivity relationships and adverserial signatures added "on top". We therefore use the algorithm Closure which finds all "trivial" relations.

Algorithm 1: Algorithm Closure

Input: $\mathcal{S} = \{(\mathsf{pk}_i, m_i)\}$, $\mathcal{D} = \{\{(\mathsf{pk}_j, m_j)\}\}$
Output: $\mathcal{D}' = \{\{(\mathsf{pk}_k, m_k)\}\}$

$l \leftarrow 0$
$l' \leftarrow 0$
$\mathcal{D}' \leftarrow \mathcal{D} \cup \mathcal{S}$
do
  $\mathcal{T} \leftarrow \mathcal{D}'$
  $l \leftarrow |\mathcal{T}|$
  foreach $\{(\mathsf{pk}_l, m_l)\} \in \mathcal{T}$ **do**
    foreach $\{(\mathsf{pk}_m, m_m)\} \neq \{(\mathsf{pk}_l, m_l)\} \in \mathcal{T}$ **do**
      if $\{(\mathsf{pk}_l, m_l)\} \subsetneq \{(\mathsf{pk}_m, m_m)\}$ **then**
        $\mathcal{D}' \leftarrow \mathcal{D}' \cup (\{\{(\mathsf{pk}_m, m_m)\} \setminus \{(\mathsf{pk}_l, m_l)\}\})$
  $l' \leftarrow |\mathcal{D}'|$
while $l \neq l'$
return $\mathcal{D}'$

$\mathbf{Exp}^{\mathsf{NoExtract}}_{\mathcal{A},\Sigma}(\lambda)$
  $\mathsf{pp}_\Sigma \leftarrow_r \mathsf{PPGen}_\Sigma(1^\lambda)$
  $\mathcal{M} \leftarrow \emptyset$
  $\mathcal{K} \leftarrow \emptyset$
  $(\{(\mathsf{pk}_i^*, m_i^*)\}, \sigma^*) \leftarrow_r \mathcal{A}^{\mathsf{KGen}'_\Sigma(\mathsf{pp}_\Sigma), \mathsf{Sign}'_\Sigma(\cdot)}(\mathsf{pp}_\Sigma)$
    where $\mathsf{KGen}'_\Sigma(\mathsf{pp}_\Sigma)$:
      $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow_r \mathsf{KGen}_\Sigma(\mathsf{pp}_\Sigma)$
      $\mathcal{K} \leftarrow \mathcal{K} \cup (\mathsf{sk}_i, \mathsf{pk}_i)$
      return pk_i
    where $\mathsf{Sign}'_\Sigma(\{(\mathsf{pk}_i, m_i)\})$:
      return $\perp$, if $\exists i : (\cdot, \mathsf{pk}_i) \notin \mathcal{K}$
      for all (pk_i, m_i), let $\sigma_i \leftarrow \mathsf{Sign}_\Sigma(\mathsf{sk}_i, m_i)$
      $\sigma_i'' \leftarrow \mathsf{Agg}_\Sigma(\{\{(\mathsf{pk}_i, m_i\}, \sigma_i)\})$
      $\mathcal{M} \leftarrow \text{Closure}(\{(\mathsf{pk}_i, m_i)\}, \mathcal{M})$
      return σ_i''
  $\mathcal{C} \leftarrow \{(\mathsf{pk}_i^*, m_i^*) \mid (\mathsf{pk}_i^*, \cdot) \in \mathcal{K}\}$
  return 1, if $\mathsf{AVerf}_\Sigma(\{(\mathsf{pk}_i^*, m_i^*)\}, \sigma^*) = 1 \ \wedge \ \mathcal{C} \notin \mathcal{M}$
  return 0

Fig. 4: Σ Extraction Secure

Definition 6 (Σ No-Extraction). *We say a Σ scheme provides no-extraction, if for every PPT adversary $\mathcal{A}$, there exists a negligible function ν such that:*

$$\Pr\left[\mathbf{Exp}^{\mathsf{NoExtract}}_{\mathcal{A},\Sigma}(\lambda) = 1\right] \leq \nu(\lambda)$$

The corresponding experiment is depicted in Figure 4, where the algorithm Closure *is given in Algorithm 1.*

We additionally require that, regardless of the message m, each signature is of constant size. This can easily be achieved by hashing the message m using a collision-resistant hash-function prior to signing. A suitable instantiation is BGLS-signatures [8], if one can enforce distinct messages (what we do).

Definition 7 (Pseudo-Random Functions). *A pseudo-random function* PRF *consists of two algorithms* $\{\mathsf{KGen}_{\mathsf{PRF}}, \mathsf{Eval}_{\mathsf{PRF}}\}$ *such that:*

$\mathsf{KGen}_{\mathsf{PRF}}$**.** *The algorithm* $\mathsf{KGen}_{\mathsf{PRF}}$ *outputs a function key* $x \in \{0,1\}^\lambda$*, where* λ *is the security parameter:*

$$x \leftarrow_r \mathsf{KGen}_{\mathsf{PRF}}(1^\lambda)$$

$\mathsf{Eval}_{\mathsf{PRF}}$**.** *The algorithm* $\mathsf{Eval}_{\mathsf{PRF}}$ *gets as input the secret key* x *and a point* $p \in \{0,1\}^\lambda$ *to evaluate. It outputs a new point* $p' \in \{0,1\}^\lambda$*:*

$$p' \leftarrow \mathsf{Eval}_{\mathsf{PRF}}(x, p)$$

Security For security, it is required that PRF is actually pseudo-random.

$\mathbf{Exp}^{\mathsf{PR}}_{\mathcal{A},\mathsf{PRF}}(\lambda)$
  $x \leftarrow_r \mathsf{KGen}_{\mathsf{PRF}}(1^\lambda)$
  $b \leftarrow_r \{0,1\}$
  $f \leftarrow_r F_\lambda$
  $a \leftarrow_r \mathcal{A}^{\mathsf{Eval}'_{\mathsf{PRF}}(x,\cdot)}(1^\lambda)$
    where oracle $\mathsf{Eval}'_{\mathsf{PRF}}(x,p)$:
      return $\perp$, if $p \notin \{0,1\}^\lambda$
      if $b = 0$, return $\mathsf{Eval}_{\mathsf{PRF}}(x,p)$
      return $f(p)$
  return 1, if $a = b$
  return 0

Fig. 5: PRF Pseudo-Randomness

Definition 8 (PRF Pseudo-Randomness). *A pseudo-random generator* PRF *is called pseudo-random, if for any PPT adversary* $\mathcal{A}$ *there exists a negligible function* ν *such that:*

$$\left|\Pr[\mathbf{Exp}^{\mathsf{PR}}_{\mathcal{A},\mathsf{PRF}}(\lambda) = 1] - \frac{1}{2}\right| \leq \nu(\lambda)$$

The corresponding experiment is depicted in Figure 5, where $F_\lambda = \{f : \{0,1\}^\lambda \to \{0,1\}^\lambda\}$ *is the set of all functions* f *mapping a value* $v \in \{0,1\}^\lambda$ *to another value* $v' \in \{0,1\}^\lambda$*.*

Sanitizable Signatures Subsequently, we restate the definitions of (standard) SSs [3,11,41]. To recap, a SS allows a semi-trusted third party, named the sanitizer, to alter signer-chosen blocks to arbitrary bit-strings. The sanitizer holds its own key-pair and can be held accountable, if it sanitizes a message.

The following framework is essentially the one given by Camenisch et al. [16], which is itself based on existing work [11]. However, some additional notation is required beforehand. The variable $\mathbb{A}^{\mathsf{SS}}$ contains the set of indices of the modifiable blocks, as well as ℓ, denoting the total number of blocks in the message m. For example, let $\mathbb{A}^{\mathsf{SS}} = (\{1,2,4\},4)$. Then, m must contain four blocks ($\ell = 4$) and all but the third are admissible. Note, $\mathbb{A}^{\mathsf{SS}}$ can be encoded in a length-invariant way by using a sequence of bits, e.g., $(1,1,0,1)$ for $\mathbb{A}^{\mathsf{SS}} = (\{1,2,4\},4)$. The variable $\mathbb{M}^{\mathsf{SS}}$ is a set containing pairs $(i, m^{i\prime})$ for those blocks that are modified,

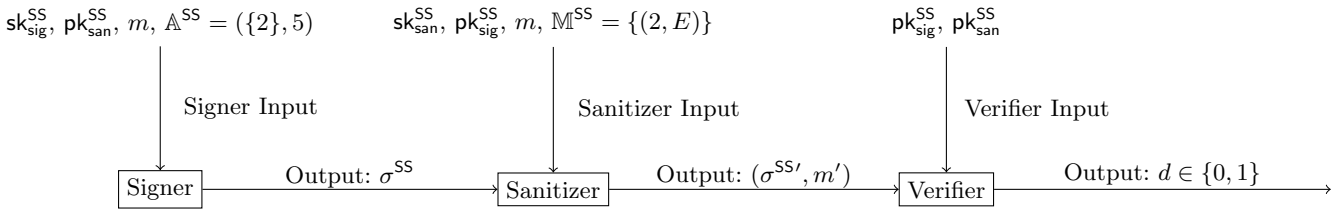


Fig. 6: Example workflow of an SS. The message m is set to (H, A, L, L, O) and is sanitized $m' = (H, E, L, L, O)$

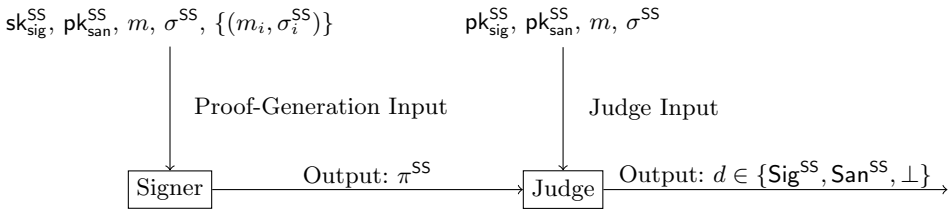


Fig. 7: Additional algorithms $\mathsf{Proof}^{\mathsf{SS}}$ and $\mathsf{Judge}^{\mathsf{SS}}$ for an accountable SS

meaning that m^i is replaced by $m^{i\prime}$. We use the shorthand notation $m' \leftarrow \mathbb{M}^{\mathsf{SS}}(m)$ to denote the result of this replacement, while $\mathbb{M}^{\mathsf{SS}} \prec (m, \mathbb{A}^{\mathsf{SS}})$ means that $\mathbb{M}^{\mathsf{SS}}$ is a valid modification instruction w.r.t. m and $\mathbb{A}^{\mathsf{SS}}$. Likewise, we use $\mathbb{A}^{\mathsf{SS}} \prec m$ to denote that $\mathbb{A}^{\mathsf{SS}}$ is valid description of the admissible blocks w.r.t. m. An example workflow is depicted in Figure 6 and Figure 7. Both are derived from Bilzhause et al. [6].

Definition 9 (Sanitizable Signatures). *A sanitizable signature scheme* SS *consists of the following eight ppt algorithms* $\{\mathsf{PPGen}^{\mathsf{SS}}, \mathsf{KG}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{KG}^{\mathsf{SS}}_{\mathsf{san}}, \mathsf{Sign}^{\mathsf{SS}}, \mathsf{Verify}^{\mathsf{SS}}, \mathsf{Sanit}^{\mathsf{SS}}, \mathsf{Proof}^{\mathsf{SS}}, \mathsf{Judge}^{\mathsf{SS}}\}$ *such that:*

$\mathsf{PPGen}^{\mathsf{SS}}$**.** *The algorithm* $\mathsf{PPGen}^{\mathsf{SS}}$ *generates the public parameters:*

$$\mathsf{pp}_{\mathsf{SS}} \leftarrow_r \mathsf{PPGen}^{\mathsf{SS}}(1^\lambda)$$

We assume that $\mathsf{pp}_{\mathsf{SS}}$ *is implicitly input to all other algorithms.*

$\mathsf{KG}^{\mathsf{SS}}_{\mathsf{sig}}$**.** *The algorithm* $\mathsf{KG}^{\mathsf{SS}}_{\mathsf{sig}}$ *generates the key pair of the signer:*

$$(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}) \leftarrow_r \mathsf{KG}^{\mathsf{SS}}_{\mathsf{sig}}(\mathsf{pp}_{\mathsf{SS}})$$

$\mathsf{KG}^{\mathsf{SS}}_{\mathsf{san}}$**.** *The algorithm* $\mathsf{KG}^{\mathsf{SS}}_{\mathsf{san}}$ *generates the key pair of the sanitizer:*

$$(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}) \leftarrow_r \mathsf{KG}^{\mathsf{SS}}_{\mathsf{san}}(\mathsf{pp}_{\mathsf{SS}})$$

$\mathsf{Sign}^{\mathsf{SS}}$**.** *The algorithm* $\mathsf{Sign}^{\mathsf{SS}}$ *generates a signature* σ^{SS} *on input of the public key* $\mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}$, $\mathbb{A}^{\mathsf{SS}}$, *a message* m *and* $\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}}$*:*

$$\sigma^{\mathsf{SS}} \leftarrow_r \mathsf{Sign}^{\mathsf{SS}}(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}, m, \mathbb{A}^{\mathsf{SS}})$$

$\mathsf{Verify}^{\mathsf{SS}}$**.** *The algorithm* $\mathsf{Verify}^{\mathsf{SS}}$ *verifies a signature* σ^{SS}*, i.e., outputs a decision* $d \in \{0, 1\}$ *w.r.t.* $\mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}$, $\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}$ *and a message* m*:*

$$d \leftarrow \mathsf{Verify}^{\mathsf{SS}}(\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}, m, \sigma^{\mathsf{SS}})$$

$\mathsf{Sanit}^{\mathsf{SS}}$**.** *The algorithm* $\mathsf{Sanit}^{\mathsf{SS}}$ *generates a sanitized signature* $\sigma^{\mathsf{SS}\prime}$ *on input* $\mathsf{sk}^{\mathsf{SS}}_{\mathsf{san}}$, $\mathbb{A}^{\mathsf{SS}}$, *a message* m *and* $\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}$*:*

$$(m', \sigma^{\mathsf{SS}\prime}) \leftarrow_r \mathsf{Sanit}^{\mathsf{SS}}(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, m, \sigma^{\mathsf{SS}}, \mathbb{M}^{\mathsf{SS}})$$

$\mathbf{Exp}^{\mathsf{Unforgeability}}_{\mathcal{A},\mathsf{SS}}(\lambda)$
$\mathsf{pp}_{\mathsf{SS}} \leftarrow_r \mathsf{PPGen}^{\mathsf{SS}}(1^\lambda)$
$(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}) \leftarrow_r \mathsf{KG}^{\mathsf{SS}}_{\mathsf{sig}}(\mathsf{pp}_{\mathsf{SS}})$
$(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}) \leftarrow_r \mathsf{KG}^{\mathsf{SS}}_{\mathsf{san}}(\mathsf{pp}_{\mathsf{SS}})$
$(m^*, \sigma^{\mathsf{SS}*}) \leftarrow_r \mathcal{A}^{\mathsf{Sign}^{\mathsf{SS}}(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}},\cdot,\cdot,\cdot),\mathsf{Sanit}^{\mathsf{SS}}(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{san}},\cdot,\cdot,\cdot,\cdot)}_{\mathsf{Proof}^{\mathsf{SS}}(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}},\cdot,\cdot,\cdot,\cdot)}(\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}})$
for $i = 1, 2, \ldots, q$ let $(\mathsf{pk}^{\mathsf{SS}}_{\mathsf{san},i}, m_i, \mathbb{A}^{\mathsf{SS}}_i)$ and σ^{SS}_i
index the queries/answers to/from $\mathsf{Sign}^{\mathsf{SS}}$
for $j = 1, 2, \ldots, q'$ let $(\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig},j}, m_j, \sigma^{\mathsf{SS}}_j, \mathbb{M}_j)$ and $(m'_j, \sigma^{\mathsf{SS}\prime}_j)$
index the queries/answers to/from $\mathsf{Sanit}^{\mathsf{SS}}$
return 1, if $\mathsf{Verify}^{\mathsf{SS}}(\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}, m^*, \sigma^{\mathsf{SS}*}) = 1 \wedge$
$\forall i \in [1..q] : (\mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}, m^*, \sigma^{\mathsf{SS}*}) \neq (\mathsf{pk}^{\mathsf{SS}}_{\mathsf{san},i}, m_i, \sigma^{\mathsf{SS}}_i) \wedge$
$\forall j \in [1..q'] : (\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, m^*, \sigma^{\mathsf{SS}*}) \neq (\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig},j}, m'_j, \sigma^{\mathsf{SS}\prime}_j)$
return 0

Fig. 8: SS Unforgeability

$\mathsf{Proof}^{\mathsf{SS}}$. *The algorithm* $\mathsf{Proof}^{\mathsf{SS}}$ *outputs a proof* π^{SS} *on input* m, σ^{SS}, $\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}}$, $\mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}$ *and a set of polynomially many additional signature/message pairs* $\{(\sigma^{\mathsf{SS}}_i, m_i)\}$. *The proof* π^{SS} *is used by the next algorithm to pinpoint the accountable party for a given signature:*

$$\pi^{\mathsf{SS}} \leftarrow_r \mathsf{Proof}^{\mathsf{SS}}(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}, m, \sigma^{\mathsf{SS}}, \{(\sigma^{\mathsf{SS}}_i, m_i)\})$$

$\mathsf{Judge}^{\mathsf{SS}}$. *The algorithm* $\mathsf{Judge}^{\mathsf{SS}}$ *outputs a decision* $d \in \{\mathsf{Sig}^{\mathsf{PS}}, \mathsf{San}^{\mathsf{PS}}, \perp\}$ *indicating whether the message/signature pair has been created by the signer, or the sanitizer:*

$$d \leftarrow \mathsf{Judge}^{\mathsf{SS}}(\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}, m, \sigma^{\mathsf{SS}}, \pi^{\mathsf{SS}})$$

Correctness was already specified by Brzuska et al. [11].

SSs Security Definitions We now introduce the security properties required. These are the ones given by Beck et al. [3], but altered for the used notation, already incorporating the strong definitions by Krenn et al. [41], but a stronger notion of invisibility, where the adversary is now able to query arbitrary signatures to the sanitization oracle. Moreover, we do neither consider unlinkability nor non-interactive public-accountability, as it depends on the context whether these properties are required [3,16]. However, non-interactive public-accountability is easy to achieve, e.g., by signing the resulting signature again [14].

Unforgeability This definition requires that an adversary $\mathcal{A}$ not having any secret keys is not able to produce any new valid signature σ^* on a message m^* which it has never seen, even if $\mathcal{A}$ has full oracle access.

Definition 10 (Unforgeability). *An* SS *is unforgeable, if for any PPT adversary* $\mathcal{A}$ *there exists a negligible function* ν *such that*

$$\Pr[\mathbf{Exp}^{\mathsf{Unforgeability}}_{\mathcal{A},\mathsf{SS}}(\lambda) = 1] \leq \nu(\lambda)\,,$$

where the corresponding experiment is defined in Figure 8.

Immutability This definition prohibits that an adversary $\mathcal{A}$ can generate a verifying signature $\sigma^{\mathsf{SS}*}$ for a message m^* not derivable from the signatures given by an honest signer, even if it can generate the sanitizer's key pair.

Definition 11 (Immutability). *An* SS *is immutable, if for any PPT adversary* $\mathcal{A}$ *there exists a negligible function* ν *such that*

$$\Pr[\mathbf{Exp}^{\mathsf{Immutability}}_{\mathcal{A},\mathsf{SS}}(\lambda) = 1] \leq \nu(\lambda)\,,$$

where the corresponding experiment is defined in Figure 9.

$\mathbf{Exp}^{\mathsf{Immutability}}_{\mathcal{A},\mathsf{SS}}(\lambda)$
$\mathsf{pp}_{\mathsf{SS}} \leftarrow_r \mathsf{PPGen}^{\mathsf{SS}}(1^\lambda)$
$(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}) \leftarrow_r \mathsf{KG}^{\mathsf{SS}}_{\mathsf{sig}}(\mathsf{pp}_{\mathsf{SS}})$
$(m^*, \sigma^{\mathsf{SS}*}, \mathsf{pk}^{\mathsf{SS}\,*}_{\mathsf{san}}) \leftarrow_r \mathcal{A}^{\mathsf{Sign}^{\mathsf{SS}}(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}},\cdot,\cdot,\cdot),\mathsf{Proof}^{\mathsf{SS}}(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}},\cdot,\cdot,\cdot,\cdot)}(\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}})$
for $i = 1, 2, \ldots, q$ let $(\mathsf{pk}^{\mathsf{SS}}_{\mathsf{san},i}, m_i, \mathbb{A}^{\mathsf{SS}}_i)$
index the queries to $\mathsf{Sign}^{\mathsf{SS}}$
return 1, if $\mathsf{Verify}^{\mathsf{SS}}(\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}\,*}_{\mathsf{san}}, m^*, \sigma^{\mathsf{SS}*}) = 1 \wedge$
$\forall i \in [1..q] : (\mathsf{pk}^{\mathsf{SS}\,*}_{\mathsf{san}} \neq \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san},i} \vee$
$m^* \notin \{\mathbb{M}(m_i) \mid \mathbb{M} \text{ with } \mathbb{M}^{\mathsf{SS}}(\mathbb{A}^{\mathsf{SS}}_i) = 1\})$
return 0

Fig. 9: SS Immutability

$\mathbf{Exp}^{\mathsf{Privacy}}_{\mathcal{A},\mathsf{SS}}(\lambda)$
$\mathsf{pp}_{\mathsf{SS}} \leftarrow_r \mathsf{PPGen}^{\mathsf{SS}}(1^\lambda)$
$(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}) \leftarrow_r \mathsf{KG}^{\mathsf{SS}}_{\mathsf{sig}}(\mathsf{pp}_{\mathsf{SS}})$
$(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}) \leftarrow_r \mathsf{KG}^{\mathsf{SS}}_{\mathsf{san}}(\mathsf{pp}_{\mathsf{SS}})$
$b \leftarrow_r \{0,1\}$
$a \leftarrow_r \mathcal{A}^{\mathsf{Sign}^{\mathsf{SS}}(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}},\cdot,\cdot,\cdot),\mathsf{Sanit}^{\mathsf{SS}}(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{san}},\cdot,\cdot,\cdot,\cdot)}_{\mathsf{Proof}^{\mathsf{SS}}(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}},\cdot,\cdot,\cdot,\cdot),\mathsf{LoRSan}(\cdot,\cdot,\cdot,\cdot,\cdot,\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}},\mathsf{sk}^{\mathsf{SS}}_{\mathsf{san}},b)}(\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}})$
where $\mathsf{LoRSan}(m_0, m_1, \mathbb{M}^{\mathsf{SS}}_0, \mathbb{M}^{\mathsf{SS}}_1, \mathbb{A}^{\mathsf{SS}}, \mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{sk}^{\mathsf{SS}}_{\mathsf{san}}, b)$:
return $\perp$, if $\mathbb{M}^{\mathsf{SS}}_0 \not\prec (m_0, \mathbb{A}^{\mathsf{SS}}) \vee \mathbb{M}^{\mathsf{SS}}_1 \not\prec (m_1, \mathbb{A}^{\mathsf{SS}}) \vee$
$\mathbb{M}^{\mathsf{SS}}_0(m_0) \neq \mathbb{M}^{\mathsf{SS}}_1(m_1) \vee \mathbb{A}^{\mathsf{SS}} \not\prec m_0 \vee \mathbb{A}^{\mathsf{SS}} \not\prec m_1$
$\sigma^{\mathsf{SS}} \leftarrow_r \mathsf{Sign}^{\mathsf{SS}}(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}, m_b, \mathbb{A}^{\mathsf{SS}})$
return $(m', \sigma^{\mathsf{SS}\prime}) \leftarrow_r \mathsf{Sanit}^{\mathsf{SS}}(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, m_b, \sigma^{\mathsf{SS}}, \mathbb{M}^{\mathsf{SS}}_b)$
return 1, if $a = b$
return 0

Fig. 10: SS Privacy

Privacy This definition prohibits that an adversary $\mathcal{A}$ can learn anything about sanitized parts. This is similar to the definition of standard encryption schemes.

Definition 12 (Privacy). *An* SS *is private, if for any PPT adversary $\mathcal{A}$ there exists a negligible function ν such that*

$$\left|\Pr[\mathbf{Exp}^{\mathsf{Privacy}}_{\mathcal{A},\mathsf{SS}}(\lambda)] - 1/2\right| \leq \nu(\lambda)\,,$$

where the corresponding experiment is defined in Figure 10.

Transparency This definition prohibits that an adversary $\mathcal{A}$ does not learn whether a signature σ^{SS} was generated through $\mathsf{Sign}^{\mathsf{SS}}$ or $\mathsf{Sanit}^{\mathsf{SS}}$. We stress that the adversary cannot query signatures obtained by the $\mathsf{Sign}/\mathsf{Sanit}$-oracle to the $\mathsf{Proof}^{\mathsf{SS}\prime}$-oracle to avoid trivial attacks.

Definition 13 (Transparency). *An* SS *is transparent, if for any PPT adversary $\mathcal{A}$ there exists a negligible function ν such that*

$$\left|\Pr[\mathbf{Exp}^{\mathsf{Transparency}}_{\mathcal{A},\mathsf{SS}}(\lambda)] - 1/2\right| \leq \nu(\lambda)\,,$$

where the corresponding experiment is defined in Figure 11.

Signer-Accountability Signer-accountability prohibits that an adversary can generate a bogus proof that makes $\mathsf{Judge}^{\mathsf{SS}}$ decide that the sanitizer is responsible for a given signature/message pair $(m^*, \sigma^{\mathsf{SS}*})$, but the sanitizer has never generated this pair. This is even true, if the adversary can generate the signer's key pair.

$\mathbf{Exp}^{\mathsf{Transparency}}_{\mathcal{A},\mathsf{SS}}(\lambda)$
$\mathsf{pp}_{\mathsf{SS}} \leftarrow_r \mathsf{PPGen}^{\mathsf{SS}}(1^\lambda)$
$(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}) \leftarrow_r \mathsf{KG}^{\mathsf{SS}}_{\mathsf{sig}}(\mathsf{pp}_{\mathsf{SS}})$
$(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}) \leftarrow_r \mathsf{KG}^{\mathsf{SS}}_{\mathsf{san}}(\mathsf{pp}_{\mathsf{SS}})$
$b \leftarrow_r \{0,1\}$
$\mathcal{Q} \leftarrow \emptyset$
$a \leftarrow_r \mathcal{A}^{\mathsf{Sign}^{\mathsf{SS}}(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}},\cdot,\cdot,\cdot),\mathsf{Sanit}^{\mathsf{SS}}(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{san}},\cdot,\cdot,\cdot,\cdot)}_{\mathsf{Proof}^{\mathsf{SS}\prime}(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}},\cdot,\cdot,\cdot,\cdot),\mathsf{Sign/Sanit}(\cdot,\cdot,\cdot,\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}},\mathsf{sk}^{\mathsf{SS}}_{\mathsf{san}},b)}(\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}})$
where $\mathsf{Proof}^{\mathsf{SS}\prime}(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}}, m, \sigma^{\mathsf{SS}}, \{(m_i, \sigma^{\mathsf{SS}}_i) \mid i \in \mathbb{N}\})$:
return $\perp$, if $\mathsf{pk}^{\mathsf{SS}\prime}_{\mathsf{san}} = \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}} \wedge$
$((m, \sigma^{\mathsf{SS}}) \in \mathcal{Q} \ \vee \ \mathcal{Q} \cap \{(m_i, \sigma^{\mathsf{SS}}_i)\} \neq \emptyset)$
return $\mathsf{Proof}^{\mathsf{SS}}(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}\prime}_{\mathsf{san}}, m, \sigma^{\mathsf{SS}}, \{(m_i, \sigma^{\mathsf{SS}}_i)\})$
where $\mathsf{Sign/Sanit}(m, \mathbb{M}^{\mathsf{SS}}, \mathbb{A}^{\mathsf{SS}}, \mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{sk}^{\mathsf{SS}}_{\mathsf{san}})$:
$\sigma^{\mathsf{SS}} \leftarrow_r \mathsf{Sign}^{\mathsf{SS}}(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}, m, \mathbb{A}^{\mathsf{SS}})$
$(m', \sigma^{\mathsf{SS}\prime}) \leftarrow_r \mathsf{Sanit}^{\mathsf{SS}}(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, m, \sigma^{\mathsf{SS}}, \mathbb{M}^{\mathsf{SS}})$
if $b = 1$:
$\sigma^{\mathsf{SS}\prime} \leftarrow_r \mathsf{Sign}^{\mathsf{SS}}(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}, m', \mathbb{A}^{\mathsf{SS}})$
If $\sigma^{\mathsf{SS}\prime} \neq \perp$, set $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m', \sigma^{\mathsf{SS}\prime})\}$
return $(m', \sigma^{\mathsf{SS}\prime})$
return 1, if $a = b$
return 0

Fig. 11: SS Transparency

$\mathbf{Exp}^{\mathsf{SigAccountability}}_{\mathcal{A},\mathsf{SS}}(\lambda)$
$\mathsf{pp}_{\mathsf{SS}} \leftarrow_r \mathsf{PPGen}^{\mathsf{SS}}(1^\lambda)$
$(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}) \leftarrow_r \mathsf{KG}^{\mathsf{SS}}_{\mathsf{san}}(\mathsf{pp}_{\mathsf{SS}})$
$(\mathsf{pk}^{\mathsf{SS}*}_{\mathsf{sig}}, \pi^{\mathsf{SS}*}, m^*, \sigma^{\mathsf{SS}*}) \leftarrow_r \mathcal{A}^{\mathsf{Sanit}^{\mathsf{SS}}(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{san}},\cdot,\cdot,\cdot,\cdot)}(\mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}})$
for $i = 1, 2, \ldots, q$ let $(m'_i, \sigma^{\mathsf{SS}\prime}_i)$ and $(m_i, \mathbb{M}^{\mathsf{SS}}_i, \sigma^{\mathsf{SS}}_i, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig},i})$
index the answers/queries from/to $\mathsf{Sanit}^{\mathsf{SS}}$
return 1, if $\mathsf{Verify}^{\mathsf{SS}}(\mathsf{pk}^{\mathsf{SS}*}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}, m^*, \sigma^{\mathsf{SS}*}) = 1 \wedge$
$\forall i \in [1..q] : (\mathsf{pk}^{\mathsf{SS}*}_{\mathsf{sig}}, m^*, \sigma^{\mathsf{SS}*}) \neq (\mathsf{pk}_{\mathsf{sig},i}, m'_i, \sigma^{\mathsf{SS}\prime}_i) \wedge$
$\mathsf{Judge}^{\mathsf{SS}}(\mathsf{pk}^{\mathsf{SS}*}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}, m^*, \sigma^{\mathsf{SS}*}, \pi^{\mathsf{SS}*}) = \mathsf{San}^{\mathsf{SS}}$
return 0

Fig. 12: SS Signer-Accountability

Definition 14 (Signer-Accountability). *An* SS *is signer-accountable, if for any PPT adversary* $\mathcal{A}$ *there exists a negligible function* ν *such that*

$$\Pr[\mathbf{Exp}^{\mathsf{SigAccountability}}_{\mathcal{A},\mathsf{SS}}(\lambda) = 1] \leq \nu(\lambda)\,,$$

where the corresponding experiment is defined in Figure 12.

Sanitizer-Accountability Sanitizer-accountability prohibits that an adversary can generate a bogus signature/message pair $(m^*, \sigma^{\mathsf{SS}*})$ that makes $\mathsf{Proof}^{\mathsf{SS}}$ outputs a (honestly generated) generated proof π^{SS} which points to the signer, but $(m^*, \sigma^{\mathsf{SS}*})$ has never been generated by the signer. This is even true, if the adversary can generate the sanitizer's key pair.

Definition 15 (Sanitizer-Accountability). *An* SS *is sanitizer-accountable, if for any PPT adversary* $\mathcal{A}$ *there exists a negligible function* ν *such that*

$$\Pr[\mathbf{Exp}^{\mathsf{SanAccountability}}_{\mathcal{A},\mathsf{SS}}(\lambda) = 1] \leq \nu(\lambda)\,,$$

where the corresponding experiment is defined in Figure 13.

$\mathbf{Exp}^{\mathsf{SanAccountability}}_{\mathcal{A},\mathsf{SS}}(\lambda)$
$\mathsf{pp}_{\mathsf{SS}} \leftarrow_r \mathsf{PPGen}^{\mathsf{SS}}(1^\lambda)$
$(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}) \leftarrow_r \mathsf{KG}^{\mathsf{SS}}_{\mathsf{sig}}(\mathsf{pp}_{\mathsf{SS}})$
$(m^*, \sigma^{\mathsf{SS}*}, \mathsf{pk}^{\mathsf{SS}\,*}_{\mathsf{san}}) \leftarrow_r \mathcal{A}^{\mathsf{Sign}^{\mathsf{SS}}(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}},\cdot,\cdot,\cdot),\mathsf{Proof}^{\mathsf{SS}}(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}},\cdot,\cdot,\cdot,\cdot)}(\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}})$
for $i = 1, 2, \ldots, q$ let $(\mathsf{pk}^{\mathsf{SS}}_{\mathsf{san},i}, m_i, \mathbb{A}^{\mathsf{SS}}_i)$ and σ^{SS}_i
index the queries/answers to/from $\mathsf{Sign}^{\mathsf{SS}}$
$\pi^{\mathsf{SS}} \leftarrow_r \mathsf{Proof}^{\mathsf{SS}}(\mathsf{sk}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}\,*}_{\mathsf{san}}, m^*, \sigma^{\mathsf{SS}*}, \{(m_i, \sigma^{\mathsf{SS}}_i) \mid 0 < i \leq q\})$
return 1, if $\mathsf{Verify}^{\mathsf{SS}}(\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}\,*}_{\mathsf{san}}, m^*, \sigma^{\mathsf{SS}*}) = 1 \wedge$
$\forall i \in [1..q] : (\mathsf{pk}^{\mathsf{SS}\,*}_{\mathsf{san}}, m^*, \sigma^{\mathsf{SS}*}) \neq (\mathsf{pk}^{\mathsf{SS}}_{\mathsf{san},i}, m_i, \sigma^{\mathsf{SS}}_i) \wedge$
$\mathsf{Judge}^{\mathsf{SS}}(\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}\,*}_{\mathsf{san}}, m^*, \sigma^{\mathsf{SS}*}, \pi^{\mathsf{SS}}) = \mathsf{Sig}^{\mathsf{SS}}$
return 0

Fig. 13: SS Sanitizer-Accountability

$\mathbf{Exp}^{\mathsf{Invisibility}}_{\mathcal{A},\mathsf{SS}}(\lambda)$
$\mathsf{pp}_{\mathsf{SS}} \leftarrow_r \mathsf{PPGen}^{\mathsf{SS}}(1^\lambda)$
$(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}) \leftarrow_r \mathsf{KG}^{\mathsf{SS}}_{\mathsf{sig}}(\mathsf{pp}_{\mathsf{SS}})$
$(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}) \leftarrow_r \mathsf{KG}^{\mathsf{SS}}_{\mathsf{san}}(\mathsf{pp}_{\mathsf{SS}})$
$b \leftarrow_r \{0, 1\}$
$\mathcal{Q} \leftarrow \emptyset$
$a \leftarrow_r \mathcal{A}^{\mathsf{Sanit}^{\mathsf{SS}\prime}(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{san}},\cdot,\cdot,\cdot,\cdot),\mathsf{Proof}^{\mathsf{SS}\prime}(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}},\cdot,\cdot,\cdot,\cdot),\mathsf{LoRADM}(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}},\cdot,\cdot,\cdot,\cdot,b)}(\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}})$
where $\mathsf{LoRADM}(\mathsf{sk}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}\,\prime}_{\mathsf{san}}, m, \mathbb{A}^{\mathsf{SS}}_0, \mathbb{A}^{\mathsf{SS}}_1, b)$:
return $\perp$, if $\mathbb{A}^{\mathsf{SS}}_0 \not\preccurlyeq m \wedge \mathbb{A}^{\mathsf{SS}}_1 \not\preccurlyeq m$
return $\perp$, if $\mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}} \neq \mathsf{pk}^{\mathsf{SS}\,\prime}_{\mathsf{san}} \ \wedge \ \mathbb{A}^{\mathsf{SS}}_0 \neq \mathbb{A}^{\mathsf{SS}}_1$
$\sigma^{\mathsf{SS}} \leftarrow_r \mathsf{Sign}^{\mathsf{SS}}(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}\,\prime}_{\mathsf{san}}, m, \mathbb{A}^{\mathsf{SS}}_b)$
if $\mathsf{pk}^{\mathsf{SS}\,\prime}_{\mathsf{san}} = \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}$, let $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, \sigma^{\mathsf{SS}}, \mathbb{A}^{\mathsf{SS}}_0 \cap \mathbb{A}^{\mathsf{SS}}_1)\}$
return σ^{SS}
where $\mathsf{Sanit}^{\mathsf{SS}\prime}(\mathsf{pk}^{\mathsf{SS}\prime}_{\mathsf{sig}}, \mathsf{sk}^{\mathsf{SS}}_{\mathsf{san}}, m, \mathbb{M}^{\mathsf{SS}}, \sigma^{\mathsf{SS}})$:
return $\perp$, if $\mathsf{pk}^{\mathsf{SS}\prime}_{\mathsf{sig}} = \mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}} \wedge \exists (m, \sigma^{\mathsf{SS}}, \mathbb{A}^{\mathsf{SS}}) \in \mathcal{Q} : \mathbb{M}^{\mathsf{SS}} \not\preccurlyeq (m, \mathbb{A}^{\mathsf{SS}})$
$(m', \sigma^{\mathsf{SS}\prime}) \leftarrow_r \mathsf{Sanit}^{\mathsf{SS}}(\mathsf{pk}^{\mathsf{SS}\prime}_{\mathsf{sig}}, \mathsf{sk}^{\mathsf{SS}}_{\mathsf{san}}, m, \mathbb{M}^{\mathsf{SS}}, \sigma^{\mathsf{SS}})$
if $\mathsf{pk}^{\mathsf{SS}\prime}_{\mathsf{sig}} = \mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}} \wedge \exists (m, \sigma^{\mathsf{SS}}, \mathbb{A}^{\mathsf{SS}\prime}) \in \mathcal{Q} : \mathbb{M}^{\mathsf{SS}} \prec (m, \mathbb{A}^{\mathsf{SS}\prime})$,
$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m', \sigma^{\mathsf{SS}\prime}, \mathbb{A}^{\mathsf{SS}\prime})\}$
return $(m', \sigma^{\mathsf{SS}\prime})$
return 1, if $a = b$
return 0

Fig. 14: SS Invisibility

Invisibility Depending on the context, an additional privacy guarantee may be required. In particular, invisibility prohibits that an outsider holding no secret keys can decide which parts of a message m are sanitizable. Note, the signing oracle can be simulated using the LoRADM oracle and setting $\mathbb{A}^{\mathsf{SS}}_0 = \mathbb{A}^{\mathsf{SS}}_1$. The notation $\mathbb{A}^{\mathsf{SS}}_0 \cap \mathbb{A}^{\mathsf{SS}}_1$ means that only those indices are admissible which are admissible in $\mathbb{A}^{\mathsf{SS}}_0$ and $\mathbb{A}^{\mathsf{SS}}_1$.

We stress, however, that we introduce a slightly stronger variant than discussed in prior work [3]; Namely, we allow that the adversary can query *any* signature to $\mathsf{Sanit}^{\mathsf{SS}\prime}$, and not only the ones generated honestly. In particular, now only if the signature was created by one of the oracles, we enforce the restriction $\mathbb{A}^{\mathsf{SS}}_0 \cap \mathbb{A}^{\mathsf{SS}}_1$. We stress, however, that all strongly invisible constructions proposed so far are also unforgeable, and thus such a signature can never be generated by the adversary.

Definition 16 (Invisibility). *An* SS *is invisible, if for any PPT adversary* $\mathcal{A}$ *there exists a negligible function* ν *such that*

$$\left|\Pr[\mathbf{Exp}^{\mathsf{Invisibility}}_{\mathcal{A},\mathsf{SS}}(\lambda)] - 1/2\right| \leq \nu(\lambda)\,,$$

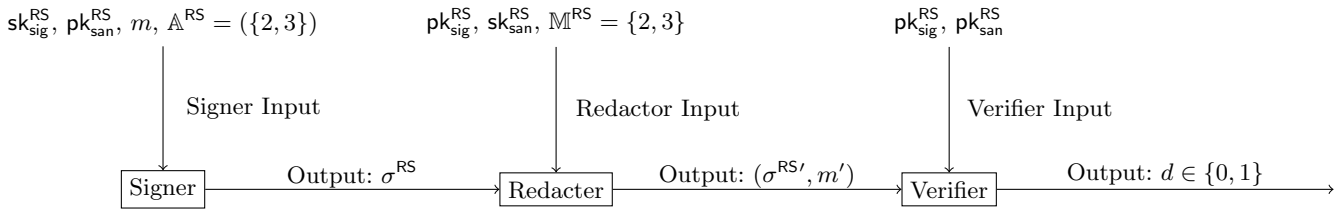


Fig. 15: Example workflow of a designated redactor RS. The message m is set to (I, do, not, like, crypto). After redacting, m' is (I, like, crypto). As we only consider private RSs, the redacted parts are not visible.

where the corresponding experiment is defined in Figure 14.

3 Invisible Redactable Signatures

We now introduce the new notion of *invisible* redactable signatures. In contrast to standard RSs [25], such schemes hide which parts of a message can be redacted from outsiders. As in standard RSs anyone can redact — and one can thus trivially decide which blocks are redactable — we need to introduce a "designated redactor", which is the only party able to decide this question. Thus, the designated redactor is the sanitizer in an RS. This is related to already existing definitions, but in a different context [24,48]. However, before we start defining and constructing our invisible RS, we need to settle some additional notation.

Additional Notation In the following, let $m = (m^1, m^2, \ldots, m^\ell)$ be some message, while $\mathbb{A}^{\mathsf{RS}} \subseteq [1..\ell]$ denotes the admissible redactions, i.e., if $i \in \mathbb{A}^{\mathsf{RS}}$, then m^i can be redacted by the designated redactor. The variable $\mathbb{M}^{\mathsf{RS}} \subseteq [1..\ell]$ denotes how a message m is to be modified, i.e., each block m^i, $i \in \mathbb{M}^{\mathsf{RS}}$, is removed from m to form the redacted message m'. In comparison to Derler et al. [25], however, we already define how those data-structures look like for preciseness. Additionally, as done for SSs, we use the shorthand notation $m' \leftarrow \mathbb{M}^{\mathsf{RS}}(m)$ to denote such a redaction. The notation $\mathbb{M}^{\mathsf{RS}} \prec (m, \mathbb{A}^{\mathsf{RS}})$ means that $\mathbb{M}^{\mathsf{RS}}$ is a valid modification instruction w.r.t. m and $\mathbb{A}^{\mathsf{RS}}$. Likewise, we use $\mathbb{A}^{\mathsf{RS}} \prec m$ to denote that $\mathbb{A}^{\mathsf{RS}}$ is valid description of the admissible blocks w.r.t. m.

An example workflow is depicted in Figure 15, derived from the work done by Bilzhause et al. [6].

3.1 Framework

The following definitions for RSs are derived from Derler et al. [25], merged with the ideas given by Pöhls and Samelin [48], while also supporting parameter generation. Moreover, we do not consider additional "redaction information" $\mathsf{RED}^{\mathsf{RS}}$, as given by Derler et al. [25], as we have a designated redactor anyway.

Definition 17 (Invisible Redactable Signatures). *An invisible redactable signature* RS *consists of the following six algorithms, i.e.,* $\{\mathsf{PPGen}^{\mathsf{RS}}, \mathsf{KG}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{KG}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{Sign}^{\mathsf{RS}}, \mathsf{Verify}^{\mathsf{RS}}, \mathsf{Red}^{\mathsf{RS}}\}$*, such that:*

$\mathsf{PPGen}^{\mathsf{RS}}$**.** *The algorithm* $\mathsf{PPGen}^{\mathsf{RS}}$ *generates the public parameters:*

$$\mathsf{pp}_{\mathsf{RS}} \leftarrow_r \mathsf{PPGen}^{\mathsf{RS}}(1^\lambda)$$

We assume that $\mathsf{pp}_{\mathsf{RS}}$ *is implicitly input to all other algorithms.*

$\mathsf{KG}^{\mathsf{RS}}_{\mathsf{sig}}$**.** *The algorithm* $\mathsf{KG}^{\mathsf{RS}}_{\mathsf{sig}}$ *generates a key pair:*

$$(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}) \leftarrow_r \mathsf{KG}^{\mathsf{RS}}_{\mathsf{sig}}(\mathsf{pp}_{\mathsf{RS}})$$

$\mathsf{KG}^{\mathsf{RS}}_{\mathsf{san}}$. *The algorithm* $\mathsf{KG}^{\mathsf{RS}}$ *generates a key pair:*

$$(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}) \leftarrow_r \mathsf{KG}^{\mathsf{RS}}_{\mathsf{san}}(\mathsf{pp}_{\mathsf{RS}})$$

$\mathsf{Sign}^{\mathsf{RS}}$. *The algorithm* $\mathsf{Sign}^{\mathsf{RS}}$ *outputs a signature* σ^{RS} *and some redaction information* $\mathsf{RED}^{\mathsf{RS}}$ *on input of* $\mathsf{sk}^{\mathsf{RS}}_{\mathsf{sig}}$, $\mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}$, $\mathbb{A}^{\mathsf{RS}}$ *and a message* $m \in \mathcal{MS}$*:*

$$\sigma^{\mathsf{RS}} \leftarrow_r \mathsf{Sign}^{\mathsf{RS}}(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, m, \mathbb{A}^{\mathsf{RS}})$$

Note, it is assumed that $\mathbb{A}^{\mathsf{RS}}$ *can always be derived.*

$\mathsf{Verify}^{\mathsf{RS}}$. *The deterministic algorithm* $\mathsf{Verify}^{\mathsf{RS}}$ *verifies a signature* σ^{RS}*, i.e., outputs a decision* $d \in \{0,1\}$ *w.r.t.* $\mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}$, $\mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}$*, and a message* m*:*

$$d \leftarrow \mathsf{Verify}^{\mathsf{RS}}(\mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, m, \sigma^{\mathsf{RS}})$$

$\mathsf{Red}^{\mathsf{RS}}$. *The algorithm* $\mathsf{Red}^{\mathsf{RS}}$ *outputs a derived signature* $\sigma^{\mathsf{RS}\prime}$ *and a derived message* m'*, along with the new admissible blocks* $\mathbb{A}^{\mathsf{RS}\prime}$*, on input of* $\mathsf{sk}^{\mathsf{RS}}_{\mathsf{san}}$, $\mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}$*, a signature* σ^{RS}*, some modification instruction* $\mathbb{M}^{\mathsf{RS}}$ *and some redaction information* $\mathsf{RED}^{\mathsf{RS}}$*:*

$$(\sigma^{\mathsf{RS}\prime}, m', \mathbb{A}^{\mathsf{RS}\prime}) \leftarrow_r \mathsf{Red}^{\mathsf{RS}}(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, m, \sigma^{\mathsf{RS}}, \mathbb{M}^{\mathsf{RS}})$$

Correctness We also require that an RS is correct. We call an RS correct, if for all $\lambda \in \mathbb{N}$, for all $\mathsf{pp}_{\mathsf{RS}} \leftarrow_r \mathsf{PPGen}^{\mathsf{RS}}(1^\lambda)$, for all $(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}})$, for all $(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}})$, for all $m \in \mathcal{MS}$, for all $\mathbb{A}^{\mathsf{RS}} \in \{\mathbb{A}^{\mathsf{RS}\prime} \mid \mathbb{A}^{\mathsf{RS}\prime} \prec m\}$, for all $\sigma^{\mathsf{RS}} \leftarrow_r \mathsf{Sign}^{\mathsf{RS}}(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, m, \mathbb{A}^{\mathsf{RS}})$ we have that $\mathsf{Verify}^{\mathsf{RS}}(\mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, m, \sigma^{\mathsf{RS}}) = 1$ and also for all $\mathbb{M}^{\mathsf{RS}} \in \{\mathbb{M}^{\mathsf{RS}\prime} \mid \mathbb{M}^{\mathsf{RS}\prime} \prec (m, \mathbb{A}^{\mathsf{RS}})\}$, for all $(\sigma^{\mathsf{RS}\prime}, m', \mathbb{A}^{\mathsf{RS}\prime}) \leftarrow_r \mathsf{Red}^{\mathsf{RS}}(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, m, \sigma^{\mathsf{RS}}, \mathbb{M}^{\mathsf{RS}})$ we have that $\mathsf{Verify}^{\mathsf{RS}}(\mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, m', \sigma^{\mathsf{RS}\prime}) = 1$.

3.2 Security Requirements

We now introduce our security model for RSs. This is an extended version derived from Derler et al. [25], which is, in turn, derived from Brzuska et al. [10]. Note, moreover, that we do not need accountability, as in our construction accountability is given by the SS, much like Pöhls and Samelin [48] and Bilzhause et al. [5].

Unforgeability This definition requires that an adversary $\mathcal{A}$ cannot derive a message which is not derivable from any signed messages. We stress that, even though the set $\bigcup_{i=1}^{q}\{\mathbb{M}^{\mathsf{RS}}(m_i) \mid \mathbb{M}^{\mathsf{RS}} \prec (m_i, \mathbb{A}^{\mathsf{RS}}_i)\}$ may grow exponentially, membership is trivially to decide, i.e., in polynomial time.

Definition 18 (Unforgeability). *An* RS *is unforgeable, if for any PPT adversary* $\mathcal{A}$ *there exists a negligible function* ν *such that:*

$$\Pr[\mathbf{Exp}^{\mathsf{Unforgeability}}_{\mathcal{A},\mathsf{RS}}(\lambda) = 1] \leq \nu(\lambda)$$

where the corresponding experiment is defined in Figure 16.

Immutability This definition requires that an adversary $\mathcal{A}$, which even can generate $\mathsf{sk}^{\mathsf{RS}}_{\mathsf{san}}$, cannot derive a message which is not derivable from any signed messages. We stress that, even though the set $\bigcup_{i=1}^{q}\{\mathbb{M}^{\mathsf{RS}}(m_i) \mid \mathbb{M}^{\mathsf{RS}} \prec (m_i, \mathbb{A}^{\mathsf{RS}}_i)\}$ may grow exponentially, membership is trivially to decide, i.e., in polynomial time.

Definition 19 (Immutability). *An* RS *is immutable, if for any PPT adversary* $\mathcal{A}$ *there exists a negligible function* ν *such that:*

$$\Pr[\mathbf{Exp}^{\mathsf{Immutability}}_{\mathcal{A},\mathsf{RS}}(\lambda) = 1] \leq \nu(\lambda)$$

where the corresponding experiment is defined in Figure 17.

$\mathbf{Exp}^{\mathsf{Unforgeability}}_{\mathcal{A},\mathsf{RS}}(\lambda)$
$\mathsf{pp}_{\mathsf{RS}} \leftarrow_r \mathsf{PPGen}^{\mathsf{RS}}(1^\lambda)$
$(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}) \leftarrow_r \mathsf{KG}^{\mathsf{RS}}_{\mathsf{sig}}(\mathsf{pp}_{\mathsf{RS}})$
$(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}) \leftarrow_r \mathsf{KG}^{\mathsf{RS}}_{\mathsf{san}}(\mathsf{pp}_{\mathsf{RS}})$
$\mathcal{Q} \leftarrow \emptyset$
$(m^*, \sigma^{\mathsf{RS}*}) \leftarrow_r \mathcal{A}^{\mathsf{Sign}^{\mathsf{RS}\prime}(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{sig}},\cdot,\cdot,\cdot),\mathsf{Red}^{\mathsf{RS}\prime}(\cdot,\mathsf{sk}^{\mathsf{RS}}_{\mathsf{san}},\cdot,\cdot,\cdot)}(\mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}})$
where $\mathsf{Sign}^{\mathsf{RS}\prime}(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}\,\prime}_{\mathsf{san}}, m, \mathbb{A}^{\mathsf{RS}})$:
$\sigma^{\mathsf{RS}} \leftarrow_r \mathsf{Sign}^{\mathsf{RS}}(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}\,\prime}_{\mathsf{san}}, m, \mathbb{A}^{\mathsf{RS}})$
if $\mathsf{pk}^{\mathsf{RS}\,\prime}_{\mathsf{san}} = \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}$, let $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\sigma^{\mathsf{RS}}, m)\}$
return σ^{RS}
and $\mathsf{Red}^{\mathsf{RS}\prime}(\mathsf{pk}^{\mathsf{RS}\prime}_{\mathsf{sig}}, \mathsf{sk}^{\mathsf{RS}}_{\mathsf{san}}, m, \sigma^{\mathsf{RS}}, \mathbb{M}^{\mathsf{RS}})$:
$(\sigma^{\mathsf{RS}\prime}, m') \leftarrow_r \mathsf{Red}^{\mathsf{RS}}(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{RS}\prime}_{\mathsf{sig}}, m, \sigma^{\mathsf{RS}}, \mathbb{M}^{\mathsf{RS}})$
if $\mathsf{pk}^{\mathsf{RS}\,\prime}_{\mathsf{sig}} = \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}} \ \wedge \ \sigma^{\mathsf{RS}\prime} \neq \bot$, let $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\sigma^{\mathsf{RS}\,\prime}, m')\}$
return σ'
return 1, if $\mathsf{Verify}^{\mathsf{RS}}(\mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, m^*, \sigma^{\mathsf{RS}*}) = 1 \ \wedge \ (\sigma^{\mathsf{RS}*}, m^*) \notin \mathcal{Q}$
return 0

Fig. 16: RS Unforgeability

$\mathbf{Exp}^{\mathsf{Immutability}}_{\mathcal{A},\mathsf{RS}}(\lambda)$
$\mathsf{pp}_{\mathsf{RS}} \leftarrow_r \mathsf{PPGen}^{\mathsf{RS}}(1^\lambda)$
$\mathcal{Q} \leftarrow \emptyset$
$(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}) \leftarrow_r \mathsf{KG}^{\mathsf{RS}}(\mathsf{pp}_{\mathsf{RS}})$
$(m^*, \sigma^{\mathsf{RS}*}, \mathsf{pk}^*) \leftarrow_r \mathcal{A}^{\mathsf{Sign}^{\mathsf{RS}}(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{sig}},\cdot,\cdot,\cdot)}(\mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}})$
where $\mathsf{Sign}^{\mathsf{RS}}(\mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, m, \mathbb{A}^{\mathsf{RS}})$
$\mathcal{Q}[\mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}] \leftarrow \mathcal{Q}[\mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}] \cup \{\mathbb{M}^{\mathsf{RS}}(m) \mid \mathbb{M}^{\mathsf{RS}} \prec (m, \mathbb{A}^{\mathsf{RS}})\}$
return 1, if $\mathsf{Verify}^{\mathsf{RS}}(\mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^*, m^*, \sigma^{\mathsf{RS}*}) = 1 \ \wedge$
$m^* \notin \mathcal{Q}[\mathsf{pk}^*]$
return 0

Fig. 17: RS Immutability

$\mathbf{Exp}^{\mathsf{Privacy}}_{\mathcal{A},\mathsf{RS}}(\lambda)$
$\mathsf{pp}_{\mathsf{RS}} \leftarrow_r \mathsf{PPGen}^{\mathsf{RS}}(1^\lambda)$
$(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}) \leftarrow_r \mathsf{KG}^{\mathsf{RS}}(\mathsf{pp}_{\mathsf{RS}})$
$(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}) \leftarrow_r \mathsf{KG}^{\mathsf{RS}}_{\mathsf{san}}(\mathsf{pp}_{\mathsf{RS}})$
$b \leftarrow_r \{0, 1\}$
$a \leftarrow_r \mathcal{A}^{\mathsf{Sign}^{\mathsf{RS}}(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{sig}},\cdot,\cdot),\mathsf{Red}^{\mathsf{RS}\prime}(\cdot,\mathsf{sk}^{\mathsf{RS}}_{\mathsf{san}},\cdot,\cdot,\cdot),\mathsf{LoRRed}(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{sig}},\mathsf{sk}^{\mathsf{RS}}_{\mathsf{san}},\cdot,\cdot,\cdot,\cdot,\cdot,\cdot,b)}(\mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}})$
where $\mathsf{LoRRed}(m_0, m_1, \mathbb{M}^{\mathsf{RS}}_0, \mathbb{M}^{\mathsf{RS}}_1, \mathbb{A}^{\mathsf{RS}}_0, \mathbb{A}^{\mathsf{RS}}_1, b)$
$\sigma^{\mathsf{RS}}_0 \leftarrow_r \mathsf{Sign}^{\mathsf{RS}}(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, m_0, \mathbb{A}^{\mathsf{RS}}_0)$
$\sigma^{\mathsf{RS}}_1 \leftarrow_r \mathsf{Sign}^{\mathsf{RS}}(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, m_1, \mathbb{A}^{\mathsf{RS}}_1)$
$(\sigma^{\mathsf{RS}\prime}_0, m'_0, \mathbb{A}^{\mathsf{RS}\prime}_0) \leftarrow_r \mathsf{Red}^{\mathsf{RS}}(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, m_0, \sigma^{\mathsf{RS}}_0, \mathbb{M}^{\mathsf{RS}}_0)$
$(\sigma^{\mathsf{RS}\prime}_1, m'_1, \mathbb{A}^{\mathsf{RS}\prime}_1) \leftarrow_r \mathsf{Red}^{\mathsf{RS}}(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, m_1, \sigma^{\mathsf{RS}}_1, \mathbb{M}^{\mathsf{RS}}_1)$
return $\bot$, if $m'_0 \neq m'_1 \ \vee \ \mathbb{A}^{\mathsf{RS}\prime}_0 \neq \mathbb{A}^{\mathsf{RS}\prime}_1$
return $(m'_b, \sigma^{\mathsf{RS}\prime}_b)$
return 1, if $a = b$
return 0

Fig. 18: RS Privacy

Privacy This definition prohibits that an adversary $\mathcal{A}$ can learn anything about redacted parts. This is similar to the definition for SSs.

$\mathbf{Exp}^{\mathsf{Transparency}}_{\mathcal{A},\mathsf{RS}}(\lambda)$
$\mathsf{pp}_{\mathsf{RS}} \leftarrow_r \mathsf{PPGen}^{\mathsf{RS}}(1^\lambda)$
$(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}) \leftarrow_r \mathsf{KG}^{\mathsf{RS}}_{\mathsf{sig}}(\mathsf{pp}_{\mathsf{RS}})$
$(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}) \leftarrow_r \mathsf{KG}^{\mathsf{RS}}_{\mathsf{san}}(\mathsf{pp}_{\mathsf{RS}})$
$b \leftarrow_r \{0,1\}$
$a \leftarrow_r \mathcal{A}^{\mathsf{Sign}^{\mathsf{RS}}(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{sig}},\cdot,\cdot),\mathsf{Red}^{\mathsf{RS}\prime}(\cdot,\mathsf{sk}^{\mathsf{RS}}_{\mathsf{san}},\cdot,\cdot,\cdot),\mathsf{Sign/Red}(\cdot,\cdot,\cdot,\mathsf{sk}^{\mathsf{RS}}_{\mathsf{sig}},b)}(\mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}})$
where $\mathsf{Sign/Red}(m, \mathbb{M}^{\mathsf{RS}}, \mathbb{A}^{\mathsf{RS}}, b)$:
$\sigma^{\mathsf{RS}} \leftarrow_r \mathsf{Sign}^{\mathsf{RS}}(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, m, \mathbb{A}^{\mathsf{RS}})$
$(m', \sigma^{\mathsf{RS}\prime}, \mathbb{A}^{\mathsf{RS}\prime}) \leftarrow_r \mathsf{Red}^{\mathsf{RS}}(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, m, \sigma^{\mathsf{RS}}, \mathbb{M}^{\mathsf{RS}})$
if $b = 1$:
$\sigma^{\mathsf{RS}\prime} \leftarrow_r \mathsf{Sign}^{\mathsf{RS}}(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, m', \mathbb{A}^{\mathsf{RS}\prime})$
return $\sigma^{\mathsf{RS}\prime}$
return 1, if $a = b$
return 0

Fig. 19: RS Transparency

Definition 20 (Privacy). *An* RS *is private, if for any PPT adversary* $\mathcal{A}$ *there exists a negligible function* ν *such that:*

$$\left|\Pr[\mathbf{Exp}^{\mathsf{Privacy}}_{\mathcal{A},\mathsf{RS}}(\lambda)] - 1/2\right| \leq \nu(\lambda)$$

where the corresponding experiment is defined in Figure 18.

Transparency This definition prohibits that an adversary $\mathcal{A}$ can decide whether a signature was generated through $\mathsf{Sign}^{\mathsf{RS}}$ or $\mathsf{Red}^{\mathsf{RS}}$.

Definition 21 (Transparency). *An* RS *is transparent, if for any PPT adversary* $\mathcal{A}$ *there exists a negligible function* ν *such that:*

$$\left|\Pr[\mathbf{Exp}^{\mathsf{Transparency}}_{\mathcal{A},\mathsf{RS}}(\lambda)] - 1/2\right| \leq \nu(\lambda)$$

where the corresponding experiment is defined in Figure 19.

3.3 Invisibility of RSs

The new notion of invisibility prohibits that an adversary can decide which blocks of a message m are redactable. This is formalized in the same fashion as done for SSs. Namely, the adversary gains access to a LoRADM-oracle which either signs using $\mathbb{A}^{\mathsf{RS}}_0$ or $\mathbb{A}^{\mathsf{RS}}_1$. To avoid trivial attacks, the adversary is limited to redaction of $\mathbb{A}^{\mathsf{RS}}_0 \cap \mathbb{A}^{\mathsf{RS}}_1$.

Definition 22 (Invisibility). *An* RS *is invisible, if for any PPT adversary* $\mathcal{A}$ *there exists a negligible function* ν *such that:*

$$\left|\Pr[\mathbf{Exp}^{\mathsf{Invisibility}}_{\mathcal{A},\mathsf{RS}}(\lambda)] - 1/2\right| \leq \nu(\lambda)$$

where the corresponding experiment is defined in Figure 20.

Construction Our construction burrows several ideas from existing ones [10,45,52]. In a nutshell, the main idea is to sign each block (bound by some overall tag and some block-specific tag) and the relation between each block (as they are symmetric, a "left-of-relation" is sufficient).

In more detail, the signer holds a long-term key-pair for a Σ, while the sanitizer holds a key-pair for Π. For signing, the signer generates random tags for each block, and one additional random overall tag to bind all signatures to a specific message. It then generates an ephemeral signature key pair. The next step is

$\mathbf{Exp}^{\mathsf{Invisibility}}_{\mathcal{A},\mathsf{RS}}(\lambda)$
$\mathsf{pp}_{\mathsf{RS}} \leftarrow_r \mathsf{PPGen}^{\mathsf{RS}}(1^\lambda)$
$(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}) \leftarrow_r \mathsf{KG}^{\mathsf{RS}}(\mathsf{pp}_{\mathsf{RS}})$
$(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}) \leftarrow_r \mathsf{KG}^{\mathsf{RS}}_{\mathsf{san}}(\mathsf{pp}_{\mathsf{RS}})$
$b \leftarrow_r \{0,1\}$
$\mathcal{Q} \leftarrow \emptyset$
$a \leftarrow_r \mathcal{A}^{\mathsf{Red}^{\mathsf{RS}\prime}(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{san}},\cdot,\cdot,\cdot,\cdot,\cdot),\mathsf{LoRADM}(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}},\cdot,\cdot,\cdot,\cdot,\cdot,b)}(\mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}})$
where $\mathsf{LoRADM}(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}\prime}_{\mathsf{san}}, m, \mathbb{A}^{\mathsf{RS}}_0, \mathbb{A}^{\mathsf{RS}}_1, b)$:
return $\perp$, if $\mathbb{A}^{\mathsf{RS}}_0 \not\preccurlyeq m \wedge \mathbb{A}^{\mathsf{RS}}_1 \not\preccurlyeq m$
return $\perp$, if $\mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}} \neq \mathsf{pk}^{\mathsf{RS}\prime}_{\mathsf{san}} \ \wedge \ \mathbb{A}^{\mathsf{RS}}_0 \neq \mathbb{A}^{\mathsf{RS}}_1$
$\sigma^{\mathsf{RS}} \leftarrow_r \mathsf{Sign}^{\mathsf{RS}}(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}\prime}_{\mathsf{san}}, m, \mathbb{A}^{\mathsf{RS}}_b)$
if $\mathsf{pk}^{\mathsf{RS}\prime}_{\mathsf{san}} = \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}$, let $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, \sigma^{\mathsf{RS}}, \mathbb{A}^{\mathsf{RS}}_0 \cap \mathbb{A}^{\mathsf{RS}}_1)\}$
return σ^{RS}
where $\mathsf{Red}^{\mathsf{RS}\prime}(\mathsf{pk}^{\mathsf{SS}\prime}_{\mathsf{sig}}, \mathsf{sk}^{\mathsf{RS}}_{\mathsf{san}}, m, \mathbb{M}^{\mathsf{RS}}, \sigma^{\mathsf{RS}})$:
return $\perp$, if $\mathsf{pk}^{\mathsf{RS}\prime}_{\mathsf{sig}} = \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}} \wedge \exists (m, \sigma^{\mathsf{RS}}, \mathbb{A}^{\mathsf{RS}}) \in \mathcal{Q} : \mathbb{M}^{\mathsf{SS}} \not\preccurlyeq (m, \mathbb{A}^{\mathsf{SS}})$
$(m', \sigma^{\mathsf{RS}\prime}, \mathbb{A}^{\mathsf{RS}\prime}) \leftarrow_r \mathsf{Red}^{\mathsf{RS}}(\mathsf{pk}^{\mathsf{RS}\prime}_{\mathsf{sig}}, \mathsf{sk}^{\mathsf{RS}}_{\mathsf{san}}, m, \mathbb{M}^{\mathsf{RS}}, \sigma^{\mathsf{RS}})$
if $\mathsf{pk}^{\mathsf{RS}\prime}_{\mathsf{sig}} = \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}} \wedge \exists (m, \sigma^{\mathsf{RS}}, \mathbb{A}^{\mathsf{RS}\prime}) \in \mathcal{Q} : \mathbb{M}^{\mathsf{RS}} \prec (m, \mathbb{A}^{\mathsf{RS}\prime})$,
$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m', \sigma^{\mathsf{RS}\prime}, \mathbb{A}^{\mathsf{RS}\prime})\}$
return $(m', \sigma^{\mathsf{RS}\prime})$
return 1, if $a = b$
return 0

Fig. 20: RS Invisibility

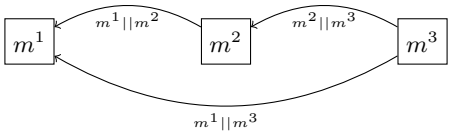


Fig. 21: Visualization of the "left-of" relation.

generating a signature on the overall tag and all public keys using the long-term key. Then, the signer has to generate a signature on each "left-of"-relation (See Figure 21 for a graphical explanation) of each block using the overall tag and the two block-specific tags, again using the long-term keys. The ephemeral secret key and each signature which should be redactable are then encrypted for the designated redactor. To avoid leaking how many blocks are redactable, the plaintext is padded accordingly. Namely, at most $\ell + \frac{\ell(\ell-1)}{2}$ signatures are given to the adversary. Moreover, to prohibit tampering with the ciphertext, it is signed along with the overall tag and all public keys using the ephemeral secret signing key. Finally, all generated signatures are aggregated. Verification simply checks the aggregate signature on the values received. For redaction, the designated redactor decrypts the ciphertext, re-generates the signature on the ciphertext and de-aggregates it. For every block to be redacted, the designated redactor simply de-aggregates all related signatures. Finally, it generates a new ciphertext with the remaining signatures, once more padded accordingly, signs it and aggregates it.

The proof of the following theorem is given in Appendix A.

Theorem 1. *If Π is correct and IND-CCA2 secure, while Σ is correct, unforgeable, unique and non-extractable, then the construction of an* RS*, given in Construction 1, is correct, unforgeable, immutable, private, transparent, and invisible.*

$\mathsf{PPGen}^{\mathsf{RS}}(1^\lambda)$. Generate $\mathsf{pp}_\Pi \leftarrow_r \mathsf{PPGen}^\Pi(1^\lambda)$ and $\mathsf{pp}_\Sigma \leftarrow_r \mathsf{PPGen}_\Sigma(1^\lambda)$. Return $(\mathsf{pp}_\Pi, \mathsf{pp}_\Sigma)$.

$\mathsf{KG}^{\mathsf{RS}}_{\mathsf{sig}}(\mathsf{pp}_{\mathsf{RS}})$. Generate $(\mathsf{sk}_\Sigma, \mathsf{pk}_\Sigma) \leftarrow_r \mathsf{KGen}_\Sigma(\mathsf{pp}_\Sigma)$. Return $(\mathsf{sk}_\Sigma, \mathsf{pk}_\Sigma)$.

$\mathsf{KG}^{\mathsf{RS}}_{\mathsf{san}}(\mathsf{pp}_{\mathsf{RS}})$. Generate $(\mathsf{sk}_\Pi, \mathsf{pk}_\Pi) \leftarrow_r \mathsf{KGen}^\Pi(\mathsf{pp}_\Pi)$. Return $(\mathsf{sk}_\Pi, \mathsf{pk}_\Pi)$.

$\mathsf{Sign}^{\mathsf{RS}}(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, m, \mathbb{A}^{\mathsf{RS}})$. The algorithm proceeds as follows:

- Draw $\ell + 1$ tags $\tau_i \leftarrow_r \{0,1\}^\lambda$ and generate a new signature key pair $(\mathsf{sk}_\Sigma', \mathsf{pk}_\Sigma') \leftarrow_r \mathsf{KGen}_\Sigma(\mathsf{pp}_\Sigma)$.
- For each i, let m'^i be the augmented block $(m^i, \tau_0, \tau_i, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}_\Sigma')$.
- Sign τ_0 and all public keys, i.e., let $\sigma_0 \leftarrow \mathsf{Sign}_\Sigma(\mathsf{sk}_\Sigma, (\tau_0, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}_\Sigma'))$, and sign each augmented block, i.e., let $\sigma_i \leftarrow \mathsf{Sign}_\Sigma(\mathsf{sk}_\Sigma, m'^i)$. Finally, sign each "left-of" relation, i.e., for all $i \in [1..\ell]$, for all $j < i$ let $\sigma_{i,j} \leftarrow \mathsf{Sign}_\Sigma(\mathsf{sk}_\Sigma, (m'^i, \tau_j))$.
- Encrypt the ephemeral secret key, all signatures related to the redactable blocks and some fake values to make the length of the encryption constant. In particular, generate $c \leftarrow_r \mathsf{Enc}^\Pi(\mathsf{pk}_\Pi, (\mathsf{sk}_\Sigma', \{\sigma_i\}_{i\in\mathbb{A}^{\mathsf{RS}}}, \{\sigma_{i,j}\}_{i\in\mathbb{A}^{\mathsf{RS}}}, t), (\mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}_\Sigma'))$, where t is a random string of length $\ell + \frac{\ell(\ell-1)}{2} - |(\{\sigma_i\}_{i\in\mathbb{A}^{\mathsf{RS}}} \cup \{\sigma_{i,j}\}_{i\in\mathbb{A}^{\mathsf{RS}}}|$ times the size of a single signature. This essentially makes the ciphertext always the same size, regardless of $\mathbb{A}^{\mathsf{RS}}$.
- Sign c using the ephemeral signature key, i.e., let $\sigma_c \leftarrow \mathsf{Sign}_\Sigma(\mathsf{sk}_\Sigma', (c, \tau_0, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}_\Sigma'))$.
- Aggregate all signatures generated, i.e., let $\sigma_a \leftarrow \mathsf{Agg}_\Sigma(\mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{S}_3 \cup \mathcal{S}_4)$, where $\mathcal{S}_1 = \{\{(\mathsf{pk}_\Sigma, \tau_0\}, \sigma_0)\}$, $\mathcal{S}_2 = \bigcup\{\{(\mathsf{pk}_\Sigma, m'^i\}, \sigma_i)\}_{i\in[1..\ell]}$, $\mathcal{S}_3 = \bigcup\{\{(\mathsf{pk}_\Sigma, (m'^i, \tau_j)\}, \sigma_{i,j})\}_{i\in[1..\ell], j\in[1..\ell], i<j}$, and $\mathcal{S}_4 = \{\{(\mathsf{pk}_\Sigma', (c, \tau_0, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}_\Sigma')\}, \sigma_c)\}$.
- Return $(\sigma_a, c, \mathsf{pk}_\Sigma', (\tau_i)_{i\in[0..\ell]})$.

$\mathsf{Verify}^{\mathsf{RS}}(\mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, m, \sigma^{\mathsf{RS}})$. Parse σ^{RS} as $(\sigma_a, c, \mathsf{pk}_\Sigma', (\tau_i)_{i\in[0..\ell]})$. Let m'^i be the augmented block $(m^i, \tau_0, \tau_i, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}_\Sigma')$. Return $\mathsf{AVerf}_\Sigma(\mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{S}_3 \cup \mathcal{S}_4, \sigma_a)$, where $\mathcal{S}_1 = \{(\mathsf{pk}_\Sigma, (\tau_0, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}_\Sigma'))\}$, $\mathcal{S}_2 = \{(\mathsf{pk}_\Sigma', (c, \tau_0, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}_\Sigma'))\}$, $\mathcal{S}_3 = \{(\mathsf{pk}_\Sigma, m'^i)\}_{i\in[1..\ell]}$, $\mathcal{S}_4 = \{(\mathsf{pk}_\Sigma, (m'^i, \tau_j))\}_{i\in[1..\ell], j\in[1..\ell], i<j}$.

$\mathsf{Red}^{\mathsf{RS}}(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, m, \sigma^{\mathsf{RS}}, \mathbb{M}^{\mathsf{RS}})$. Parse σ^{RS} as $(\sigma_a, c, \mathsf{pk}_\Sigma', (\tau_i)_{i\in[0..\ell_m]})$ and proceed as follows:

- If $\mathsf{Verify}^{\mathsf{RS}}(\mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, m, \sigma^{\mathsf{RS}}) \neq 1$, return false. Let $m'' \leftarrow \mathbb{M}^{\mathsf{RS}}(m)$.
- Let $(\mathsf{sk}_\Sigma', \{\sigma_i\}_{i\in\mathbb{A}^{\mathsf{RS}}}, \{\sigma_{i,j}\}_{i\in\mathbb{A}^{\mathsf{RS}}}, t) \leftarrow \mathsf{Dec}^\Pi(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{san}}, c, (\mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}_\Sigma'))$.
- For each i, let m'^i be the augmented block $(m^i, \tau_0, \tau_i, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}_\Sigma')$. Likewise, for each i, let m''^i be the augmented block $(m''^i, \tau_0, \tau_i, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}_\Sigma')$.
- Let $\sigma_c \leftarrow \mathsf{Sign}_\Sigma(\mathsf{sk}_\Sigma', (c, \tau_0, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}_\Sigma'))$.
- Update $\mathbb{A}^{\mathsf{RS}}$ to $\mathbb{A}^{\mathsf{RS}\prime}$ by removing all indices in $\mathbb{M}^{\mathsf{RS}}_2$ and adjusting the remaining indices by reducing each i in $\mathbb{A}^{\mathsf{RS}}$ by $|\{j \in \mathbb{M}^{\mathsf{RS}} : j < i\}|$.
- De-aggregate the signatures for the cipher and the messages (and relations) to be removed, i.e., compute $\sigma_a' \leftarrow \mathsf{DAgg}_\Sigma((\mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{S}_3), (\mathcal{S}_4 \cup \mathcal{S}_5 \cup \mathcal{S}_6 \cup \mathcal{S}_7), \sigma_a)$, where $\mathcal{S}_1 = \bigcup\{\{(\mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, m_i')\}, \sigma_i\}_{i\in\mathbb{M}^{\mathsf{RS}}}$, $\mathcal{S}_2 = \{\{(\mathsf{pk}_\Sigma', (c, \tau_0, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}_\Sigma')\}, \sigma_c)\}$, $\mathcal{S}_3 = \bigcup\{\{(\mathsf{pk}_\Sigma, (m'^i, \tau_j)\}, \sigma_{i,j})\}_{i\in\mathbb{M}^{\mathsf{RS}} \vee j\in\mathbb{M}^{\mathsf{RS}}}$, $\mathcal{S}_4 = \{(\mathsf{pk}_\Sigma, (\tau_0, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}_\Sigma'))\}$, $\mathcal{S}_5 = \{((\mathsf{pk}_\Sigma'), (c, \tau_0, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}_\Sigma'))\}$, $\mathcal{S}_6 = \bigcup\{((\mathsf{pk}_\Sigma), m'^i)\}_{i\in[1..\ell_m]}$, $\mathcal{S}_7 = \bigcup\{((\mathsf{pk}_\Sigma), (m'^i, \tau_j))\}_{i\in[1..\ell_m], j\in[1..\ell_m], i<j}$.
- Generate $c' \leftarrow_r \mathsf{Enc}^\Pi(\mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, (\mathsf{sk}_\Sigma', \{\sigma_i\}_{i\in\mathbb{A}^{\mathsf{RS}\prime}}, \{\sigma_{i,j}\}_{i\in\mathbb{A}^{\mathsf{RS}\prime}}, t'), (\mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}_\Sigma'))$, where t' is a random string of length $\ell_{m''} + \frac{\ell_{m''}(\ell_{m''}-1)}{2} - |(\{\sigma_i\}_{i\in\mathbb{A}^{\mathsf{RS}\prime}} \cup \{\sigma_{i,j}\}_{i\in\mathbb{A}^{\mathsf{RS}\prime}})|$ times the size of a single signature.
- Sign c', i.e., let $\sigma_c' \leftarrow \mathsf{Sign}_\Sigma(\mathsf{sk}_\Sigma', (c', \tau_0, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}_\Sigma'))$.
- Aggregate σ_c' onto σ_a' by calculating $\sigma_a'' \leftarrow \mathsf{Agg}_\Sigma(\mathcal{S}_1 \cup \{\mathcal{S}_2 \cup \mathcal{S}_3\}, \sigma_a'\})$, where $m'' = \mathbb{M}^{\mathsf{RS}}(m)$ is of length $\ell_{m''}$, $\mathcal{S}_1 = \{\{(\mathsf{pk}_\Sigma', (c, \tau_0, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}_\Sigma')\}, \sigma_c')\}$, $\mathcal{S}_2 = \bigcup\{(\mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, m_i''')\}_{i\in[1..\ell_{m''}]}$, and $\mathcal{S}_3 = \bigcup\{(\mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, (m''''^i, \tau_j))\}_{i\in[1..\ell_{m''}], j\in[1..\ell_{m''}], i<j}$.
- Return $((\sigma_a'', c', \mathsf{pk}_\Sigma', (\tau_i)_{i\in[0..\ell_{m''}]}), m'', \mathbb{A}^{\mathsf{RS}\prime})$.

Construction 1: Construction of an invisible RS

4 Non-Accountable Invisible SS

In our construction of fully invisible PS (See Section 5), we use non-accountable, yet invisible, SS. As accountability is one of the main concerns of SSs [2], this notion has, for obvious reasons, not been considered before [40]. However, as we show, in certain contexts such a notion has its merits.

Background In a nutshell, a non-accountable invisible SS, from now on denoted by SS′, behaves as a standard SS, but the algorithms $\mathsf{Proof}^{\mathsf{SS}}$ and $\mathsf{Judge}^{\mathsf{SS}}$ are simply set to $\perp$, i.e., effectively all algorithms related to accountability are dropped, clearly also affecting the correctness definition [11]. This also means that an SS′ may still achieve all security notions, but sanitizer-accountability and signer-accountability. This is exactly what our construction, given in Construction 2, achieves.

The reason for doing so is that accountability of the signatures can result from a different mechanism, what is exactly what we do in our final construction using an accountable SS.

Construction Our construction is given in Construction 2. The basic idea is that each block is signed using a fresh ephemeral signature key. If a block is admissible, the corresponding secret key is given to the sanitizer. This paradigm follows already existing ideas [27]. However, their scheme does not achieve transparency, while ours is not accountable.

In more detail, the signer holds a long-term key-pair for a Σ, while the sanitizer holds a key-pair for Π. At signing, the signer generates a fresh ephemeral key-pair for each block in the message m to sign. If a block is admissible, the randomness used to generate those key-pairs is derived from a PRF to achieve a smaller signature size. If a block is not admissible, fresh random coins are drawn. The secret key x for the PRF is encrypted to the sanitizer. All ephemeral public keys and the resulting ciphertext are signed using the long-term keys. For sanitizing, the sanitizer reconstructs the secret key x for the PRF and with it the signing keys for each admissible block, and then simply signs the blocks to be sanitized.

It is easy to see that this construction is invisible and does not provide any form of accountability, while we stress that we cannot avoid the encryption due to recent results [27]. Moreover, strictly speaking, our construction is even transparent in the sense of Brzuska et al. [11], i.e., the proof-restriction is not necessary.

$\mathsf{PPGen}^{\mathsf{SS}}(1^\lambda)$. Generate $\mathsf{pp}_\Pi \leftarrow_r \mathsf{PPGen}^\Pi(1^\lambda)$ and $\mathsf{pp}_\Sigma \leftarrow_r \mathsf{PPGen}_\Sigma(1^\lambda)$. Return $(\mathsf{pp}_\Pi, \mathsf{pp}_\Sigma)$.

$\mathsf{KG}^{\mathsf{SS}}_{\mathsf{sig}}(\mathsf{pp}_{\mathsf{SS}})$. Generate $(\mathsf{sk}_\Sigma, \mathsf{pk}_\Sigma) \leftarrow_r \mathsf{KGen}_\Sigma(\mathsf{pp}_\Sigma)$. Return $(\mathsf{sk}_\Sigma, \mathsf{pk}_\Sigma)$.

$\mathsf{KG}^{\mathsf{SS}}_{\mathsf{san}}(\mathsf{pp}_{\mathsf{SS}})$. Generate $(\mathsf{sk}_\Pi, \mathsf{pk}_\Pi) \leftarrow_r \mathsf{KGen}^\Pi(\mathsf{pp}_\Pi)$. Return $(\mathsf{sk}_\Pi, \mathsf{pk}_\Pi)$.

$\mathsf{Sign}^{\mathsf{SS}}(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}, m, \mathbb{A}^{\mathsf{SS}})$. The algorithm proceeds as follows:

- If $\mathbb{A}^{\mathsf{SS}} \prec m$ does not hold, return $\perp$.
- Draw $x \leftarrow_r \mathsf{KGen}_{\mathsf{PRF}}(1^\lambda)$.
- For $i \in [1..\ell]$, if $i \in \mathbb{A}^{\mathsf{SS}}.1$, let $r_i \leftarrow \mathsf{Eval}_{\mathsf{PRF}}(x, i)$ and $r_i \leftarrow_r \{0,1\}^\lambda$ otherwise. Set $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow_r \mathsf{KGen}_\Sigma(\mathsf{pp}_\Sigma; r_i)$.
- Encrypt the seed and $\mathbb{A}^{\mathsf{SS}}$, i.e., let $c \leftarrow_r \mathsf{Enc}^\Pi(\mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}, (x, \mathbb{A}^{\mathsf{SS}}), (\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}, (\mathsf{pk}_i)_{[1..\ell]}))$.
- Sign the public keys, and the ciphertext, i.e., let $\sigma_s \leftarrow \mathsf{Sign}_\Sigma(\mathsf{sk}_\Sigma, (\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}, (\mathsf{pk}_i)_{[1..\ell]}, c))$.
- Sign each randomized block using each sk_i, i.e., let $\sigma_i \leftarrow \mathsf{Sign}_\Sigma(\mathsf{sk}_i, ((\mathsf{pk}_i, m^i)_{[1..\ell]}, c, \sigma_s, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}))$.
- Return $(c, \sigma_s, (\mathsf{pk}_i, \sigma_i)_{i \in [1..\ell]})$.

$\mathsf{Verify}^{\mathsf{SS}}(\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}, m, \sigma^{\mathsf{SS}})$. Parse σ^{SS} as $(c, \sigma_s, (\mathsf{pk}_i, \sigma_i)_{i \in [1..\ell]})$. If $\mathsf{Verf}_\Sigma(\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, (\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}, (\mathsf{pk}_i)_{[1..\ell]}, c), \sigma_s) = 0$, return 0. If, for all $i \in [1..\ell]$, we have that $\mathsf{Verf}_\Sigma(\mathsf{pk}_i, ((\mathsf{pk}_i, m^i), c, \sigma_s, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}})), \sigma_i) = 1$, return 1. Otherwise, return 0.

$\mathsf{Sanit}^{\mathsf{SS}}(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, m, \sigma^{\mathsf{SS}}, \mathbb{M}^{\mathsf{SS}})$. The algorithm parses σ^{SS} as $(c, \sigma_s, (\mathsf{pk}_i, \sigma_i)_{i \in [1..\ell]})$ and proceeds as follows:

- If $\mathsf{Verify}^{\mathsf{SS}}(\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}, m, \sigma^{\mathsf{SS}}) \neq 1$, return $\perp$.
- Let $(x, \mathbb{A}^{\mathsf{SS}}) \leftarrow \mathsf{Dec}^\Pi(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{san}}, c, (\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}, (\mathsf{pk}_i)_{[1..\ell]}))$. If decryption fails, return $\perp$.
- For each $i \in \mathbb{A}^{\mathsf{SS}}.1$, let $(\mathsf{sk}'_i, \mathsf{pk}'_i) \leftarrow_r \mathsf{KGen}_\Sigma(\mathsf{pp}_\Sigma; \mathsf{Eval}_{\mathsf{PRF}}(x, i))$.
- If $\mathbb{M}^{\mathsf{SS}} \setminus \mathbb{A}^{\mathsf{SS}}.1 \neq \emptyset$, return $\perp$.
- For each $(i, m^{i\prime}) \in \mathbb{M}^{\mathsf{SS}}$, let $\sigma'_i \leftarrow \mathsf{Sign}_\Sigma(\mathsf{sk}'_i, ((\mathsf{pk}_i, m'^i)_{[1..\ell]}, c, \sigma_s, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}))$.
- Return $(\mathbb{M}^{\mathsf{SS}}(m), (c, \sigma_s, (\mathsf{pk}_i, \sigma'_i)_{i \in [1..\ell]}))$.

Construction 2: Construction of a non-accountable invisible SS'

We stress that we could also aggregate all signatures generated. However, to keep the description short, we opted for not doing this.

The proof of the following theorem is found in Appendix B.

Theorem 2. *If Π is correct and IND-CCA2 secure, PRF pseudorandom and correct, while Σ is correct and unforgeable, the construction of a SS', given in Construction 2, is correct, unforgeable, immutable, private, transparent, and invisible.*

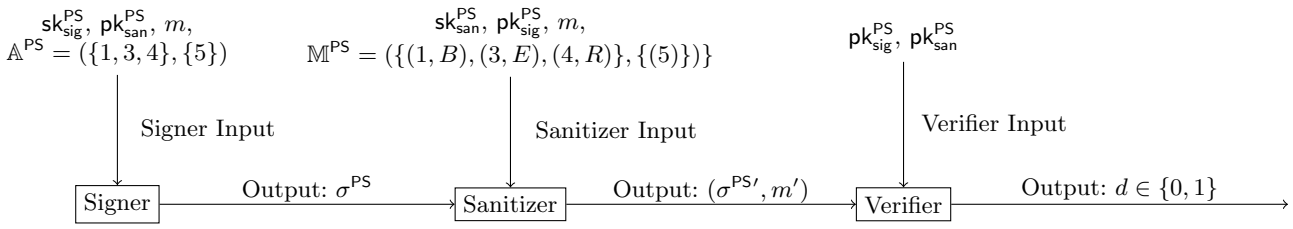


Fig. 22: Example workflow of a PS. The message m is set to (H,E,L,L,O) and is modified to (B,E,E,R).

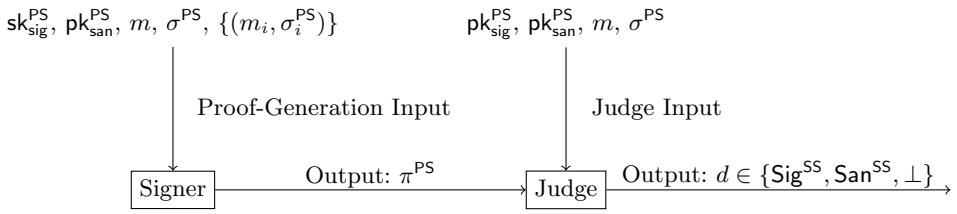


Fig. 23: Proof-generation and $\mathsf{Judge}^{\mathsf{PS}}$

5 Fully Invisible Protean Signatures

We now present our framework for PSs, which is taken from Krenn et al. [40].

To recap, a PS allows to remove and alter signer-chosen parts of a signed message by a semi-trusted third party, i.e., the sanitizer. The sanitizer can also be held accountable, if it chose to edit a signed message. For the framework, we settle some additional notation, which is derived from the ones used for RSs and SSs, to ease understanding.

5.1 Framework

For the framework, we use the following notation. The variable $\mathbb{A}^{\mathsf{PS}}$ is a list containing the set of indices of the editable blocks, as well as the blocks which can be redacted. For example, let $\mathbb{A}^{\mathsf{PS}} = (\{1,2\},\{4\})$. Then, the first and second block are editable, while only the fourth block can be redacted. The variable $\mathbb{M}^{\mathsf{PS}}$ is a list containing a set of pairs $(i, m^{i\prime})$ for those blocks that are modified, meaning that m^i is replaced by $m^{i\prime}$ and a set of indices to be redacted. In more detail, if $\mathbb{M}^{\mathsf{PS}} = (\{(1,b),(2,b)\},\{3\})$ means that the first two blocks are altered to contain a b, while the third block is redacted.

We use the shorthand notation $m' \leftarrow \mathbb{M}^{\mathsf{PS}}(m)$ to denote the result of this replacement, while $\mathbb{M}^{\mathsf{PS}} \prec (m, \mathbb{A}^{\mathsf{PS}})$ means that $\mathbb{M}^{\mathsf{PS}}$ is a valid modification instruction w.r.t. m and $\mathbb{A}^{\mathsf{PS}}$. Likewise, we use $\mathbb{A}^{\mathsf{PS}} \prec m$ to denote that $\mathbb{A}^{\mathsf{PS}}$ is valid description of the admissible blocks w.r.t. m.

An example workflow is depicted in Figure 22 and Figure 23. To ease understanding and the description of our construction, we define that the replacements are done first and the redactions afterwards.

Definition 23 (Protean Signatures). *A protean signature scheme* PS *consists of the following eight PPT algorithms* $(\mathsf{PPGen}^{\mathsf{PS}}, \mathsf{KG}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{KG}^{\mathsf{PS}}_{\mathsf{san}}, \mathsf{Sign}^{\mathsf{PS}}, \mathsf{Verify}^{\mathsf{PS}}, \mathsf{Edit}^{\mathsf{PS}}, \mathsf{Proof}^{\mathsf{PS}}, \mathsf{Judge}^{\mathsf{PS}})$ *such that:*

$\mathsf{PPGen}^{\mathsf{PS}}$**.** *The algorithm* $\mathsf{PPGen}^{\mathsf{PS}}$ *generates the public parameters:*

$$\mathsf{pp}_{\mathsf{PS}} \leftarrow_r \mathsf{PPGen}^{\mathsf{PS}}(1^\lambda)$$

We assume that $\mathsf{pp}_{\mathsf{PS}}$ *is implicitly input to all other algorithms.*

$\mathsf{KG}^{\mathsf{PS}}_{\mathsf{sig}}$. *The algorithm* $\mathsf{KG}^{\mathsf{PS}}_{\mathsf{sig}}$ *generates the key pair of the signer:*

$$(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}) \leftarrow_r \mathsf{KG}^{\mathsf{PS}}_{\mathsf{sig}}(\mathsf{pp}_{\mathsf{PS}})$$

$\mathsf{KG}^{\mathsf{PS}}_{\mathsf{san}}$. *The algorithm* $\mathsf{KG}^{\mathsf{SS}}_{\mathsf{san}}$ *generates the key pair of the sanitizer:*

$$(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}) \leftarrow_r \mathsf{KG}^{\mathsf{PS}}_{\mathsf{san}}(\mathsf{pp}_{\mathsf{PS}})$$

$\mathsf{Sign}^{\mathsf{PS}}$. *The algorithm* $\mathsf{Sign}^{\mathsf{PS}}$ *generates a signature* σ^{PS} *on input of the public key* $\mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}$, $\mathbb{A}^{\mathsf{PS}}$, *a message* m, *and* $\mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}}$*:*

$$\sigma^{\mathsf{PS}} \leftarrow_r \mathsf{Sign}^{\mathsf{PS}}(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}, m, \mathbb{A}^{\mathsf{PS}})$$

It is assumed that $\mathbb{A}^{\mathsf{PS}}$ *can be derived from any verifying signature* σ^{PS}*, if* $\mathsf{sk}^{\mathsf{PS}}_{\mathsf{san}}$ *is known.*

$\mathsf{Verify}^{\mathsf{PS}}$. *The algorithm* $\mathsf{Verify}^{\mathsf{PS}}$ *verifies a signature* σ^{PS}*, i.e., outputs a decision* $d \in \{0,1\}$ *w.r.t.* $\mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}$, $\mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}$, *and a message* m*:*

$$d \leftarrow \mathsf{Verify}^{\mathsf{PS}}(\mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}, m, \sigma^{\mathsf{PS}})$$

$\mathsf{Edit}^{\mathsf{PS}}$. *The algorithm* $\mathsf{Edit}^{\mathsf{PS}}$ *generates a sanitized signature* $\sigma^{\mathsf{PS}\prime}$ *and updated* $\mathbb{A}^{\mathsf{PS}\prime}$*, given inputs* $\mathsf{sk}^{\mathsf{PS}}_{\mathsf{san}}$, $\mathbb{A}^{\mathsf{PS}}$, *a message* m*, a signature* σ*, and* $\mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}$*:*

$$(m', \sigma^{\mathsf{PS}\prime}, \mathbb{A}^{\mathsf{PS}\prime}) \leftarrow_r \mathsf{Edit}^{\mathsf{PS}}(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \sigma, m, \mathbb{M}^{\mathsf{PS}})$$

$\mathsf{Proof}^{\mathsf{PS}}$. *The algorithm* $\mathsf{Proof}^{\mathsf{PS}}$ *outputs a proof* π^{PS} *on input* m, σ^{PS}, $\mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}}$, $\mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}$*, and a set of polynomially many additional signature/message pairs* $\{(\sigma^{\mathsf{PS}}_i, m^i)\}$*. The proof* π^{PS} *is used by the next algorithm to pinpoint the accountable party for a given signature:*

$$\pi^{\mathsf{PS}} \leftarrow_r \mathsf{Proof}^{\mathsf{PS}}(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}, m, \sigma^{\mathsf{PS}}, \{(\sigma^{\mathsf{PS}}_i, m^i)\})$$

$\mathsf{Judge}^{\mathsf{PS}}$. *The algorithm* $\mathsf{Judge}^{\mathsf{PS}}$ *outputs a decision* $d \in \{\mathsf{Sig}^{\mathsf{PS}}, \mathsf{San}^{\mathsf{PS}}, \perp\}$ *indicating whether the message/signature pair has been created by the signer, or the sanitizer:*

$$d \leftarrow \mathsf{Judge}^{\mathsf{PS}}(\mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}, m, \sigma^{\mathsf{PS}}, \pi^{\mathsf{PS}})$$

5.2 PSs Security Definitions

We now introduce the security properties for PSs. Clearly, the goals are similar to the ones for SSs and RSs. However, due to the extended capabilities, the semantic is quite different, while we need to take extra care for changed indices after redactions.

Unforgeability This definition requires that an adversary $\mathcal{A}$ not having any secret keys is not able to produce any valid signature $\sigma^{\mathsf{PS}*}$ on a message m^* which it has never not seen, even if $\mathcal{A}$ has full oracle access, i.e., this captures "strong unforgeability" [41].

Definition 24 (Unforgeability). *A* PS *is unforgeable, if for any PPT adversary* $\mathcal{A}$ *there exists a negligible function* ν *such that*

$$\Pr[\mathbf{Exp}^{\mathsf{Unforgeability}}_{\mathcal{A},\mathsf{PS}}(\lambda) = 1] \leq \nu(\lambda)\,,$$

where the corresponding experiment is defined in Figure 24.

$\mathbf{Exp}^{\mathsf{Unforgeability}}_{\mathcal{A},\mathsf{PS}}(\lambda)$
$\mathsf{pp}_{\mathsf{PS}} \leftarrow_r \mathsf{PPGen}^{\mathsf{PS}}(1^\lambda)$
$(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}) \leftarrow_r \mathsf{KG}^{\mathsf{PS}}_{\mathsf{sig}}(\mathsf{pp}_{\mathsf{PS}})$
$(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}) \leftarrow_r \mathsf{KG}^{\mathsf{PS}}_{\mathsf{san}}(\mathsf{pp}_{\mathsf{PS}})$
$(m^*, \sigma^{\mathsf{PS}*}) \leftarrow_r \mathcal{A}^{\mathsf{Sign}^{\mathsf{PS}}(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}},\cdot,\cdot,\cdot),\mathsf{Edit}^{\mathsf{PS}}(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{san}},\cdot,\cdot,\cdot,\cdot),\mathsf{Proof}^{\mathsf{PS}}(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}},\cdot,\cdot,\cdot,\cdot)}(\mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}})$
for $i = 1, 2, \ldots, q$ let $(\mathsf{pk}^{\mathsf{PS}}_{\mathsf{san},i}, m_i, \mathbb{A}^{\mathsf{PS}}_i)$ and σ^{PS}_i
index the queries/answers to/from $\mathsf{Sign}^{\mathsf{PS}}$
for $j = 1, 2, \ldots, q'$ let $(\mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig},j}, m_j, \sigma^{\mathsf{PS}}_j, \mathbb{M}_j)$ and $(m'_j, \sigma^{\mathsf{PS}\prime}_j, \mathbb{A}^{\mathsf{PS}\prime}_j)$
index the queries/answers to/from $\mathsf{Edit}^{\mathsf{PS}}$
return 1, if $\mathsf{Verify}^{\mathsf{PS}}(\mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}, m^*, \sigma^{\mathsf{PS}*}) = 1 \wedge$
$\forall i \in [1..q] : (\mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}, m^*, \sigma^{\mathsf{PS}*}) \neq (\mathsf{pk}^{\mathsf{PS}}_{\mathsf{san},i}, m_i, \sigma^{\mathsf{PS}}_i) \wedge$
$\forall j \in [1..q'] : (\mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, m^*, \sigma^{\mathsf{PS}*}) \neq (\mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig},j}, m'_j, \sigma^{\mathsf{PS}\prime}_j)$
return 0

Fig. 24: PS Unforgeability

$\mathbf{Exp}^{\mathsf{Immutability}}_{\mathcal{A},\mathsf{PS}}(\lambda)$
$\mathsf{pp}_{\mathsf{PS}} \leftarrow_r \mathsf{PPGen}^{\mathsf{PS}}(1^\lambda)$
$(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}) \leftarrow_r \mathsf{KG}^{\mathsf{PS}}_{\mathsf{sig}}(\mathsf{pp}_{\mathsf{PS}})$
$(m^*, \sigma^{\mathsf{PS}*}, \mathsf{pk}^{\mathsf{PS}\,*}_{\mathsf{san}}) \leftarrow_r \mathcal{A}^{\mathsf{Sign}^{\mathsf{PS}}(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}},\cdot,\cdot,\cdot),\mathsf{Proof}^{\mathsf{PS}}(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}},\cdot,\cdot,\cdot,\cdot)}(\mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}})$
for $i = 1, 2, \ldots, q$ let $(\mathsf{pk}^{\mathsf{PS}}_{\mathsf{san},i}, m_i, \mathbb{A}^{\mathsf{PS}}_i)$
index the queries to $\mathsf{Sign}^{\mathsf{PS}}$
return 1, if $\mathsf{Verify}^{\mathsf{PS}}(\mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}\,*}_{\mathsf{san}}, m^*, \sigma^{\mathsf{PS}*}) = 1 \wedge$
$\forall i \in [1..q] : (\mathsf{pk}^{\mathsf{PS}\,*}_{\mathsf{san}} \neq \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san},i} \vee$
$m^* \notin \{\mathbb{M}(m_i) \mid \mathbb{M} \text{ with } \mathbb{M} \prec (m_i, \mathbb{A}^{\mathsf{PS}}_i)\})$
return 0

Fig. 25: PS Immutability

Immutability This definition prohibits that an adversary $\mathcal{A}$ can generate a verifying signature $\sigma^{\mathsf{PS}*}$ for a message m^* not derivable from the signatures given by an honest signer, even if it can generate the sanitizer's key pair.

Definition 25 (Immutability). *A* PS *is immutable, if for any PPT adversary $\mathcal{A}$ there exists a negligible function ν such that*

$$\Pr[\mathbf{Exp}^{\mathsf{Immutability}}_{\mathcal{A},\mathsf{PS}}(\lambda) = 1] \leq \nu(\lambda)\,,$$

where the corresponding experiment is defined in Figure 25.

Privacy This definition prohibits that an adversary $\mathcal{A}$ can learn anything about edited (redacted or sanitized) parts.

Definition 26 (Privacy). *A* PS *is private, if for any PPT adversary $\mathcal{A}$ there exists a negligible function ν such that*

$$\left|\Pr[\mathbf{Exp}^{\mathsf{Immutability}}_{\mathcal{A},\mathsf{PS}}(\lambda)] - 1/2\right| \leq \nu(\lambda)\,,$$

where the corresponding experiment is defined in Figure 26.

Transparency This definition requires that an adversary $\mathcal{A}$ does not learn whether a signature σ^{PS} was generated through $\mathsf{Sign}^{\mathsf{PS}}$ or $\mathsf{Edit}^{\mathsf{PS}}$.

$\mathbf{Exp}^{\mathsf{Privacy}}_{\mathcal{A},\mathsf{PS}}(\lambda)$
$\mathsf{pp}_{\mathsf{PS}} \leftarrow_r \mathsf{PPGen}^{\mathsf{PS}}(1^\lambda)$
$(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}) \leftarrow_r \mathsf{KG}^{\mathsf{PS}}_{\mathsf{sig}}(\mathsf{pp}_{\mathsf{PS}})$
$(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}) \leftarrow_r \mathsf{KG}^{\mathsf{PS}}_{\mathsf{san}}(\mathsf{pp}_{\mathsf{PS}})$
$b \leftarrow_r \{0,1\}$
$a \leftarrow_r \mathcal{A}^{\mathsf{Sign}^{\mathsf{PS}}(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}},\cdot,\cdot,\cdot),\mathsf{Edit}^{\mathsf{PS}}(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{san}},\cdot,\cdot,\cdot,\cdot)}_{\mathsf{Proof}^{\mathsf{PS}}(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}},\cdot,\cdot,\cdot,\cdot),\mathsf{LoREdit}(\cdot,\cdot,\cdot,\cdot,\cdot,\mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}},\mathsf{sk}^{\mathsf{PS}}_{\mathsf{san}},b)}(\mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}})$
where $\mathsf{LoREdit}(m_0, m_1, \mathbb{M}^{\mathsf{PS}}_0, \mathbb{M}^{\mathsf{PS}}_1, \mathbb{A}^{\mathsf{PS}}_0, \mathbb{A}^{\mathsf{PS}}_1, \mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{sk}^{\mathsf{PS}}_{\mathsf{san}}, b)$
$\sigma^{\mathsf{PS}}_i \leftarrow_r \mathsf{Sign}^{\mathsf{PS}}(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}, m_i, \mathbb{A}^{\mathsf{PS}}_i)$ for $i \in \{0,1\}$
$(m'_i, \sigma^{\mathsf{PS}\prime}_i, \mathbb{A}^{\mathsf{PS}\prime}_i) \leftarrow_r \mathsf{Edit}^{\mathsf{PS}}(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, m_i, \sigma^{\mathsf{PS}}_i, \mathbb{M}^{\mathsf{PS}}_i)$ for $i \in \{0,1\}$
return $\bot$, if $m'_0 \neq m'_1 \ \vee \ \mathbb{A}^{\mathsf{PS}\prime}_0 \neq \mathbb{A}^{\mathsf{PS}\prime}_1$
return $(m'_b, \sigma^{\mathsf{PS}\prime}_b, \mathbb{A}^{\mathsf{PS}\prime}_b)$
return 1, if $a = b$
return 0

Fig. 26: PS Privacy

$\mathbf{Exp}^{\mathsf{Transparency}}_{\mathcal{A},\mathsf{PS}}(\lambda)$
$\mathsf{pp}_{\mathsf{PS}} \leftarrow_r \mathsf{PPGen}^{\mathsf{PS}}(1^\lambda)$
$(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}) \leftarrow_r \mathsf{KG}^{\mathsf{PS}}_{\mathsf{sig}}(\mathsf{pp}_{\mathsf{PS}})$
$(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}) \leftarrow_r \mathsf{KG}^{\mathsf{PS}}_{\mathsf{san}}(\mathsf{pp}_{\mathsf{PS}})$
$b \leftarrow_r \{0,1\}$
$\mathcal{Q} \leftarrow \emptyset$
$a \leftarrow_r \mathcal{A}^{\mathsf{Sign}^{\mathsf{PS}}(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}},\cdot,\cdot,\cdot),\mathsf{Edit}^{\mathsf{PS}}(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{san}},\cdot,\cdot,\cdot,\cdot)}_{\mathsf{Proof}^{\mathsf{PS}\prime}(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}},\cdot,\cdot,\cdot,\cdot),\mathsf{Sign/Edit}(\cdot,\cdot,\cdot,\mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}},\mathsf{sk}^{\mathsf{PS}}_{\mathsf{san}},b)}(\mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}})$
where $\mathsf{Proof}^{\mathsf{PS}\prime}(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}}, m, \sigma^{\mathsf{PS}}, \{(m_i, \sigma^{\mathsf{PS}}_i) \mid i \in \mathbb{N}\})$:
return $\bot$, if $\mathsf{pk}^{\mathsf{PS}\prime}_{\mathsf{san}} = \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}} \ \wedge \ ((m, \sigma^{\mathsf{PS}}) \in \mathcal{Q} \ \vee \ \mathcal{Q} \cap \{(m_i, \sigma^{\mathsf{PS}}_i)\} \neq \emptyset)$
return $\mathsf{Proof}^{\mathsf{PS}}(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}\prime}_{\mathsf{san}}, m, \sigma^{\mathsf{PS}}, \{(m_i, \sigma^{\mathsf{PS}}_i)\})$
where $\mathsf{Sign/Edit}(m, \mathbb{M}^{\mathsf{PS}}, \mathbb{A}^{\mathsf{PS}}, \mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{sk}^{\mathsf{PS}}_{\mathsf{san}}, b)$:
$\sigma^{\mathsf{PS}} \leftarrow_r \mathsf{Sign}^{\mathsf{PS}}(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}, m, \mathbb{A}^{\mathsf{PS}})$
$(m', \sigma^{\mathsf{PS}\prime}, \mathbb{A}^{\mathsf{PS}\prime}) \leftarrow_r \mathsf{Edit}^{\mathsf{PS}}(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, m, \sigma^{\mathsf{PS}}, \mathbb{M}^{\mathsf{PS}})$
if $b = 1$:
$\sigma^{\mathsf{PS}\prime} \leftarrow_r \mathsf{Sign}^{\mathsf{PS}}(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}, m', \mathbb{A}^{\mathsf{PS}\prime})$
if $\sigma^{\mathsf{PS}\prime} \neq \bot$, set $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m', \sigma^{\mathsf{PS}\prime})\}$
return $(m', \sigma^{\mathsf{PS}\prime})$
return 1, if $a = b$
return 0

Fig. 27: PS Transparency

Definition 27 (Transparency). *A* PS *is transparent, if for any PPT adversary $\mathcal{A}$ there exists a negligible function ν such that*

$$\left|\Pr[\mathbf{Exp}^{\mathsf{Transparency}}_{\mathcal{A},\mathsf{PS}}(\lambda)] - 1/2\right| \leq \nu(\lambda)\,,$$

where the corresponding experiment is defined in Figure 27.

Signer-Accountability Signer-accountability prohibits that an adversary can generate a bogus proof that makes $\mathsf{Judge}^{\mathsf{PS}}$ decide that the sanitizer is responsible for a given signature/message pair $(m^*, \sigma^{\mathsf{PS}*})$, but the sanitizer has never generated this pair. This is even true, if the adversary can generate the signer's key pair.

Definition 28 (Signer-Accountability). *A* PS *is signer-accountable, if for any PPT adversary $\mathcal{A}$ there exists a negligible function ν such that*

$\mathbf{Exp}^{\mathsf{SigAccountability}}_{\mathcal{A},\mathsf{PS}}(\lambda)$
$\mathsf{pp}_{\mathsf{PS}} \leftarrow_r \mathsf{PPGen}^{\mathsf{PS}}(1^\lambda)$
$(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}) \leftarrow_r \mathsf{KG}^{\mathsf{PS}}_{\mathsf{san}}(\mathsf{pp}_{\mathsf{PS}})$
$(\mathsf{pk}^{\mathsf{PS}*}_{\mathsf{sig}}, \pi^{\mathsf{PS}*}, m^*, \sigma^{\mathsf{PS}*}) \leftarrow_r \mathcal{A}^{\mathsf{Edit}^{\mathsf{PS}}(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{san}},\cdot,\cdot,\cdot,\cdot)}(\mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}})$
for $i = 1, 2, \ldots, q$ let $(m'_i, \sigma^{\mathsf{PS}\prime}_i, \mathbb{A}^{\mathsf{PS}\prime}_j)$ and $(m_i, \mathbb{M}^{\mathsf{PS}}_i, \sigma^{\mathsf{PS}}_i, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig},i})$
index the answers/queries from/to $\mathsf{Edit}^{\mathsf{PS}}$
return 1, if $\mathsf{Verify}^{\mathsf{PS}}(\mathsf{pk}^{\mathsf{PS}*}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}, m^*, \sigma^{\mathsf{PS}*}) = 1 \wedge$
$\forall i \in [1..q] : (\mathsf{pk}^{\mathsf{PS}*}_{\mathsf{sig}}, m^*, \sigma^{\mathsf{PS}*}) \neq (\mathsf{pk}_{\mathsf{sig}}{}^{\mathsf{PS}}_{,i}, m'_i, \sigma^{\mathsf{PS}\prime}_i) \wedge$
$\mathsf{Judge}^{\mathsf{PS}}(\mathsf{pk}^{\mathsf{PS}*}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}, m^*, \sigma^{\mathsf{PS}*}, \pi^{\mathsf{PS}*}) = \mathsf{San}^{\mathsf{PS}}$
return 0

Fig. 28: PS Signer-Accountability

$\mathbf{Exp}^{\mathsf{SanAccountability}}_{\mathcal{A},\mathsf{PS}}(\lambda)$
$\mathsf{pp}_{\mathsf{PS}} \leftarrow_r \mathsf{PPGen}^{\mathsf{PS}}(1^\lambda)$
$(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}) \leftarrow_r \mathsf{KG}^{\mathsf{PS}}_{\mathsf{sig}}(\mathsf{pp}_{\mathsf{PS}})$
$(m^*, \sigma^{\mathsf{PS}*}, \mathsf{pk}^{\mathsf{PS}*}_{\mathsf{san}}) \leftarrow_r \mathcal{A}^{\mathsf{Sign}^{\mathsf{PS}}(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}},\cdot,\cdot,\cdot),\mathsf{Proof}^{\mathsf{PS}}(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}},\cdot,\cdot,\cdot,\cdot)}(\mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}})$
for $i = 1, 2, \ldots, q$ let $(\mathsf{pk}^{\mathsf{PS}}_{\mathsf{san},i}, m_i, \mathbb{A}^{\mathsf{PS}}_i)$ and σ^{PS}_i
index the queries/answers to/from $\mathsf{Sign}^{\mathsf{PS}}$
$\pi^{\mathsf{PS}} \leftarrow_r \mathsf{Proof}^{\mathsf{PS}}(\mathsf{sk}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}*}_{\mathsf{san}}, m^*, \sigma^{\mathsf{PS}*}, \{(m_i, \sigma^{\mathsf{PS}}_i) \mid 0 < i \leq q\})$
return 1, if $\mathsf{Verify}^{\mathsf{PS}}(\mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}*}_{\mathsf{san}}, m^*, \sigma^{\mathsf{PS}*}) = 1 \wedge$
$\forall i \in [1..q] : (\mathsf{pk}^{\mathsf{PS}*}_{\mathsf{san}}, m^*, \sigma^{\mathsf{PS}*}) \neq (\mathsf{pk}^{\mathsf{PS}}_{\mathsf{san},i}, m_i, \sigma^{\mathsf{PS}}_i) \wedge$
$\mathsf{Judge}^{\mathsf{PS}}(\mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}*}_{\mathsf{san}}, m^*, \sigma^{\mathsf{PS}*}, \pi^{\mathsf{PS}}) = \mathsf{Sig}^{\mathsf{PS}}$
return 0

Fig. 29: PS Sanitizer-Accountability

$$\Pr[\mathbf{Exp}^{\mathsf{SigAccountability}}_{\mathcal{A},\mathsf{PS}}(\lambda) = 1] \leq \nu(\lambda)\,,$$

where the corresponding experiment is defined in Figure 28.

Sanitizer-Accountability Sanitizer-accountability prohibits that an adversary can generate a bogus signature/message pair $(m^*, \sigma^{\mathsf{PS}*})$ that makes $\mathsf{Proof}^{\mathsf{SS}}$ output an honestly generated proof π^{PS} which points to the signer, but $(m^*, \sigma^{\mathsf{PS}*})$ has never been generated by the signer. This is even true, if the adversary can generate the sanitizer's key pair.

Definition 29 (Sanitizer-Accountability). *A* PS *is sanitizer-accountable, if for any PPT adversary* $\mathcal{A}$ *there exists a negligible function* ν *such that*

$$\Pr[\mathbf{Exp}^{\mathsf{SanAccountability}}_{\mathcal{A},\mathsf{PS}}(\lambda) = 1] \leq \nu(\lambda)\,,$$

where the corresponding experiment is defined in Figure 29.

5.3 (Full) Invisibility

Invisibility prohibits that an outsider can decide which blocks can be edited and also which blocks can be redacted. Note, the signing oracle can be simulated by using the same $\mathbb{A}^{\mathsf{PS}}$ in the LoRADM oracle. Moreover, as done for SSs (See Section 2), we define a slightly stronger variant than defined by Krenn et al. [40]: the adversary $\mathcal{A}$ can now query arbitrary signature to the LoRADM-oracle.

$\mathbf{Exp}^{\mathsf{Invisibility}}_{\mathcal{A},\mathsf{PS}}(\lambda)$
$\mathsf{pp}_{\mathsf{PS}} \leftarrow_r \mathsf{PPGen}^{\mathsf{PS}}(1^\lambda)$
$(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}) \leftarrow_r \mathsf{KG}^{\mathsf{PS}}_{\mathsf{sig}}(\mathsf{pp}_{\mathsf{PS}})$
$(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}) \leftarrow_r \mathsf{KG}^{\mathsf{PS}}_{\mathsf{san}}(\mathsf{pp}_{\mathsf{PS}})$
$b \leftarrow_r \{0,1\}$
$\mathcal{Q} \leftarrow \emptyset$
$a \leftarrow_r \mathcal{A}^{\mathsf{Edit}^{\mathsf{PS}\prime}(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{san}},\cdot,\cdot,\cdot,\cdot,\cdot),\mathsf{Proof}^{\mathsf{PS}}(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}},\cdot,\cdot,\cdot,\cdot,\cdot),\mathsf{LoRADM}(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}},\cdot,\cdot,\cdot,\cdot,\cdot,b)}(\mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}})$
where $\mathsf{LoRADM}(\mathsf{sk}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}\prime}_{\mathsf{san}}, m, \mathbb{A}^{\mathsf{PS}}_0, \mathbb{A}^{\mathsf{PS}}_1, b)$:
return $\bot$, if $\mathbb{A}^{\mathsf{PS}}_0 \not\propto m \wedge \mathbb{A}^{\mathsf{PS}}_1 \not\propto m$
return $\bot$, if $\mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}} \neq \mathsf{pk}^{\mathsf{PS}\prime}_{\mathsf{san}} \ \wedge \ \mathbb{A}^{\mathsf{PS}}_0 \neq \mathbb{A}^{\mathsf{PS}}_1$
$\sigma^{\mathsf{PS}} \leftarrow_r \mathsf{Sign}^{\mathsf{PS}}(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}\prime}_{\mathsf{san}}, m, \mathbb{A}^{\mathsf{PS}}_b)$
if $\mathsf{pk}^{\mathsf{PS}\prime}_{\mathsf{san}} = \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}$
$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, \sigma^{\mathsf{PS}}, (\mathbb{A}^{\mathsf{PS}}_0.1 \cap \mathbb{A}^{\mathsf{PS}}_1.1, \mathbb{A}^{\mathsf{PS}}_0.2 \cap \mathbb{A}^{\mathsf{PS}}_1.2))\}$
return σ^{PS}
where $\mathsf{Edit}^{\mathsf{PS}\prime}(\mathsf{pk}^{\mathsf{PS}\prime}_{\mathsf{sig}}, \mathsf{sk}^{\mathsf{PS}}_{\mathsf{san}}, \sigma^{\mathsf{PS}}, m, \mathbb{M}^{\mathsf{PS}})$:
return $\bot$, if $\mathsf{pk}^{\mathsf{PS}\prime}_{\mathsf{sig}} = \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}} \wedge \exists (m, \sigma^{\mathsf{PS}}, \mathbb{A}) \in \mathcal{Q} : \mathbb{M}^{\mathsf{PS}} \not\propto (m, \mathbb{A})$
$(m', \sigma^{\mathsf{PS}\prime}, \mathbb{A}^{\mathsf{PS}\prime\prime}) \leftarrow_r \mathsf{Edit}^{\mathsf{PS}}(\mathsf{pk}^{\mathsf{PS}\prime}_{\mathsf{sig}}, \mathsf{sk}^{\mathsf{PS}}_{\mathsf{san}}, m, \mathbb{M}^{\mathsf{PS}}, \sigma^{\mathsf{PS}})$
if $\mathsf{pk}^{\mathsf{PS}\prime}_{\mathsf{sig}} = \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}} \wedge \exists (m, \sigma^{\mathsf{PS}}, \mathbb{A}^{\mathsf{PS}\prime}) \in \mathcal{Q} : \mathbb{M}^{\mathsf{PS}} \prec (m, \mathbb{A}^{\mathsf{PS}\prime})$,
$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m', \sigma^{\mathsf{PS}\prime}, \mathbb{A}^{\mathsf{PS}\prime\prime})\}$
return $(m', \sigma^{\mathsf{PS}\prime})$
return 1, if $a = b$
return 0

Fig. 30: PS Invisibility

Definition 30 (Invisibility). *A* PS *is (fully) invisible, if for any PPT adversary* $\mathcal{A}$ *there exists a negligible function* ν *such that*

$$\left|\Pr[\mathbf{Exp}^{\mathsf{Invisibility}}_{\mathcal{A},\mathsf{PS}}(\lambda)] - 1/2\right| \leq \nu(\lambda)\,,$$

where the corresponding experiment is defined in Figure 30.

5.4 Construction

We now present our construction of a PS. It is essentially the same as given by Krenn et al. [40], but with some minor, yet very important, quirks.

The basic idea is to combine RSs and SSs by bridging them using unique tags. In more detail, each block $m^i \in m$ is signed using an invisible non-accountable SS′, while an additional (non-admissible) tag τ is used to identify the "overall" message m the block m^i belongs to. Moreover, each block m^i is also assigned a (non-admissible) additional tag τ_i, along with all public keys, used by an invisible RS to allow for redactions. Thus, there are ℓ_m σ^{SS}_i, where each signature protects $(m^i, \tau, \tau_i, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}})$. If a block m_i is sanitizable, it is marked as admissible within $\mathbb{A}^{\mathsf{SS}}_i$. This allows to sanitize the block m^i. Then, each tag τ_i is put into an RS to allow for transparent redactions, additionally bound to the non-redactable "overall" tag τ and all (non-redactable) public keys. If a block m^i is non-redactable, this is marked in $\mathbb{A}^{\mathsf{RS}}$. Thus, σ^{RS} protects $(\tau_1, \ldots, \tau_{\ell_m}, \tau, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}})$. Finally, to achieve accountability, all tags, all signatures generated so far, the resulting values are signed again using an additional, non-invisible but accountable, SS, while in this outer SS everything, but the public keys and the tag τ are admissible. To maintain transparency, the overall message m is a *single* block in the outer SS.

In more detail, the outer signature σ^{SS}_0 protects $(m, \sigma^{\mathsf{RS}}, (\tau_i, \sigma^{\mathsf{SS}}_i)_i, \tau, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}})$. Thus, changing the message or any signature requires changing σ^{SS}_0, implying accountability. Upon sanitization of a block m^i, σ^{SS}_i is sanitized, while the outer signature σ^{SS}_0 needs to be adjusted as well. For redaction of a block m^i, σ^{RS} is adjusted and the corresponding signature is no longer given out. This also means that σ^{SS}_0 must be adjusted.

The resulting construction is depicted in Construction 3. To give a graphical overview of the construction idea, see Figure 31 (before editing) and Figure 32 (after editing). Moreover, we do not consider unlinkability [13], as it seems to be very hard to achieve with the underlying construction paradigm, especially considering that there are no SSs yet which are unlinkable and invisible at the same time.

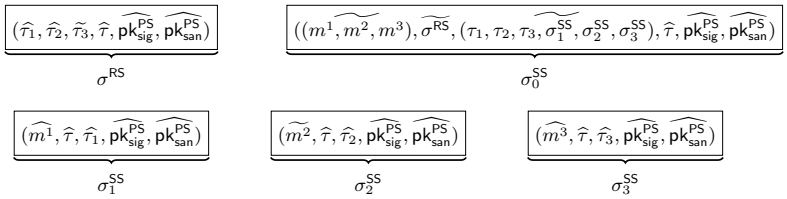


Fig. 31: Our main construction idea. Let $\mathbb{A}^{\mathsf{PS}} = (\{2\}, \{3\})$ and $m = (m^1, m^2, m^3)$ for preciseness, i.e., only the second block of m is sanitizable, while only the last block of m is redactable. Redactable elements for the RS (or sanitizable for the SS) are marked with a tilde, i.e., $\widetilde{\cdot}$. Blocks which are not redactable (or sanitizable resp.) are marked with a hat, i.e., $\widehat{\cdot}$.

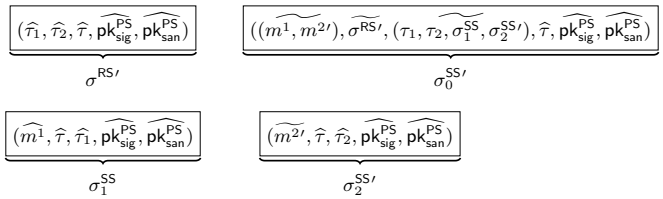


Fig. 32: State after sanitization. Here, block m^3 was redacted and m^2 was changed to $m^{2\prime}$. Block m^1 must stay the same.

The proof of the following Theorem is found in Appendix C.

Theorem 3. *If* SS *is unforgeable, immutable, private, transparent, signer-accountable, and sanitizer-accountable,* RS *is correct, unforgeable, immutable, private, transparent, and invisible, while* SS′ *is unforgeable, immutable, private, transparent, and invisible, then the construction of a* PS*, given in Construction 3, is correct, unforgeable, private, transparent, immutable, signer-accountable, sanitizer-accountable, and (fully) invisible.*

5.5 Implementation

To show that the construction really is practical, we provide an evaluation of our implementation. To maintain comparability with existing measurements [40], we have chosen to use the same parameters (as far as possible). Namely, our implementation is done in Java 10 and measured on an Intel i5-2400@3.10GHz with 16GiB of RAM. As the (aggregate) signature scheme, we implemented BGLS [8], while for Π we chose CS-encryption [21]. As the underlying groups we chose IAIK's ECCelerate pairings library[5]. In particular, the underlying pairing curves are "SNARK_2" (Π uses only the first group). As the outer SS, we use the construction given by Gong et al. [32], but altered using the results given by Krenn et al. [39] and Beck et al. [3] to meet the stronger security definitions. Namely, we use the unique chameleon-hash [38] by Krenn et al. [39], with 2'048 Bit moduli, also hash the nonce, encrypt the original signature to the signer, and use

[5] https://jce.iaik.tugraz.at/sic/Products/Core_Crypto_Toolkits/ECCelerate

$\mathsf{PPGen}^{\mathsf{PS}}(1^\lambda)$. Let $\mathsf{pp}_{\mathsf{SS}} \leftarrow_r \mathsf{SS.PPGen}^{\mathsf{SS}}(1^\lambda)$, $\mathsf{pp}_{\mathsf{SS}}{}' \leftarrow_r \mathsf{SS'.PPGen}^{\mathsf{SS}}(1^\lambda)$, and $\mathsf{pp}_{\mathsf{RS}} \leftarrow_r \mathsf{PPGen}^{\mathsf{RS}}(1^\lambda)$.
Return $\mathsf{pp}_{\mathsf{PS}} = (\mathsf{pp}_{\mathsf{SS}}, \mathsf{pp}_{\mathsf{SS}}{}', \mathsf{pp}_{\mathsf{RS}})$.

$\mathsf{KG}^{\mathsf{PS}}_{\mathsf{sig}}(\mathsf{pp}_{\mathsf{PS}})$. Let $(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}) \leftarrow_r \mathsf{SS.KG}^{\mathsf{SS}}_{\mathsf{sig}}(\mathsf{pp}_{\mathsf{SS}})$, $(\mathsf{sk}^{\mathsf{SS}\prime}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}\prime}_{\mathsf{sig}}) \leftarrow_r \mathsf{SS'.KG}^{\mathsf{SS}}_{\mathsf{sig}}(\mathsf{pp}_{\mathsf{SS}}{}')$, and $(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}) \leftarrow_r \mathsf{KG}^{\mathsf{RS}}_{\mathsf{sig}}(\mathsf{pp}_{\mathsf{RS}})$.
Return $(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}) = ((\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{sk}^{\mathsf{SS}\prime}_{\mathsf{sig}}, \mathsf{sk}^{\mathsf{RS}}_{\mathsf{sig}}), (\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}\prime}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}))$.

$\mathsf{KG}^{\mathsf{PS}}_{\mathsf{san}}(\mathsf{pp}_{\mathsf{PS}})$. Let $(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}) \leftarrow_r \mathsf{SS.KG}^{\mathsf{SS}}_{\mathsf{san}}(\mathsf{pp}_{\mathsf{SS}})$, $(\mathsf{sk}^{\mathsf{SS}\prime}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{SS}\prime}_{\mathsf{san}}) \leftarrow_r \mathsf{SS'.KG}^{\mathsf{SS}}_{\mathsf{san}}(\mathsf{pp}_{\mathsf{SS}})$, and $(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}) \leftarrow_r \mathsf{KG}^{\mathsf{RS}}_{\mathsf{san}}(\mathsf{pp}_{\mathsf{RS}})$.
Return $(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}) = ((\mathsf{sk}^{\mathsf{SS}}_{\mathsf{san}}, \mathsf{sk}^{\mathsf{SS}\prime}_{\mathsf{san}}, \mathsf{sk}^{\mathsf{RS}}_{\mathsf{san}}), (\mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{SS}\prime}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}))$.

$\mathsf{Sign}^{\mathsf{PS}}(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}, m, \mathbb{A}^{\mathsf{PS}})$. The algorithm proceeds as follows:
- If $\mathbb{A}^{\mathsf{PS}} \prec m = (m^1, m^2, \ldots, m^\ell)$ does not hold, return $\perp$, otherwise parse $\mathbb{A}^{\mathsf{PS}} = (\mathbb{A}^{\mathsf{PS}}_1, \mathbb{A}^{\mathsf{PS}}_2)$.
- Draw $\tau \leftarrow_r \{0,1\}^\lambda$.
- For all $i \in [1..\ell_m]$, let $\sigma^{\mathsf{SS}}_i \leftarrow_r \mathsf{SS'.Sign}^{\mathsf{SS}}(\mathsf{sk}^{\mathsf{SS}\prime}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}, (m^i, \tau, \tau_i, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}), \mathbb{A}^{\mathsf{SS}}_i)$, where each $\tau_i \leftarrow_r \{0,1\}^\lambda$. Furthermore, if $i \in \mathbb{A}^{\mathsf{PS}}_1$, let $\mathbb{A}^{\mathsf{SS}}_i = (\{1\}, 5)$ and $\mathbb{A}^{\mathsf{SS}}_i = (\emptyset, 5)$ otherwise.
- Let $\sigma^{\mathsf{RS}} \leftarrow_r \mathsf{Sign}^{\mathsf{RS}}(\mathsf{sk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, m', \mathbb{A}^{\mathsf{RS}})$, where $\mathbb{A}^{\mathsf{RS}} = \mathbb{A}^{\mathsf{PS}}_2$ and the message $m' = (\tau_1, \ldots, \tau_\ell, \tau, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}})$.
- Generate $\sigma^{\mathsf{SS}}_0 \leftarrow_r \mathsf{SS.Sign}^{\mathsf{SS}}(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}, (m, \sigma^{\mathsf{RS}}, (\tau_i, \sigma^{\mathsf{SS}}_i)_{1 \le i \le \ell_m}, \tau, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}), \mathbb{A}^{\mathsf{SS}}_0)$, where $\mathbb{A}^{\mathsf{SS}}_0 = (\{1,2,3\}, 6)$.
- Return $((\sigma^{\mathsf{SS}}_i)_{0 \le i \le \ell_m}, \sigma^{\mathsf{RS}}, \tau, (\tau_i)_{1 \le i \le \ell_m})$.

$\mathsf{Verify}^{\mathsf{PS}}(\mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}, m, \sigma^{\mathsf{PS}})$. If $\mathsf{Verify}^{\mathsf{SS}}(\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}, (m, \sigma^{\mathsf{RS}}, \tau, (\tau_i, \sigma^{\mathsf{SS}}_i)_{1 \le i \le \ell_m}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}), \sigma^{\mathsf{SS}}_0) = 0$, return 0. If $\mathsf{Verify}^{\mathsf{RS}}(\mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, (\tau_1, \ldots, \tau_\ell, \tau, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}), \sigma^{\mathsf{RS}}) = 0$, return 0. If for any $i \in [1..\ell_m] : \mathsf{SS'.Verify}^{\mathsf{SS}}(\mathsf{pk}^{\mathsf{SS}\prime}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}, (m^i, \tau, \tau_i, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}), \sigma^{\mathsf{SS}}_i) = 0$, return 0. Return 1.

$\mathsf{Edit}^{\mathsf{PS}}(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \sigma^{\mathsf{PS}}, m, \mathbb{M}^{\mathsf{PS}})$. The algorithm proceeds as follows:
- If $\mathsf{Verify}^{\mathsf{PS}}(\mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}, m, \sigma^{\mathsf{PS}}) = 0$, return $\perp$, otherwise parse $\mathbb{M}^{\mathsf{PS}} = (\mathbb{M}^{\mathsf{PS}}_1, \mathbb{M}^{\mathsf{PS}}_2)$ and continue.
- For all $(i, m^{i\prime}) \in \mathbb{M}^{\mathsf{PS}}_1$, let $(m^{i\prime}, \sigma^{\mathsf{SS}\prime}_i) \leftarrow_r \mathsf{SS'.Sanit}^{\mathsf{SS}}(\mathsf{sk}^{\mathsf{SS}\prime}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, (m^i, \tau, \tau_i, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}), \sigma^{\mathsf{SS}}_i, \{(0, m^{i\prime})\})$. Return $\perp$ if $\sigma^{\mathsf{SS}\prime}_i = \perp$, otherwise set $\sigma^{\mathsf{SS}}_i \leftarrow \sigma^{\mathsf{SS}\prime}_i$.
- Generate $(\sigma^{\mathsf{RS}\prime}, \cdot, \cdot) \leftarrow_r \mathsf{Red}^{\mathsf{RS}}(\mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, m'', \sigma^{\mathsf{RS}}, \mathbb{M}^{\mathsf{PS}}_2, \mathsf{RED}^{\mathsf{RS}})$, where $m'' = (\tau_1, \ldots, \tau_\ell, \tau, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}})$. If $\sigma^{\mathsf{RS}\prime} = \perp$, return $\perp$.
- Let $(m'_0, \sigma^{\mathsf{SS}\prime}_0) \leftarrow_r \mathsf{SS.Sanit}^{\mathsf{SS}}(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{san}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, (m, \sigma^{\mathsf{RS}}, (\tau_i, \sigma^{\mathsf{SS}}_i)_{i \in [1..\ell]}, \tau, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}), \sigma^{\mathsf{SS}}_0, \{(1, m'), (2, \sigma^{\mathsf{RS}\prime}), (3, (\tau_i, \sigma^{\mathsf{SS}\prime}_i)_{i \in [1..\ell] \setminus \mathbb{M}^{\mathsf{PS}}_2})\})$. If $\sigma^{\mathsf{SS}\prime}_0 = \perp$, return $\perp$.
- Update $\mathbb{A}^{\mathsf{PS}}$ to $\mathbb{A}^{\mathsf{PS}\prime}$ by removing all indices in $\mathbb{M}^{\mathsf{PS}}_2$ and adjusting the remaining indices by reducing each i in $\mathbb{A}^{\mathsf{PS}}$ by $|\{j \in \mathbb{M}^{\mathsf{PS}}_2 : j < i\}|$.
- Return $(\mathbb{M}^{\mathsf{PS}}(m), ((\sigma^{\mathsf{SS}}_i)_{i \in [1..\ell] \setminus \mathbb{M}^{\mathsf{PS}}_2}, \sigma^{\mathsf{RS}\prime}, \tau, (\tau_i)_{i \in [1..\ell] \setminus \mathbb{M}^{\mathsf{PS}}_2}), \mathbb{A}^{\mathsf{PS}\prime})$.

$\mathsf{Proof}^{\mathsf{PS}}(\mathsf{sk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}, m, \sigma^{\mathsf{PS}}, \{(\sigma^{\mathsf{PS}}_i, m^i)\})$. If for any $(\sigma^{\mathsf{PS}}_i, m_i)$, $\mathsf{Verify}^{\mathsf{PS}}(\mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}, m_i, \sigma^{\mathsf{PS}}_i) = 0$, return $\perp$. If $\mathsf{Verify}^{\mathsf{PS}}(\mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}, m, \sigma^{\mathsf{PS}}) = 0$, return $\perp$. Return $\mathsf{SS.Proof}^{\mathsf{SS}}(\mathsf{sk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}, m', \sigma^{\mathsf{SS}}, \{(\sigma^{\mathsf{SS}}_i, m'_i)\})$, where $m' = (m, \sigma^{\mathsf{RS}}, (\tau_i)_{1 \le i \le \ell_m}, \tau, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}})$ and each $m'_i = (m_i, \sigma^{\mathsf{RS}}_i, (\tau_{i,j})_{1 \le j \le \ell_{m_i}}, \tau_i, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}})$.

$\mathsf{Judge}^{\mathsf{PS}}(\mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}, m, \sigma^{\mathsf{PS}}, \pi^{\mathsf{PS}})$. If $\mathsf{Verify}^{\mathsf{PS}}(\mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}, m, \sigma^{\mathsf{PS}}) = 0$, return $\perp$. Return $\mathsf{SS.Judge}^{\mathsf{SS}}(\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}, m', \sigma^{\mathsf{SS}}, \pi^{\mathsf{PS}})$, where $m' = (m, \sigma^{\mathsf{RS}}, (\tau_i)_{1 \le i \le \ell_m}, \tau, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}})$.

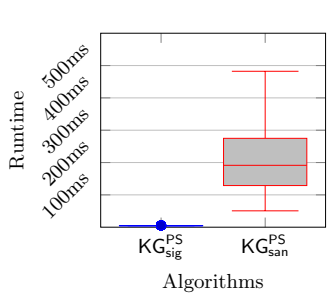
Construction 3: Our fully invisible PS scheme

unique signatures as done by Beck et al. [3], to achieve the stronger unforgeability, privacy and accountability definitions (but invisibility).

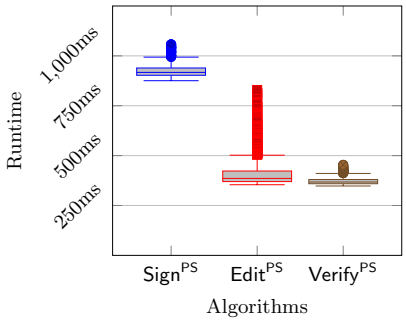
We did not implement any particular optimizations with the following exceptions: BGLS signatures allow a significant performance gain at verification, if the aggregate contains many signatures signed under the same public key [8]. As this is the case for the used RS, we chose to implement this optimization. An additional optimization we have implemented is that the signatures σ^{SS}_i for the invisible SS′ are signed again using the outer SS, and thus are not required to be unique in this context — and can be replaced by standard unforgeable signatures.

As Krenn et al. [40], we evaluated our implementation with 32 blocks, whereas 25% were marked as admissible, and an additional 25% as redactable. For editing, 50% of the admissible blocks were sanitized and redacted. We omit proof generation and the judge, as they are simple database look-ups, and parameter generation as it is a one-time setup. The overall results are depicted in Figure 33a, Figure 33b, and Table 33c. They are based on 1'000 runs, while verification was measured after sanitization. Note, however, that we have

measured $\mathsf{Edit}^{\mathsf{PS}}$ without verifying each signature to see the actual runtime of the editing part, and not the verification one.



(a) Box-plots of the run-times in ms



(b) Box-plots of the run-times in ms

	$\mathsf{KG}^{\mathsf{PS}}_{\mathsf{sig}}$	$\mathsf{KG}^{\mathsf{PS}}_{\mathsf{san}}$	$\mathsf{Sign}^{\mathsf{PS}}$	$\mathsf{Edit}^{\mathsf{PS}}$	$\mathsf{Verify}^{\mathsf{PS}}$
Min.:	4	50	875	354	347
25th PCTL:	4	128	902	369	359
Median:	4	191	917	385	367
75th PCTL:	4	274	939	422	380
90th PCTL:	5	365	972	499	400
95th PCTL:	5	425	991	571	412
Max.:	5	482	1'057	828	454
Average:	4	211	925	416	372
SD:	0.40	103.51	34.07	75.79	18.99

(c) Percentiles for our implementation in ms

Fig. 33: Performance Evaluation Results

As it can easily be seen, even our not entirely optimized implementation is a order of magnitude faster than the original construction, and even offers stronger security guarantees. Interestingly enough, our construction is even faster than the invisible SS introduced by Beck et al. [3], while, as already clarified by Krenn et al. [40], a PS can simply mimic a SS by prohibiting redactions.

6 Conclusion

We have strengthened the state-of-the-art definition of invisibility for Protean Signatures (PS) to also account for the redactable parts. In particular, using our new notion an outsider can neither decide which parts are redactable nor which parts are editable. To achieve this, we introduced the new notions of invisible redactable signatures (RS), non-accountable invisible sanitizable signature schemes (SS), and a novel framework for aggregate signatures which explicitly allow for de-aggregation of signatures. Using those primitives, our resulting provably secure construction becomes practically efficient, proven by our prototypical implementation.

Acknowledgements. The projects leading to this work have received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 653454 (CREDENTIAL), No 644962 (PRISMACLOUD), No 321310 (PERCY), and No 783119 (SECREDAS), No 780315 (SEMIOTICS), respectively.

References

1. J. H. Ahn, D. Boneh, J. Camenisch, S. Hohenberger, a. shelat, and B. Waters. Computing on authenticated data. *J. Cryptology*, 28(2):351–395, 2015.
2. G. Ateniese, D. H. Chou, B. de Medeiros, and G. Tsudik. Sanitizable signatures. In *ESORICS*, pages 159–177, 2005.
3. M. T. Beck, J. Camenisch, D. Derler, S. Krenn, H. C. Pöhls, K. Samelin, and D. Slamanig. Practical strongly invisible and strongly accountable sanitizable signatures. In *ACISP, Part I*, pages 437–452, 2017.
4. M. Bellare, C. Namprempre, and G. Neven. Unrestricted aggregate signatures. In *ICALP*, pages 411–422, 2007.
5. A. Bilzhause, M. Huber, H. C. Pöhls, and K. Samelin. Cryptographically enforced four-eyes principle. In *Ares*, pages 760–767, 2016.
6. A. Bilzhause, H. C. Pöhls, and K. Samelin. Position paper: The past, present, and future of sanitizable and redactable signatures. In *Ares*, pages 87:1–87:9, 2017.
7. D. Boneh and D. M. Freeman. Homomorphic signatures for polynomial functions. In *EuroCrypt*, pages 149–168, 2011.

8. D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *EuroCrypt*, pages 416–432, 2003.
9. E. Boyle, S. Goldwasser, and I. Ivan. Functional signatures and pseudorandom functions. In *PKC*, pages 501–519, 2014.
10. C. Brzuska, H. Busch, Ö. Dagdelen, M. Fischlin, M. Franz, S. Katzenbeisser, M. Manulis, C. Onete, A. Peter, B. Poettering, and D. Schröder. Redactable signatures for tree-structured data: Definitions and constructions. In *ACNS*, pages 87–104, 2010.
11. C. Brzuska, M. Fischlin, T. Freudenreich, A. Lehmann, M. Page, J. Schelbert, D. Schröder, and F. Volk. Security of sanitizable signatures revisited. In *PKC*, pages 317–336, 2009.
12. C. Brzuska, M. Fischlin, A. Lehmann, and D. Schröder. Santizable signatures: How to partially delegate control for authenticated data. In *BIOSIG*, 2009.
13. C. Brzuska, M. Fischlin, A. Lehmann, and D. Schröder. Unlinkability of sanitizable signatures. In *PKC*, pages 444–461, 2010.
14. C. Brzuska, H. C. Pöhls, and K. Samelin. Non-interactive public accountability for sanitizable signatures. In *EuroPKI*, pages 178–193, 2012.
15. C. Brzuska, H. C. Pöhls, and K. Samelin. Efficient and perfectly unlinkable sanitizable signatures without group signatures. In *EuroPKI*, pages 12–30, 2013.
16. J. Camenisch, D. Derler, S. Krenn, H. C. Pöhls, K. Samelin, and D. Slamanig. Chameleon-hashes with ephemeral trapdoors - and applications to invisible sanitizable signatures. In *PKC, Part II*, pages 152–182, 2017.
17. S. Canard and A. Jambert. On extended sanitizable signature schemes. In *CT-RSA*, pages 179–194, 2010.
18. S. Canard, A. Jambert, and R. Lescuyer. Sanitizable signatures with several signers and sanitizers. In *AfricaCrypt*, pages 35–52, 2012.
19. S. Canard, F. Laguillaumie, and M. Milhau. Trapdoor sanitizable signatures and their application to content protection. In *ACNS*, pages 258–276, 2008.
20. J.-S. Coron and D. Naccache. Boneh et al.'s k-element aggregate extraction assumption is equivalent to the diffie-hellman assumption. In *AsiaCrypt*, pages 392–397, 2003.
21. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Crypto*, pages 13–25, 1998.
22. H. de Meer, H. C. Pöhls, J. Posegga, and K. Samelin. On the relation between redactable and sanitizable signature schemes. In *ESSOS*, pages 113–130, 2014.
23. D. Demirel, D. Derler, C. Hanser, H. C. Pöhls, D. Slamanig, and G. Traverso. PRISMACLOUD D4.4: Overview of Functional and Malleable Signature Schemes. Technical report, H2020 Prismacloud, www.prismacloud.eu, 2015.
24. D. Derler, S. Krenn, and S. Slamanig. Signer-anonymous designated-verifier redactable signatures for cloud-based data sharing. In *CANS*, pages 211–227, 2016.
25. D. Derler, H. C. Pöhls, K. Samelin, and D. Slamanig. A general framework for redactable signatures and new constructions. In *ICISC*, pages 3–19, 2015.
26. D. Derler and D. Slamanig. Rethinking privacy for extended sanitizable signatures and a black-box construction of strongly private schemes. In *ProvSec*, pages 455–474, 2015.
27. M. Fischlin and P. Harasser. Invisible sanitizable signatures and public-key encryption are equivalent. In *ACNS*, pages 202–220, 2018.
28. N. Fleischhacker, J. Krupp, G. Malavolta, J. Schneider, D. Schröder, and M. Simkin. Efficient unlinkable sanitizable signatures from signatures with re-randomizable keys. In *PKC Part-I*, pages 301–330, 2016.
29. E. Ghosh, M. T. Goodrich, O. Ohrimenko, and R. Tamassia. Verifiable zero-knowledge order queries and updates for fully dynamic lists and trees. In *SCN*, pages 216–236, 2016.
30. E. Ghosh, O. Ohrimenko, and R. Tamassia. Zero-knowledge authenticated order queries and order statistics on a list. In *ACNS*, pages 149–171, 2015.
31. S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.
32. J. Gong, H. Qian, and Y. Zhou. Fully-secure and practical sanitizable signatures. In *Inscrypt*, pages 300–317, 2010.
33. S. Haber, Y. Hatano, Y. Honda, W. G. Horne, K. Miyazaki, T. Sander, S. Tezoku, and D. Yao. Efficient signature schemes supporting redaction, pseudonymization, and data deidentification. In *AsiaCCS*, pages 353–362, 2008.
34. T. Izu, N. Kunihiro, K. Ohta, M. Sano, and M. Takenaka. Sanitizable and deletable signature. In *WISA*, pages 130–144, 2008.
35. T. Izu, N. Kunihiro, K. Ohta, M. Sano, and M. Takenaka. Yet another sanitizable signature from bilinear maps. In *Ares*, pages 941–946, 2009.
36. R. Johnson, D. Molnar, D. Xiaodong Song, and D. A. Wagner. Homomorphic signature schemes. In *CT-RSA*, pages 244–262, 2002.
37. M. Klonowski and A. Lauks. Extended sanitizable signatures. In *ICISC*, pages 343–355, 2006.
38. H. Krawczyk and T. Rabin. Chameleon signatures. In *NDSS*, 2000.
39. S. Krenn, H. C. Pöhls, K. Samelin, and D. Slamanig. Chameleon-hashes with dual long-term trapdoors and their applications. In *AfricaCrypt*, pages 11–32, 2018.

40. S. Krenn, H. C. Pöhls, K. Samelin, and D. Slamanig. Protean signature schemes. In *CANS*, pages 256–276, 2018.
41. S. Krenn, K. Samelin, and D. Sommer. Stronger security for sanitizable signatures. In *DPM/QASA*, pages 100–117, 2015.
42. V. Kuchta and M. Manulis. Unique aggregate signatures with applications to distributed verifiable random functions. In *CANS*, pages 251–270, 2013.
43. A. Kundu and E. Bertino. Privacy-preserving authentication of trees and graphs. *Int. J. Inf. Sec.*, 12(6):467–494, 2013.
44. R. W. F. Lai, T. Zhang, S. S. M. Chow, and D. Schröder. Efficient sanitizable signatures without random oracles. In *ESORICS Part-I*, pages 363–380, 2016.
45. K. Miyazaki, G. Hanaoka, and H. Imai. Digitally signed document sanitizing scheme based on bilinear maps. In *AsiaCCS*, pages 343–354, 2006.
46. K. Miyazaki, M. Iwamura, T. Matsumoto, R. Sasaki, H. Yoshiura, S. Tezuka, and H. Imai. Digitally signed document sanitizing scheme with disclosure condition control. *IEICE Transactions*, 88-A(1):239–246, 2005.
47. E. Mykletun, M. Narasimha, and G. Tsudik. Signature bouquets: Immutability for aggregated/condensed signatures. In *ESORICS*, pages 160–176, 2004.
48. H. C. Pöhls and K. Samelin. Accountable redactable signatures. In *Ares*, pages 60–69, 2015.
49. H. C. Pöhls, K. Samelin, and J. Posegga. Sanitizable signatures in XML signature - performance, mixing properties, and revisiting the property of transparency. In *ACNS*, pages 166–182, 2011.
50. R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.
51. K. Samelin, H. C. Pöhls, A. Bilzhause, J. Posegga, and H. de Meer. On structural signatures for tree data structures. In *ACNS*, pages 171–187, 2012.
52. K. Samelin, H. C. Pöhls, A. Bilzhause, J. Posegga, and H. de Meer. Redactable signatures for independent removal of structure and content. In *ISPEC*, pages 17–33, 2012.
53. D. Slamanig and S. Rass. Generalizations and extensions of redactable signatures with applications to electronic healthcare. In *CMS*, pages 201–213, 2010.
54. R. Steinfeld, L. Bull, and Y. Zheng. Content extraction signatures. In *ICISC*, pages 285–304, 2001.
55. G. Traverso, D. Demirel, and J. A. Buchmann. *Homomorphic Signature Schemes - A Survey*. Springer Briefs in Computer Science. Springer, 2016.
56. R. Tsabary. An equivalence between attribute-based signatures and homomorphic signatures, and new constructions for both. In *TCC*, pages 489–518, 2017.
57. Z. Y. Wu, C.-W. Hsueh, C.-Y. Tsai, F. Lai, H.-C. Lee, and Y.-F. Chung. Redactable signatures for signed CDA documents. *J. Medical Systems*, 36(3):1795–1808, 2012.
58. D. H. Yum, J. W. Seo, and P. Joong Lee. Trapdoor sanitizable signatures made easy. In *ACNS*, pages 53–68, 2010.

A Proof of Theorem 1

We now provide the proof of security for Construction 1.

Proof. Correctness follows from inspection. Each security property is proven on its own. However, we already keep all queries and answers to and from the oracle. This does not change the view of the adversary.

Unforgeability To prove that our scheme is unforgeable, we use a sequence of games:

Game 0: The original unforgeability game.

Game 1: We now abort, if we draw a tag twice.

Transition - Game 0 → Game 1: Due to the birthday paradox, this can only happen with negligible probability. $|\Pr[S_0] - \Pr[S_1]| \leq \frac{q_s^2}{2^\lambda}$ follows, where q_s is the number of tags drawn. Note, this also means that no message under $\mathsf{pk}_{\mathsf{sig}}^{\mathsf{RS}}$ is signed twice.

Game 2: We now abort, if the adversary was able to generate a signature for $\mathsf{pk}_{\mathsf{sig}}^{\mathsf{RS}}$ which protects a message not signed by the signer.

Transition - Game 1 → Game 2: In this case, $\mathcal{A}$ returns $(m^*, (\sigma_a, c, \mathsf{pk}_\Sigma{}', (\tau_i)_{i\in[0..\ell]}))$ for which σ_a contains a signature not explicitly generated by the signer. We can use this forgery to construct an adversary $\mathcal{B}$ which breaks the unforgeability of the underlying Σ. The reduction works as follows. It receives the parameters and the public pk to forge and directly embeds them into the values given to the adversary $\mathcal{A}$. Queries to the signing oracle are answered honestly; all inner signatures to be generated are delegated to $\mathcal{B}$'s own oracle (with the exception of the ephemeral signature, which can generated honestly). Then, as by assumption

σ_a protects at least one message which was not signed by the signing oracle, $\mathcal{B}$ can return $(\{\mathcal{S} \cup (\{\mathsf{pk}_\Sigma{}', (c, \tau_0, \mathsf{pk}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}_\Sigma{}'))\}, m'^*, \sigma_a)$, where $\mathcal{S}$ is the set of all messages checked for the underlying aggregate w.r.t. to pk as derivable from the construction, but a single forged message m'^* (which can easily be spotted) which can be arbitrarily chosen from the set of forged messages. $|\Pr[S_1] - \Pr[S_2]| \leq \nu_{\mathsf{aggsig\text{-}unf}}(\lambda)$ follows, as, by assumption, at least m'^* must be fresh. Note, this also covers the case of changes public keys, tags and "mix-and-match" attacks by merging multiple signatures, as we already ruled out tag-collisions, while all public keys are always signed as well, and all signed values are bound to τ_0.

Game 3: We now abort, if the adversary was able to generate a new signature for $\mathsf{pk}_\Sigma{}'$ for the challenge $\mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}$, which was never generated by the challenger.

Transition - Game 2 → Game 3: In this case, $\mathcal{A}$ returns $(m^*, (\sigma_a, c, \mathsf{pk}_\Sigma{}', (\tau_i)_{i\in[0..\ell]}))$ for which σ_a contains a signature under $\mathsf{pk}_\Sigma{}'$ not explicitly generated by the signer. We can use this forgery to construct an adversary $\mathcal{B}$ which breaks the unforgeability of the underlying Σ. The reduction works as follows. It receives the parameters and the public pk to forge and directly embeds them into the values given to the adversary $\mathcal{A}$. Queries to the signing oracle are answered honestly; all inner signatures to be generated are delegated to $\mathcal{B}$'s own oracle (with the exception of the ephemeral signature, which can generated honestly). Then, as by assumption σ_a protects at least one message which was not signed by the signing oracle, $\mathcal{B}$ can return $(\{\mathcal{S} \cup (\{\mathsf{pk}_\Sigma{}', (c, \tau_0, \mathsf{pk}, \mathsf{pk}^*, \mathsf{pk}_\Sigma{}'))\}, m'^*, \sigma_a)$, where $\mathcal{S}$ is the set of all messages checked for the underlying aggregate w.r.t. to pk as derivable from the construction, but a single forged message m'^* (which can easily be spotted) which can be arbitrarily chosen from the set of forged messages. $|\Pr[S_2] - \Pr[S_3]| \leq q_s \nu_{\mathsf{aggsig\text{-}unf}}(\lambda)$, where q_s is the number of signatures generates, follows, as the reduction $\mathcal{B}$ has to guess where the adversary $\mathcal{A}$ forges a signature. Note, this also covers the case of changes public keys, tags and "mix-and-match" attacks by merging multiple signatures, as we already ruled out tag-collisions, while all public keys are always signed as well, and all signed values are bound to τ_0.

Game 3: As Game 2, but we abort, if the adversary was able to redact a non-redactable block.

Transition - Game 2 → Game 3: In this case, the adversary $\mathcal{A}$ was able to remove a signature from the aggregate, which should not be possible. Thus, this means that the adversary $\mathcal{A}$ was able to break the signature scheme. We show this by construction of an adversary $\mathcal{B}$ which uses $\mathcal{A}$ to break the no-extraction notion of the used Σ. The reduction works as follows. It receives the parameters and the public pk to forge. The public parameters are embedded honestly; all other values are generated as in the prior game. Next, $\mathcal{B}$ draws a random index $i \leftarrow_r [1..q_s]$, where q_s is an upper bound on the number of queries to the signing oracle. Then, every jth query to the signing oracle, where $i \neq j$, is answered honestly. On the ith query, however, $\mathcal{B}$ embeds the challenge pk and uses its own signing oracle to generate the signature σ_c. The other signatures can be generated honestly. Then, as by assumption σ_a contains a signature on the string $(c, \tau_0, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}_\Sigma{}')$, $\mathcal{B}$ can use to de-aggregate all other signatures from σ_a to obtain σ_c using its honestly generated $\mathsf{sk}^{\mathsf{RS}}_{\mathsf{sig}}$ and return $(\emptyset, \emptyset, (c, \tau_0, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}_\Sigma{}'), \sigma_c)$ as its own forgery. In the case that $\mathsf{pk}_\Sigma{}' \neq \mathsf{pk}$, $\mathcal{B}$ aborts. Note, we have already ruled out forgeries of never signed messages, while each messages signed is fresh due to the tags. $|\Pr[S_2] - \Pr[S_3]| \leq \nu_{\mathsf{aggsig\text{-}noExt}}(\lambda)$ follows.

Game 4: As Game 3, but we abort, if the adversary was able to generate a new aggregate σ_a which protects a set of messages returned by the challenger.

Transition - Game 3 → Game 4: In this case, the adversary $\mathcal{A}$ was able to break the uniqueness of the underlying signature scheme. The reduction works as follows. $\mathcal{B}$ receives the public parameters from its own challenger and embeds them accordingly. All other values are generated honestly and given to the adversary $\mathcal{A}$. Then, once the adversary $\mathcal{A}$ outputs $(m^*, \sigma^{\mathsf{RS}*})$, and by assumption, σ_a^* was never seen, but the messages protected by an honestly generated signature σ_a, $\mathcal{B}$ can directly return $(\mathcal{S}, \sigma_a, \sigma_a^*)$, where $\mathcal{S}$ is the set of public key/messages protected in the aggregate which can be derived as in the construction. $|\Pr[S_3] - \Pr[S_4]| \leq \nu_{\mathsf{aggsig\text{-}unique}}(\lambda)$ follows.

Game 5: As Game 4, but we abort, if the adversary was able to exchange a signature on $(c, \tau_0, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}, \mathsf{pk}_\Sigma{}')$ on the aggregate from some already seen aggregate.

Transition - Game 4 → Game 5: If an adversary outputs $(m^*, \sigma^{\mathsf{RS}*})$, where $\sigma^{\mathsf{RS}*} = (\sigma_a, c, \mathsf{pk}_\Sigma{}', (\tau_i)_{i\in[0..\ell]})$, meeting the above winning conditions, we can construct an adversary $\mathcal{B}$ which breaks the No-Extraction property of the underlying Σ. It proceeds as follows. It first receives the public parameters. It then queries its own challenge oracle to obtain a long-term public-key pk. Both are embedded honestly; other values are generated honestly. For every signing query, $\mathcal{B}$ first requests an additional key pk' if $\mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}$ is the challenge one. If this is not the case, $\mathcal{B}$ generates one honestly. It then proceeds as in the signing algorithm, but requests a full aggregate on all signatures generated under the keys generated. If $\mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}$ is not the challenge one, all signatures which are related to redacting are queried to the challenge oracle and embedded for the adversary. If $\mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}$ is the challenge one, it also gets all signatures, but the one for σ_0 and σ_c. For redaction, if $\mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}$ and $\mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}$ are the challenge ones, $\mathcal{B}$ requests a complete new aggregate signature on the redacted message. Then, whenever $\mathcal{A}$ outputs $\sigma^{\mathsf{RS}*}$ meeting the winning requirements, $\mathcal{B}$ can simply output $(\mathcal{S}, \sigma_a)$, where $\mathcal{S}$ is as in the verification algorithm. $|\Pr[S_4] - \Pr[S_5]| \leq \nu_{\mathsf{aggsig\text{-}noExt}}(\lambda)$ follows.

Now, the adversary can no longer win the unforgeability game. Moreover, each hop changes the view of the adversary only negligibly, concluding the proof.

Immutability To prove that our scheme is immutable, we use a sequence of games:

Game 0: The original immutability game.

Game 1: We now abort, if we draw a tag twice.

Transition - Game 0 → Game 1: Due to the birthday paradox, this can only happen with negligible probability. $|\Pr[S_0] - \Pr[S_1]| \leq \frac{q_s^2}{2^\lambda}$ follows, where q_s is the number of key pairs generated. Note, this also means that no message under $\mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}$ is signed twice.

Game 2: We now abort, if the adversary was able to generate a signature for $\mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}$ which protects a message not signed by the signer.

Transition - Game 1 → Game 2: In this case, $\mathcal{A}$ returns $(m^*, (\sigma_a, c, \mathsf{pk}_\Sigma{}', (\tau_i)_{i\in[0..\ell]}), \mathsf{pk}^*)$ for which σ_a contains a signature not explicitly generated by the signer. We can use this forgery to construct an adversary $\mathcal{B}$ which breaks the unforgeability of the underlying Σ. The reduction works as follows. It receives the parameters and the public pk to forge and directly embeds them into the values given to the adversary $\mathcal{A}$. Queries to the signing oracle are answered honestly; all inner signatures to be generated are delegated to $\mathcal{B}$'s own oracle (with the exception of the ephemeral signature, which can generated honestly). Then, as by assumption σ_a protects at least one message which was not signed by the signing oracle, $\mathcal{B}$ can return $(\{\mathcal{S} \cup (\{\mathsf{pk}_\Sigma{}', (c, \tau_0, \mathsf{pk}, \mathsf{pk}^*, \mathsf{pk}_\Sigma{}'))\}, m'^*, \sigma_a)$, where $\mathcal{S}$ is the set of all messages checked for the underlying aggregate w.r.t. to pk as derivable from the construction, but a single forged message m'^* (which can easily be spotted) which can be arbitrarily chosen from the set of forged messages. $|\Pr[S_1] - \Pr[S_2]| \leq \nu_{\mathsf{aggsig\text{-}unf}}(\lambda)$ follows, as, by assumption, at least m'^* must be fresh. Note, this also covers the case of changed public keys, tags and "mix-and-match" attacks by merging multiple signatures, as we already ruled out tag-collisions, while all public keys are always signed as well, and all signed values are bound to τ_0.

Game 3: As Game 2, but we abort, if the adversary was able to redact a non-redactable block.

Transition - Game 2 → Game 3: In this case, the adversary $\mathcal{A}$ was able to remove a signature from the aggregate, which should not be possible. Thus, this means that the adversary $\mathcal{A}$ was able to break the signature scheme. We show this by construction of an adversary $\mathcal{B}$ which uses $\mathcal{A}$ to break the no-extraction notion of the used Σ. The reduction works as follows. It receives the parameters and the public pk to forge and directly embeds them into the values given to the adversary $\mathcal{A}$. Queries to the signing oracle are answered honestly; all inner signatures to be generated are delegated to $\mathcal{B}$'s own oracle as a bulk (with the exception of the ephemeral signature, which can generated honestly). Then, as by assumption σ_a protects less messages as given as aggregate by the signing oracle, $\mathcal{B}$ can return $(\mathcal{S}, \sigma'_a)$, where $\mathcal{S}$ is the set of all messages checked for the underlying aggregate w.r.t. to pk as derivable from the construction. The signature σ_c can be removed using $\mathsf{sk}_\Sigma{}'$, generating σ'_a, as we have already ruled out forgeries of never

signed messages, while each messages signed is fresh due to the tags. $|\Pr[S_2] - \Pr[S_3]| \leq \nu_{\mathsf{aggsig\text{-}noExt}}(\lambda)$ follows.

Now, the adversary can no longer win the immutability game. Moreover, each hop changes the view of the adversary only negligibly, concluding the proof.

Privacy To prove that our scheme is private, we use a sequence of games:

Game 0: The original privacy game in the case $b = 0$.
Game 1: We now switch to $b = 1$.
Transition - Game 0 → Game 1: As the signatures are distributed exactly the same, our scheme is private in an information theoretical sense. Thus, $|\Pr[S_0] - \Pr[S_1]| = 0$ follows.

Transparency To prove that our scheme is transparent, we use a sequence of games:

Game 0: The original transparency game in the case $b = 0$.
Game 1: We now switch to $b = 1$.
Transition - Game 0 → Game 1: As the signatures are distributed exactly the same, our scheme is transparent in an information theoretical sense. Thus, $|\Pr[S_0] - \Pr[S_1]| = 0$ follows.

Invisibility To prove that our scheme is invisible, we use a sequence of games:

Game 0: The original invisibility game.
Game 1: We now abort, if the adversary queries some $(\sigma^{\mathsf{RS}}, m)$ for the challenge public keys which verifies, but was never returned by either LoRADM or $\mathsf{Red}^{\mathsf{RS}\prime}$.
Transition - Game 0 → Game 1: Given this adversary $\mathcal{A}$, we can construct an adversary $\mathcal{B}$ which breaks the unforgeability of the RS. The reduction works as follows. First, it draws a random bit $b \leftarrow_r \{0, 1\}$. Then, it receives $\mathsf{pk}^{\mathsf{RS}}_{\mathsf{sig}}$ and $\mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}$ (along with the parameters) and then passes those keys to the adversary. Every redaction query is done using the $\mathsf{Red}^{\mathsf{RS}\prime}$ provided (imposing the limitation the invisibility game gives). Likewise, queries to LoRADM are answered by using the $\mathsf{Sign}^{\mathsf{RS}\prime}$ provided, but using $\mathbb{A}^{\mathsf{RS}}_b$ in the case the challenge $\mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}$ is queried. Then, whenever $\mathcal{A}$ queries $(\sigma^{\mathsf{RS}}, m)$ for the challenge $\mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}$, $\mathcal{B}$ can simply return $(\sigma^{\mathsf{RS}}, m)$ as its own forgery, as, by assumption, $(\sigma^{\mathsf{RS}}, m)$ was not seen before. Thus, $|\Pr[S_0] - \Pr[S_1]| \leq \nu_{\mathsf{rss\text{-}unf}}(\lambda)$ follows.
Game 2: We now start replace *each* ciphertext generated for the challenge $\mathsf{pk}^{\mathsf{RS}}_{\mathsf{san}}$ with an encryption of 0 (with the appropriate length).
Transition - Game 1 → Game 2: Assume that the adversary can distinguish this replacement with a non-negligible probability. We can then construct a reduction $\mathcal{B}$ which uses $\mathcal{A}$ to break the IND-CCA2 security of the underlying encryption scheme. The reduction works via a series of hybrids. Our reduction $\mathcal{B}$ proceeds as follows. It receives pk and (and the corresponding parameters) from its own challenger and embeds them correctly. All other values are generated as in Game 1. For the first i ciphertexts generated, encrypt a 0. If, however, the ith ciphertext is generated, $\mathcal{B}$ asks its own challenge oracle to either encrypt 0 or the correct value. The response is embedded to $\mathcal{B}$'s response to $\mathcal{A}$. All following ciphertexts are generated honestly. Thus, Game 2.0 is the same as Game 1 while in Game 2.1., however, we make the first replacement. Then, whatever $\mathcal{A}$ outputs in Game 2.i is also output by $\mathcal{B}$. Note, decryption queries for ciphertexts generated by the adversary can be queried to decryption oracle provided; the content for all other ciphertexts are known and thus the ciphertexts do not need to be decrypted at all. Thus, $|\Pr[S_1] - \Pr[S_2]| \leq q\nu_{\mathsf{ind\text{-}cca2}}(\lambda)$ follows, where q is the number of ciphertexts generated.

As now the game is independent of the bit b, invisibility is proven.

B Proof of Theorem 2

We now provide the proof of security for Construction 2.

Proof. Correctness follows from inspection. Each security property is proven on its own. However, we already keep all queries and answers to and from the oracle. This does not change the view of the adversary.

Unforgeability To prove that our scheme is unforgeable, we use a sequence of games:

Game 0: The original unforgeability game.

Game 1: We now abort, if we draw an x twice.

Transition - Game 0 → Game 1: Due to the birthday paradox, this can only happen with negligible probability. $|\Pr[S_0] - \Pr[S_1]| \leq \frac{q_s^2}{2^\lambda}$ follows, where q_s is the number of key pairs generated.

Game 2: We now start replace *each* ciphertext generated for the challenge $\mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}$ with an encryption of 0 (with the appropriate length).

Transition - Game 1 → Game 2: Assume that the adversary can distinguish this replacement with a non-negligible probability. We can then construct a reduction $\mathcal{B}$ which uses $\mathcal{A}$ to break the IND-CCA2 security of the underlying encryption scheme. The reduction works via a series of hybrids. Our reduction $\mathcal{B}$ proceeds as follows. It receives pk and (and the corresponding parameters) from its own challenger and embeds them correctly. All other values are generated as in Game 1. For the first i ciphertexts generated, encrypt a 0. If, however, the ith ciphertext is generated, $\mathcal{B}$ asks its own challenge oracle to either encrypt 0 or the correct value. The response is embedded to $\mathcal{B}$'s response to $\mathcal{A}$. All following ciphertexts are generated honestly. Thus, Game 2.0 is the same as Game 1 while in Game 2.1., however, we make the first replacement. Then, whatever $\mathcal{A}$ outputs in Game 2.i is also output by $\mathcal{B}$. Note, decryption queries for ciphertexts generated by the adversary can be queried to decryption oracle provided; the content for all other ciphertexts are known and thus the ciphertexts do not need to be decrypted at all. Thus, $|\Pr[S_1] - \Pr[S_2]| \leq q\nu_{\mathsf{ind\text{-}cca2}}(\lambda)$ follows, where q is the number of ciphertexts generated.

Game 3: We now replace all r_i with a purely random value $r_i \leftarrow_r \{0,1\}^\lambda$.

Transition - Game 2 → Game 3: An adversary distinguishing this replacement can be turned into an adversary against the pseudo-randomness of the PRF. We prove this via a series of hybrids. Let Game 3.0 the same as Game 2. In Game 3.i $\mathcal{B}$ uses its $\mathsf{Eval}'_{\mathsf{PRF}}$ oracle to generate the random coins. All other values are generated as in prior game. Then, whatever $\mathcal{A}$ outputs, is also output by $\mathcal{B}$. $|\Pr[S_2] - \Pr[S_3]| \leq q_s\nu_{\mathsf{prf\text{-}pr}}(\lambda)$ follows, where q_s is the number of calls to the signature-generation oracle.

Game 4: We now abort, if we draw some random coins twice.

Transition - Game 3 → Game 4: Due to the birthday paradox, this can only happen with negligible probability. $|\Pr[S_3] - \Pr[S_4]| \leq \frac{q_s^2}{2^\lambda}$ follows, where q_s is the number of key pairs generated.

Game 5: We now abort, if the adversary was able to generate a signature on a string of public keys and ciphertexts not signed.

Transition - Game 4 → Game 5: In this case, the adversary $\mathcal{A}$ was able to generate a signature σ_s (contained in $\sigma^{\mathsf{SS}*}$) on $(\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}, (\mathsf{pk}_i)_{[1..\ell]}, c)$ which was never generated by the signer. We can use this to construct a reduction $\mathcal{B}$ which forges a signature of the underlying Σ. The reduction works as follows. It receives the pk (and the corresponding parameters) to forge, and embeds it accordingly into the parameters/public key. Everything else is generated honestly. For every signature generated, $\mathcal{B}$ queries its signature-generation oracle; this signature is then embedded in the response. Finally, once $\mathcal{A}$ outputs its forgery, $\mathcal{B}$ can return $(\emptyset, \emptyset, (\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}, (\mathsf{pk}_i)_{[1..\ell]}, c), \sigma_s)$ as its own forgery. $|\Pr[S_4] - \Pr[S_5]| \leq \nu_{\mathsf{aggsig\text{-}unf}}(\lambda)$ follows.

Game 6: We now abort, if the adversary was able to generate a new signature σ'_s on a string of public keys already signed.

Transition - Game 5 → Game 6: In this case, the adversary $\mathcal{A}$ was able to generate a new signature σ'_s (contained in $\sigma^{\mathsf{SS}*}$) on $(\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}, (\mathsf{pk}_i)_{[1..\ell]}, c)$ which was never generated by the signer, but σ_s was. We

can use this to construct a reduction $\mathcal{B}$ which breaks the uniqueness of the underlying Σ. The reduction works as follows. It receives the corresponding parameters of the Σ to forge, and embeds it accordingly into the parameters. Everything else is generated honestly. Finally, once $\mathcal{A}$ outputs its forgery, $\mathcal{B}$ can return $(\{(\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, (\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}, (\mathsf{pk}_i)_{[1..\ell]}, c))\}, \sigma_s, \sigma'_s)$ as its own forgery. $|\Pr[S_5] - \Pr[S_6]| \leq \nu_{\mathsf{aggsig\text{-}unique}}(\lambda)$ follows.

Game 7: We now abort, if the adversary was able to generate a new inner signature σ'_i on a string signed before.

Transition - Game 6 → Game 7: In this case, the adversary $\mathcal{A}$ was able to generate a new signature σ'_i (contained in $\sigma^{\mathsf{SS}*}$) on on some string $((\mathsf{pk}_i, m^i)_{[1..\ell]}, c, \sigma_s, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}})$ which was signed before, but $\sigma_i \neq \sigma'_i$ holds. but σ_s was. We can use this to construct a reduction $\mathcal{B}$ which breaks the uniqueness of the underlying Σ. The reduction works as follows. It receives the corresponding parameters of the Σ to forge, and embeds it accordingly into the parameters. Everything else is generated honestly. Finally, once $\mathcal{A}$ outputs its forgery, $\mathcal{B}$ can return $(\{(\mathsf{pk}_i, ((\mathsf{pk}_i, m^i)_{[1..\ell]}, c, \sigma_s, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}))\}, \sigma_i, \sigma'_i)$ as its own forgery, where pk_i is the corresponding public key. $|\Pr[S_6] - \Pr[S_7]| \leq \nu_{\mathsf{aggsig\text{-}unique}}(\lambda)$ follows.

Game 8: We now abort, if the adversary outputs a validating $(m^*, \sigma^{\mathsf{SS}*})$, where $\sigma^{\mathsf{SS}*} = (c, \sigma_s, (\mathsf{pk}_i, \sigma_i)_{i\in[1..\ell]})$, where m^* was never returned from any query to the signing or sanitization oracle.

Transition - Game 7 → Game 8: In this case, there must be a block m^{i*} which was changed, but the adversary never saw a signature for that block. We can use this to break the unforgeability of the underlying signature scheme. Our reduction $\mathcal{B}$ works as follows. Let q_s be an upper bound on the number of signature key pairs created. First, $\mathcal{B}$ randomly selects an index $i \leftarrow_r [1..q_s]$. It receives the pk (and the corresponding parameters) to forge. The parameters are embedded honestly. Everything else is generated honestly. Then, once the ith inner signature is generated, $\mathcal{B}$ embeds pk and uses its own signature-generation oracle to receive the corresponding signature. The result is embedded honestly. The same is true for sanitization: for every change, $\mathcal{B}$ queries the signature-generation oracle to obtain a new signature. As, by assumption, at least one block must be fresh, but $\mathcal{B}$ needs to guess where this happens, $|\Pr[S_7] - \Pr[S_8]| \leq q_s \nu_{\mathsf{aggsig\text{-}unf}}(\lambda)$ follows.

Now, the adversary can no longer win the unforgeability game. Moreover, each hop changes the view of the adversary only negligibly, concluding the proof.

Immutability To prove that our scheme is immutable, we use a sequence of games:

Game 0: The original immutability game.

Game 1: We now abort, if an ephemeral public key (for which the corresponding secret keys are not given to the sanitizer) was drawn twice.

Transition - Game 0 → Game 1: Due to the birthday paradox, this can only happen with negligible probability. $|\Pr[S_0] - \Pr[S_1]| \leq \frac{q_s^2}{2^\lambda}$ follows, where q_s is the number of key pairs generated.

Game 2: We now abort, if the adversary outputs a validating $(m^*, \sigma^{\mathsf{SS}*}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}{}^*)$, for which the ephemeral public keys or c have not been signed in that particular order.

Transition - Game 1 → Game 2: In this case, the adversary $\mathcal{A}$ was able to generate a signature σ_s (contained in $\sigma^{\mathsf{SS}*}$) on $(\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^*, (\mathsf{pk}_i)_{[1..\ell]}, c)$ which was never generated by the signer. We can use this to construct a reduction $\mathcal{B}$ which forges a signature of the underlying Σ. The reduction works as follows. It receives the pk (and the corresponding parameters) to forge, and embeds it accordingly into the parameters/public key. Everything else is generated honestly. For every signature generated, $\mathcal{B}$ queries its signature-generation oracle; this signature is then embedded in the response. Finally, once $\mathcal{A}$ outputs its forgery, $\mathcal{B}$ can return $(\emptyset, \emptyset, (\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^*, (\mathsf{pk}_i)_{[1..\ell]}, c), \sigma_s)$ as its own forgery. $|\Pr[S_1] - \Pr[S_2]| \leq \nu_{\mathsf{aggsig\text{-}unf}}(\lambda)$ follows.

Game 3: We now abort, if the adversary outputs a validating $(m^*, \sigma^{\mathsf{SS}*}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}{}^*)$, where $\sigma^{\mathsf{SS}*} = (c, \sigma_s, (\mathsf{pk}_i, \sigma_i)_{i\in[1..\ell]})$, where m^* was never derivable from any query to the signing oracle.

Transition - Game 2 → Game 3: In this case, there must be a block m^{i*} which was changed, but the adversary does not have the corresponding secret key. Our reduction $\mathcal{B}$ works as follows. Let q_s be an upper bound on the number of signature key pairs created for which the sanitizer does not receive the secret key, breaking the unforgeability of the underlying Σ. First, $\mathcal{B}$ randomly selects an index $i \leftarrow_r [1..q_s]$. It receives the pk (and the corresponding parameters) to forge. The parameters are embedded honestly. Everything else is generated honestly. Then, once the ith inner signature, for which the sanitizer does not receive the secret key, is generated, $\mathcal{B}$ embeds pk and uses its own signature-generation oracle to receive the corresponding signature. The result is embedded honestly. As, by assumption, at least one block must be fresh, but $\mathcal{B}$ needs to guess where this happens, $|\Pr[S_2] - \Pr[S_3]| \leq q_s \nu_{\mathsf{aggsig\text{-}unf}}(\lambda)$ follows.

Now, the adversary can no longer win the immutability game. Moreover, each hop changes the view of the adversary only negligibly, concluding the proof.

Privacy To prove that our scheme is private, we use a sequence of games:

Game 0: The original privacy game in the case $b = 0$.
Game 1: We now switch to $b = 1$.
Transition - Game 0 → Game 1: As the signatures are distributed exactly the same, our scheme is private in an information theoretical sense. Thus, $|\Pr[S_0] - \Pr[S_1]| = 0$ follows.

Transparency To prove that our scheme is transparent, we use a sequence of games:

Game 0: The original transparency game in the case $b = 0$.
Game 1: We now switch to $b = 1$.
Transition - Game 0 → Game 1: As the signatures are distributed exactly the same, our scheme is transparent in an information theoretical sense. Thus, $|\Pr[S_0] - \Pr[S_1]| = 0$ follows.

Invisibility To prove that our scheme is invisible, we use a sequence of games:

Game 0: The original invisibility game.
Game 1: We now abort, if the adversary queries some $(\sigma^{\mathsf{SS}}, m)$ for the challenge public keys which verifies, but was never returned by either LoRADM or $\mathsf{Sanit}^{\mathsf{SS}\prime}$.
Transition - Game 0 → Game 1: Given this adversary $\mathcal{A}$, we can construct an adversary $\mathcal{B}$ which breaks the unforgeability of the SS. The reduction works as follows. First, it draws a random bit $b \leftarrow_r \{0, 1\}$. Then, it receives $\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}$ and $\mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}$ and then passes those keys to the adversary. Every redaction query is done using the $\mathsf{Sanit}^{\mathsf{SS}\prime}$ provided (imposing the limitation the invisibility game gives). Likewise, queries to LoRADM are answered by using the $\mathsf{Sign}^{\mathsf{SS}\prime}$ provided, but using $\mathbb{A}^{\mathsf{SS}}_b$ in the case the challenge $\mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}$ is queried. Then, whenever $\mathcal{A}$ queries $(\sigma^{\mathsf{SS}}, m)$ for the challenge $\mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}$, $\mathcal{B}$ can simply return $(\sigma^{\mathsf{SS}}, m)$ as its own forgery, as, by assumption, $(\sigma^{\mathsf{SS}}, m)$ was not seen before. Thus, $|\Pr[S_0] - \Pr[S_1]| \leq \nu_{\mathsf{sss\text{-}unf}}(\lambda)$ follows.
Game 2: We now abort, if we draw an x twice.
Transition - Game 0 → Game 1: Due to the birthday paradox, this can only happen with negligible probability. $|\Pr[S_1] - \Pr[S_2]| \leq \frac{q_s^2}{2^\lambda}$ follows, where q_s is the number of key pairs generated.
Game 3: We now start replace *each* ciphertext generated for the challenge $\mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}}$ with an encryption of 0 (with the appropriate length).
Transition - Game 2 → Game 3: Assume that the adversary can distinguish this replacement with a non-negligible probability. We can then construct a reduction $\mathcal{B}$ which uses $\mathcal{A}$ to break the IND-CCA2 security of the underlying encryption scheme. The reduction works via a series of hybrids. Our reduction $\mathcal{B}$ proceeds as follows. It receives pk and (and the corresponding parameters) from its own challenger and embeds them correctly. All other values are generated as in Game 2. For the first i ciphertexts generated, encrypt a 0. If, however, the ith ciphertext is generated, $\mathcal{B}$ asks its own challenge oracle to either encrypt 0 or the correct

value. The response is embedded to $\mathcal{B}$'s response to $\mathcal{A}$. All following ciphertexts are generated honestly. Thus, Game 3.0 is the same as Game 2 while in Game 3.1., however, we make the first replacement. Then, whatever $\mathcal{A}$ outputs in Game 3.i is also output by $\mathcal{B}$. Note, decryption queries for ciphertexts generated by the adversary can be queried to decryption oracle provided; the content for all other ciphertexts are known and thus the ciphertexts do not need to be decrypted at all. Thus, $|\Pr[S_2] - \Pr[S_3]| \leq q\nu_{\mathsf{ind\text{-}cca2}}(\lambda)$ follows, where q is the number of ciphertexts generated.

Game 4: We now replace all r_i with a purely random value $r_i \leftarrow_r \{0,1\}^\lambda$.

Transition - Game 3 → Game 4: An adversary distinguishing this replacement can be turned into an adversary against the pseudorandomness of the PRF. We prove this via a series of hybrids. Let Game 4.0 the same as Game 3. In Game 4.i $\mathcal{B}$ uses its $\mathsf{Eval}'_{\mathsf{PRF}}$ oracle to generate the random coins. All other values are generated as in prior game. Then, whatever $\mathcal{A}$ outputs, is also output by $\mathcal{B}$. $|\Pr[S_3] - \Pr[S_4]| \leq q_s\nu_{\mathsf{prf\text{-}pr}}(\lambda)$ follows, where q_s is the number of calls to the signature-generation oracle.

Game 5: We now abort, if we draw an r_i twice.

Transition - Game 4 → Game 5: Due to the birthday paradox, this can only happen with negligible probability. $|\Pr[S_4] - \Pr[S_5]| \leq \frac{q_s^2}{2^\lambda}$ follows, where q_s is the number of random coins drawn.

As now the game is independent of the bit b, invisibility is proven.

C Proof of Theorem 3

We now provide the proof of security for Construction 3.

Proof. Correctness follows from inspection. Each security property is proven on its own. However, we already keep all queries and answers to and from the oracle. This does not change the view of the adversary. We also directly generate any keys required, but not received by the reduction's own challenger, honestly, also embedding them, without mentioning it, to shorten the proof.

Unforgeability To prove that our scheme is unforgeable, we use a sequence of games:

Game 0: The original unforgeability game.

Game 1: We now abort, if the adversary outputs $(m, \sigma^{\mathsf{PS}})$, where $\sigma^{\mathsf{PS}} = ((\sigma_i^{\mathsf{SS}})_{0\leq i\leq \ell_m}, \sigma^{\mathsf{RS}}, c, \tau, (\tau_i)_{1\leq i\leq \ell_m})$, where any $(m, \sigma^{\mathsf{RS}}, c, (\tau_i, \sigma_i^{\mathsf{SS}})_{1\leq i\leq \ell_m}, \tau, \mathsf{pk}_{\mathsf{sig}}^{\mathsf{PS}}, \mathsf{pk}_{\mathsf{san}}^{\mathsf{PS}})$, protected by σ_0^{SS}, has never been returned by the challenger.

Transition - Game 0 → Game 1: In this case, $((m, \sigma^{\mathsf{RS}}, c, (\tau_i, \sigma_i^{\mathsf{SS}})_{1\leq i\leq \ell_m}, \tau, \mathsf{pk}_{\mathsf{sig}}^{\mathsf{PS}}, \mathsf{pk}_{\mathsf{san}}^{\mathsf{PS}}), \sigma_0^{\mathsf{SS}})$ is a valid forgery of the outer SS. A reduction is simple. Namely, the reduction $\mathcal{B}$ receives the challenge keys $\mathsf{pk}_{\mathsf{sig}}^{\mathsf{SS}\prime}$ and $\mathsf{pk}_{\mathsf{san}}^{\mathsf{SS}\prime}$ from its own challenger, and embeds them into $\mathsf{pk}_{\mathsf{sig}}^{\mathsf{PS}}$ and $\mathsf{pk}_{\mathsf{san}}^{\mathsf{PS}}$. Every underlying signing and sanitization request for the SSs is performed by the reduction's oracles. As, by assumption, the message protected by σ_0^{SS} must be fresh, it thus breaks the unforgeability of the underlying SS in any case. Thus, $|\Pr[S_0] - \Pr[S_1]| \leq \nu_{\mathsf{sss\text{-}unf}}(\lambda)$ follows.

Game 2: We now abort, if the adversary outputs $(m, \sigma^{\mathsf{PS}})$, where $\sigma^{\mathsf{PS}} = ((\sigma_i^{\mathsf{SS}})_{0\leq i\leq \ell_m}, \sigma^{\mathsf{RS}}, \tau, (\tau_i)_{1\leq i\leq \ell_m})$, where any $((\sigma_i^{\mathsf{SS}})_{0\leq i\leq \ell_m}, \sigma^{\mathsf{RS}}, \tau, (\tau_i)_{1\leq i\leq \ell_m})$ protected by σ_0^{SS} was returned by the challenger, but σ_0^{SS} was never created by the challenger.

Transition - Game 1 → Game 2: In this case, $(((\sigma_i^{\mathsf{SS}})_{0\leq i\leq \ell_m}, \sigma^{\mathsf{RS}}, \tau, (\tau_i)_{1\leq i\leq \ell_m}), \sigma_0^{\mathsf{SS}})$ is a valid forgery of the outer SS. The reduction works as in the prior hop. As, by assumption, the message protected by σ_0^{SS} must be fresh, it thus breaks the unforgeability of the underlying SS in any case. Thus, $|\Pr[S_1] - \Pr[S_2]| \leq \nu_{\mathsf{sss\text{-}unf}}(\lambda)$ follows.

Now, the adversary can no longer win the unforgeability game, as also each public key is bound to a tag. Moreover, each hop changes the view of the adversary only negligibly, concluding the proof.

Immutability To prove that our scheme is immutable, we use a sequence of games:

Game 0: The original immutability game.

Game 1: We now abort, if the challenger draws a tag twice.

Transition - Game 0 → Game 1: The probability that this event happens is bounded by the birthday paradox. $|\Pr[S_0] - \Pr[S_1]| \leq q_t^2/2^\lambda$ follows, where q_t is the number of drawn tags.

Game 2: We now abort, if the adversary outputs $(m, \sigma^{\mathsf{PS}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}})$, where $\sigma^{\mathsf{PS}} = ((\sigma_i^{\mathsf{SS}})_{0\leq i\leq \ell_m}, \sigma^{\mathsf{RS}}, \tau, (\tau_i)_{1\leq i\leq \ell_m})$, but $\mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}$ was never signed by the signing oracle w.r.t. to τ.

Transition - Game 1 → Game 2: This breaks the immutability property of the outer SS. The reduction proceeds as follows. It receives $\mathsf{pk}^{\mathsf{SS}\prime}_{\mathsf{sig}}$ from its own challenger and embeds it into $\mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}$. Then, every signing query is performed by the reduction's own oracles. Then, after $\mathcal{A}$ returned $(m, \sigma^{\mathsf{PS}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}})$, $(m', \sigma_0^{\mathsf{SS}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}})$ with $m' = (m, \sigma^{\mathsf{RS}}, (\tau_i, \sigma_i^{\mathsf{SS}})_{1\leq i\leq \ell_m}, \tau, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}})$ is a valid forgery. $|\Pr[S_1] - \Pr[S_2]| \leq \nu_{\mathsf{sss\text{-}imm}}(\lambda)$ follows.

Game 3: We now abort, if the adversary outputs $(m, \sigma^{\mathsf{PS}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}})$, where $\sigma^{\mathsf{PS}} = ((\sigma_i^{\mathsf{SS}})_{0\leq i\leq \ell_m}, \sigma^{\mathsf{RS}}, \tau, (\tau_i)_{1\leq i\leq \ell_m})$, but τ was never drawn by the challenger.

Transition - Game 2 → Game 3: As τ is non-admissible, the adversary was able to generate a signature not derivable, breaking the immutability of the outer SS. The reduction works exactly as in the prior game. $|\Pr[S_2] - \Pr[S_3]| \leq \nu_{\mathsf{sss\text{-}imm}}(\lambda)$ follows.

Game 4: We now abort, if the adversary outputs $(m, \sigma^{\mathsf{PS}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}})$, where $\sigma^{\mathsf{PS}} = ((\sigma_i^{\mathsf{SS}})_{0\leq i\leq \ell_m}, \sigma^{\mathsf{RS}}, \tau, (\tau_i)_{1\leq i\leq \ell_m})$, but some τ_i was never signed by the challenger w.r.t. τ or the ordering is inconsistent.

Transition - Game 3 → Game 4: As each τ_i is signed by the RS, the adversary was able to generate a forgery of the RS. It receives $\mathsf{pk}^{\mathsf{RS}\prime}_{\mathsf{sig}}$ from its own challenger and embeds it into $\mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}$. Then, every signing query is performed by the reduction's own oracles. Then, $((\tau_1, \tau_2, \ldots, \tau_\ell, \tau, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}), \sigma^{\mathsf{RS}})$ is a valid forgery. $|\Pr[S_4] - \Pr[S_5]| \leq \nu_{\mathsf{rss\text{-}unf}}(\lambda)$ follows.

Game 5: We now abort, if the adversary outputs $(m, \sigma^{\mathsf{PS}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}})$, where $\sigma^{\mathsf{PS}} = ((\sigma_i^{\mathsf{SS}})_{0\leq i\leq \ell_m}, \sigma^{\mathsf{RS}}, \tau, (\tau_i)_{1\leq i\leq \ell_m})$, but it was able to redact a block not marked as redactable.

Transition - Game 4 → Game 5: Note, we already ruled out tag-collisions, and thus the messages are uniquely identifiable. The reduction is same as in the prior hop. $|\Pr[S_4] - \Pr[S_5]| \leq \nu_{\mathsf{rss\text{-}unf}}(\lambda)$ follows.

Game 6: We now abort, if the adversary outputs $(m, \sigma^{\mathsf{PS}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}})$, where $\sigma^{\mathsf{PS}} = ((\sigma_i^{\mathsf{SS}})_{0\leq i\leq \ell_m}, \sigma^{\mathsf{RS}}, \tau, (\tau_i)_{1\leq i\leq \ell_m})$, but it was able to sanitize a block with tag τ_i which was not marked as sanitizable.

Transition - Game 5 → Game 6: Note, we already ruled out tag-collisions and thus the messages are uniquely identifiable. The reduction is same as in Game 3, but the reduction returns $((m^i, \tau, \tau_i), \sigma_i^{\mathsf{SS}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}})$. $|\Pr[S_5] - \Pr[S_6]| \leq \nu_{\mathsf{sss\text{-}imm}}(\lambda)$ follows.

Now, the adversary can no longer win the immutability game. Moreover, each hop changes the view of the adversary only negligibly, concluding the proof.

Privacy To prove that our scheme is private, we use a sequence of games:

Game 0: The original privacy game, where $b = 0$.

Game 1: Instead of signing $(m_0^i, \tau, \tau_i, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}})$ in the inner SSs and adjusting them to $(m'^i, \tau, \tau_i, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}})$, sign $(m_1^i, \tau, \tau_i, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}})$ and adjust accordingly.

Transition - Game 1 → Game 2: Assume that the adversary can distinguish these two games. We can then construct a reduction $\mathcal{B}$ which uses the adversary $\mathcal{A}$ to break the privacy of the underlying SS'. Namely, $\mathcal{B}$ proceeds as follows. It receives $\mathsf{pk}^{\mathsf{SS}\prime}_{\mathsf{sig}}$ and $\mathsf{pk}^{\mathsf{SS}\,\prime}_{\mathsf{san}}$, and embeds them into $\mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}$ and $\mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}$. Then, every signing, editing and proof oracle queries are answered by $\mathcal{B}$'s own oracles. However, for the calls to the $\mathsf{LoREdit}$ oracle, the calls for the SSs are redirected to the LoRSan oracle and the result embedded to the answer. Clearly, the simulation is perfect. Then, whatever $\mathcal{A}$ outputs is also output by $\mathcal{B}$. $|\Pr[S_0] - \Pr[S_1]| \leq \nu_{\mathsf{sss'\text{-}priv}}$ follows.

Game 2: Instead of signing $(m_0, \sigma^{\mathsf{RS}}, (\tau_i, \sigma_i^{\mathsf{SS}})_{1\le i\le \ell_m}, \tau, \mathsf{pk}_{\mathsf{sig}}^{\mathsf{PS}}, \mathsf{pk}_{\mathsf{san}}^{\mathsf{PS}})$ in the outer SSs and adjusting them to $(m', \sigma^{\mathsf{RS}\prime}, (\tau_i, \sigma_i^{\mathsf{SS}})_{1\le i\le \ell_m}, \tau, \mathsf{pk}_{\mathsf{sig}}^{\mathsf{PS}}, \mathsf{pk}_{\mathsf{san}}^{\mathsf{PS}})$, sign $(m_1, \sigma^{\mathsf{RS}}, (\tau_i, \sigma_i^{\mathsf{SS}})_{1\le i\le \ell_m}, \tau, \mathsf{pk}_{\mathsf{sig}}^{\mathsf{PS}}, \mathsf{pk}_{\mathsf{san}}^{\mathsf{PS}})$ and adjust. Note, the distribution of σ^{RS} and the tags are still exactly the same, even if reused, as the redactions are still performed as in the case $b = 0$.

Transition - Game 1 → Game 2: Assume that the adversary can distinguish these two games. We can then construct a reduction $\mathcal{B}$ which uses the adversary $\mathcal{A}$ to break the privacy of the underlying SS. Namely, $\mathcal{B}$ proceeds as follows. It receives $\mathsf{pk}_{\mathsf{sig}}^{\mathsf{SS}\prime}$ and $\mathsf{pk}_{\mathsf{san}}^{\mathsf{SS}\,\prime}$, and embeds them into $\mathsf{pk}_{\mathsf{sig}}^{\mathsf{PS}}$ and $\mathsf{pk}_{\mathsf{san}}^{\mathsf{PS}}$. Then, every signing, editing and proof oracle queries are answered by $\mathcal{B}$'s own oracles. However, for the calls to the LoREdit oracle, the calls for the SSs are redirected to the LoRSan oracle and the result embedded to the answer. Clearly, the simulation is perfect. Then, whatever $\mathcal{A}$ outputs is also output by $\mathcal{B}$. $|\Pr[S_1]-\Pr[S_2]| \le \nu_{\mathsf{sss\text{-}priv}}$ follows.

Game 3: Instead of signing $(\tau_1, \tau_2, \ldots, \tau_\ell, \tau, \mathsf{pk}_{\mathsf{sig}}^{\mathsf{PS}}, \mathsf{pk}_{\mathsf{san}}^{\mathsf{PS}})$ in the RSs from the first message, use the second message and then redact as required. Note, the distribution of the tags are still exactly the same due to the uniform distribution.

Transition - Game 2 → Game 3: Assume that the adversary can distinguish these two games. We can then construct a reduction $\mathcal{B}$ which uses the adversary $\mathcal{A}$ to break the privacy of the underlying RS. Namely, $\mathcal{B}$ proceeds as follows. It receives $\mathsf{pk}_{\mathsf{sig}}^{\mathsf{RS}\prime}$ and embeds them into $\mathsf{pk}_{\mathsf{sig}}^{\mathsf{PS}}$. Then, every signing query is answered by $\mathcal{B}$'s own signing oracle. However, for the calls to the LoREdit oracle, the calls for the RSs are redirected to the LoRRedact oracle and the result embedded to the answer. Clearly, the simulation is perfect. Then, whatever $\mathcal{A}$ outputs is also output by $\mathcal{B}$. Note, here we no longer need $\mathsf{RED}^{\mathsf{RS}}$, as this done via the oracles. $|\Pr[S_1] - \Pr[S_2]| \le \nu_{\mathsf{rss\text{-}priv}}$ follows.

Now, we are in the case $b = 1$. As each hop only changes the view of the adversary negligibly, privacy is proven.

Transparency To prove that our scheme is transparent, we use a sequence of games:

Game 0: The original transparency game, where $b = 0$.

Game 1: Instead of signing $(m_i, \tau, \tau_i, \mathsf{pk}_{\mathsf{sig}}^{\mathsf{PS}}, \mathsf{pk}_{\mathsf{san}}^{\mathsf{PS}})$ in the inner SS′s and adjusting them to $(m', \tau, \tau_i, \mathsf{pk}_{\mathsf{sig}}^{\mathsf{PS}}, \mathsf{pk}_{\mathsf{san}}^{\mathsf{PS}})$, directly sign $(m_i', \tau, \tau_i, \mathsf{pk}_{\mathsf{sig}}^{\mathsf{PS}}, \mathsf{pk}_{\mathsf{san}}^{\mathsf{PS}})$. Again, the distribution of σ^{RS} and the tags are still exactly the same, even if reused, as the redactions are still performed as in the case $b = 0$. Note, the restrictions on the proof-oracle are still implicitly enforced.

Transition - Game 1 → Game 2: Assume that the adversary can distinguish these two games. We can then construct a reduction $\mathcal{B}$ which uses the adversary $\mathcal{A}$ to break the transparency of the underlying SS. Namely, $\mathcal{B}$ proceeds as follows. It receives $\mathsf{pk}_{\mathsf{sig}}^{\mathsf{SS}\prime}$ and $\mathsf{pk}_{\mathsf{san}}^{\mathsf{SS}\,\prime}$, and embeds them into $\mathsf{pk}_{\mathsf{sig}}^{\mathsf{PS}}$ and $\mathsf{pk}_{\mathsf{san}}^{\mathsf{PS}}$. Then, every signing, editing and proof oracle queries are answered by $\mathcal{B}$'s own oracles. However, for the calls to the Sign/Edit oracle, the calls for the SSs are redirected to the Sign/Sanit oracle and the result embedded to the answer. Clearly, the simulation is perfect. Then, whatever $\mathcal{A}$ outputs is also output by $\mathcal{B}$. $|\Pr[S_1]-\Pr[S_2]| \le \nu_{\mathsf{sss'\text{-}tran}}$ follows.

Game 2: Instead of signing $(m_i, \tau, \tau_i, \mathsf{pk}_{\mathsf{sig}}^{\mathsf{PS}}, \mathsf{pk}_{\mathsf{san}}^{\mathsf{PS}})$ in the inner SS′s and adjusting them to $(m', \tau, \tau_i, \mathsf{pk}_{\mathsf{sig}}^{\mathsf{PS}}, \mathsf{pk}_{\mathsf{san}}^{\mathsf{PS}})$, directly sign $(m_i', \tau, \tau_i, \mathsf{pk}_{\mathsf{sig}}^{\mathsf{PS}}, \mathsf{pk}_{\mathsf{san}}^{\mathsf{PS}})$. Again, the distribution of σ^{RS} and the tags are still exactly the same, even if reused, as the redactions are still performed as in the case $b = 0$. Note, the restrictions on the proof-oracle are still implicitly enforced.

Transition - Game 1 → Game 2: Assume that the adversary can distinguish these two games. We can then construct a reduction $\mathcal{B}$ which uses the adversary $\mathcal{A}$ to break the transparency of the underlying SS. Namely, $\mathcal{B}$ proceeds as follows. It receives $\mathsf{pk}_{\mathsf{sig}}^{\mathsf{SS}\prime}$ and $\mathsf{pk}_{\mathsf{san}}^{\mathsf{SS}\,\prime}$, and embeds them into $\mathsf{pk}_{\mathsf{sig}}^{\mathsf{PS}}$ and $\mathsf{pk}_{\mathsf{san}}^{\mathsf{PS}}$. Then, every signing, editing and proof oracle queries are answered by $\mathcal{B}$'s own oracles. However, for the calls to the Sign/Edit oracle, the calls for the SSs are redirected to the Sign/Sanit oracle and the result embedded to the answer. Clearly, the simulation is perfect. Then, whatever $\mathcal{A}$ outputs is also output by $\mathcal{B}$. $|\Pr[S_1]-\Pr[S_2]| \le \nu_{\mathsf{sss'\text{-}tran}}$ follows.

Game 3: Instead of signing $(\tau_1, \tau_2, \ldots, \tau_\ell, \tau, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}})$ in the RSs from the first message and then redacting it, directly sign the redacted messages. Note, the distribution of the tags are still exactly the same due to the uniform distribution.

Transition - Game 2 → Game 3: Assume that the adversary can distinguish these two games. We can then construct a reduction $\mathcal{B}$ which uses the adversary $\mathcal{A}$ to break the transparency of the underlying RS. Namely, $\mathcal{B}$ proceeds as follows. It receives $\mathsf{pk}^{\mathsf{RS}\prime}_{\mathsf{sig}}$ and embeds them into $\mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}$. Then, every signing query is answered by $\mathcal{B}$'s own signing oracle. However, for the calls to the Sign/Edit oracle, the calls for the RSs are redirected to the Sign/Redact oracle and the result embedded to the answer. Clearly, the simulation is perfect. Then, whatever $\mathcal{A}$ outputs is also output by $\mathcal{B}$. Note, here we no longer need $\mathsf{RED}^{\mathsf{RS}}$, as this done via the oracles and are already replaced with a 0. $|\Pr[S_2] - \Pr[S_3]| \leq \nu_{\mathsf{rss\text{-}tran}}$ follows.

Now, we are in the case $b = 1$. As each hop only changes the view of the adversary negligibly, transparency is proven.

Signer-Accountability To prove that our scheme is signer-accountable, we use a sequence of games:

Game 0: The original signer-accountability game.

Game 1: We now abort, if the adversary outputs $(\mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \pi^{\mathsf{PS}}, m, \sigma^{\mathsf{PS}})$, where $\sigma^{\mathsf{PS}} = ((\sigma_i^{\mathsf{SS}})_{0\leq i\leq \ell_m}, \sigma^{\mathsf{RS}}, \tau, (\tau_i)_{1\leq i\leq \ell_m})$, where any $(m, \sigma^{\mathsf{RS}}, (\tau_i, \sigma_i^{\mathsf{SS}})_{1\leq i\leq \ell_m}, \tau, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}})$, protected by σ_0^{SS}, has never been returned by the challenger.

Transition - Game 0 → Game 1: In this case, $(\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, \pi^{\mathsf{PS}}, (m, \sigma^{\mathsf{RS}}, (\tau_i, \sigma_i^{\mathsf{SS}})_{1\leq i\leq \ell_m}, \tau, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}), \sigma_0^{\mathsf{SS}})$ is a valid forgery of the outer SS. For the reduction, $\mathcal{B}$ receives the challenge keys $\mathsf{pk}^{\mathsf{SS}\prime}_{\mathsf{san}}$ from its own challenger, and embeds them into $\mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}$. Every underlying sanitization request for the SSs is performed by the reduction's oracles. As, by assumption, the proof is wrong for σ_0^{SS}, it breaks the signer-accountability of the underlying SS in any case. Thus, $|\Pr[S_0] - \Pr[S_1]| \leq \nu_{\mathsf{sss\text{-}sigacc}}(\lambda)$ follows.

Game 2: $(\mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \pi^{\mathsf{PS}}, m, \sigma^{\mathsf{PS}})$, where $\sigma^{\mathsf{PS}} = ((\sigma_i^{\mathsf{SS}})_{0\leq i\leq \ell_m}, \sigma^{\mathsf{RS}}, \tau, (\tau_i)_{1\leq i\leq \ell_m})$, where $(m, \sigma^{\mathsf{RS}}, (\tau_i, \sigma_i^{\mathsf{SS}})_{1\leq i\leq \ell_m}, \tau, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}})$ is not new, but σ_0^{SS} has never been returned by the challenger.

Transition - Game 1 → Game 2: In this case, $(\mathsf{pk}^{\mathsf{SS}}_{\mathsf{sig}}, \pi^{\mathsf{PS}}, (m, \sigma^{\mathsf{RS}}, c, (\tau_i, \sigma_i^{\mathsf{SS}})_{1\leq i\leq \ell_m}, \tau, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}), \sigma_0^{\mathsf{SS}})$ is a valid forgery of the outer SS. The reduction works as in the prior hop. As, by assumption, the proof is wrong for σ_0^{SS}, it breaks the signer-accountability of the underlying SS in any case. Thus, $|\Pr[S_1] - \Pr[S_2]| \leq \nu_{\mathsf{sss\text{-}sigacc}}(\lambda)$ follows.

Now, the adversary can no longer win the signer-accountability game. Moreover, each hop changes the view of the adversary only negligibly, concluding the proof.

Sanitizer-Accountability To prove that our scheme is sanitizer-accountable, we use a sequence of games:

Game 0: The original sanitizer-accountability game.

Game 1: We now abort, if the adversary outputs $(m, \sigma^{\mathsf{PS}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}})$, where $\sigma^{\mathsf{PS}} = ((\sigma_i^{\mathsf{SS}})_{0\leq i\leq \ell_m}, \sigma^{\mathsf{RS}}, \tau, (\tau_i)_{1\leq i\leq \ell_m})$, where any $(m, \sigma^{\mathsf{RS}}, , (\tau_i, \sigma_i^{\mathsf{SS}})_{1\leq i\leq \ell_m}, \tau, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}})$, protected by σ_0^{SS}, has never been returned by the challenger.

Transition - Game 0 → Game 1: Here, $((m, \sigma^{\mathsf{RS}}, (\tau_i, \sigma_i^{\mathsf{SS}})_{1\leq i\leq \ell_m}, \tau, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}}), \sigma_0^{\mathsf{SS}}, \mathsf{pk}^{\mathsf{SS}}_{\mathsf{san}})$ is a valid forgery of the outer SS. For the reduction, $\mathcal{B}$ receives the challenge keys $\mathsf{pk}^{\mathsf{SS}\prime}_{\mathsf{sig}}$ from its own challenger, and embeds them into $\mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}$. Every underlying signing and proof-generation request for the SSs is performed by the reduction's oracles. As, by assumption, the signer outputs a wrong proof for σ_0^{SS}, it breaks the sanitizer-accountability of the underlying SS in any case. Thus, $|\Pr[S_0] - \Pr[S_1]| \leq \nu_{\mathsf{sss\text{-}sanacc}}(\lambda)$ follows.

Game 2: We now abort, if the adversary outputs $(m, \sigma^{\mathsf{PS}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}})$, where $\sigma^{\mathsf{PS}} = ((\sigma_i^{\mathsf{SS}})_{0\leq i\leq \ell_m}, \sigma^{\mathsf{RS}}, \tau, (\tau_i)_{1\leq i\leq \ell_m})$, where any $(m, \sigma^{\mathsf{RS}}, (\tau_i, \sigma_i^{\mathsf{SS}})_{1\leq i\leq \ell_m}, \tau, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{PS}}_{\mathsf{san}})$ is not new, but σ_0^{SS} has never been returned by the challenger.

Transition - Game 1 → Game 2: Here, $((m, \sigma^{\mathsf{RS}}, (\tau_i, \sigma_i^{\mathsf{SS}})_{1 \le i \le \ell_m}, \tau, \mathsf{pk}_{\mathsf{sig}}^{\mathsf{PS}}, \mathsf{pk}_{\mathsf{san}}^{\mathsf{PS}}), \sigma_0^{\mathsf{SS}}, \mathsf{pk}_{\mathsf{san}}^{\mathsf{SS}})$ is a valid forgery of the outer SS. The reduction works as in the prior hop. As, by assumption, the signer outputs a wrong proof for σ_0^{SS}, it breaks the sanitizer-accountability of the underlying SS in any case. Thus, $|\Pr[S_1] - \Pr[S_2]| \le \nu_{\mathsf{sss\text{-}sanacc}}(\lambda)$ follows.

Now, the adversary can no longer win the sanitizer-accountability game. Moreover, each hop changes the view of the adversary only negligibly, concluding the proof.

Invisibility To prove that our scheme is invisible, we use a sequence of games:

Game 0: The original invisibility game where $b = 0$.
Game 1: Instead of using $\mathbb{A}_0^{\mathsf{PS}}.1$ use $\mathbb{A}_1^{\mathsf{PS}}.1$ as $\mathbb{A}^{\mathsf{SS}}$ in the SS'.
Transition - Game 0 → Game 1: This does changes the view of the adversary only negligibly due to the invisibility of the underlying SS'. Namely, assume that an adversary $\mathcal{A}$ can distinguish these games with non-negligible probability. We can then construct an adversary $\mathcal{B}$ which breaks the invisibility guarantees of the used SS'. In particular, $\mathcal{B}$ receives $\mathsf{pk}_{\mathsf{sig}}^{\mathsf{SS}\prime}$ and $\mathsf{pk}_{\mathsf{san}}^{\mathsf{SS}\prime}$, and embeds them into $\mathsf{pk}_{\mathsf{sig}}^{\mathsf{PS}}$ and $\mathsf{pk}_{\mathsf{san}}^{\mathsf{PS}}$. For all oracle queries, $\mathcal{B}$ uses its own oracles to answer correctly, but makes block 1 in each underlying SS admissible or not using its own challenge oracle. Then, whatever $\mathcal{A}$ outputs, is also output by $\mathcal{B}$. $|\Pr[S_0] - \Pr[S_1]| \le \nu_{\mathsf{sss'\text{-}invis}}$ follows.
Game 2: Instead of using $\mathbb{A}_0^{\mathsf{PS}}.2$ use $\mathbb{A}_1^{\mathsf{PS}}.2$ as $\mathbb{A}^{\mathsf{RS}}$ in the RS.
Transition - Game 1 → Game 2: This does changes the view of the adversary only negligibly due to the invisibility of the underlying RS. Namely, assume that an adversary $\mathcal{A}$ can distinguish these games with non-negligible probability. We can then construct an adversary $\mathcal{B}$ which breaks the invisibility guarantees of the used RS. In particular, $\mathcal{B}$ receives $\mathsf{pk}_{\mathsf{sig}}^{\mathsf{RS}\prime}$ and $\mathsf{pk}_{\mathsf{san}}^{\mathsf{RS}\prime}$, and embeds them into $\mathsf{pk}_{\mathsf{sig}}^{\mathsf{PS}}$ and $\mathsf{pk}_{\mathsf{san}}^{\mathsf{PS}}$. For all oracle queries, $\mathcal{B}$ uses its own oracles to answer correctly using its own challenge oracle (the last three blocks are never redactable). Then, whatever $\mathcal{A}$ outputs, is also output by $\mathcal{B}$. $|\Pr[S_1] - \Pr[S_2]| \le \nu_{\mathsf{rss\text{-}invis}}$ follows.

Now, we are in the case $b = 1$. As each hop only changes the view of the adversary negligibly, invisibility is proven.