# Supplementary Material – PGUS: Pretty Good User Security for Thick MVNOs *with* a Novel Sanitizable Blind Signature

## Organization

Section 1 provides the full version of the notations and abbreviations used in this work. Section 2 introduces all the cryptographic primitives used in our SBS and PGUS construction, including Bilinear Group, Structure-Preserving Signatures on Equivalence Classes, etc. Section 3 introduces some oracles used in the definitions of security properties and formal proofs. Section 4 gives the experiment-based formal definitions of SBS's all security properties, such as immutability, EUF-CMA, etc. Section 5 proves all the security properties of SBS, the proofs of transparency, invisibility, unlinkability, responsibility, excupability, and immutability are in 5.1, while the proof of EUF-CMA is in 5.2. Finally, section 6 gives the formal security analysis of PGUS-AKA in a universally composable model.

## 1. Full Version of Notations and abbreviations

In this section, we provide Table 1, a full version of the notations used in our proposed SBS and PGUS protocol. We also provide Table 2, a full version of the abbreviations and definitions related to 5G and MVNO mentioned in our paper.

| Notation | Description |
|---|---|
| $\mathsf{pk_{sig}}, \mathsf{sk_{sig}}$ | keys for SBS Signer. |
| $\mathsf{spksig}, \mathsf{ssksig}$ | keys generated by SPSEQ for Signer. |
| $\mathsf{tpksig}, \mathsf{tsksig}$ | keys generated by TRS for Signer. |
| $\mathsf{pk_{san}}, \mathsf{sk_{san}}$ | keys for SBS Sanitizer. |
| $\mathsf{spksan}, \mathsf{ssksan}$ | keys generated by SPSEQ for Sanitizer. |
| $\mathsf{tpksan}, \mathsf{tsksan}$ | keys generated by TRS for Sanitizer. |
| $pk_{sig}^{CN}, sk_{sig}^{CN}$ | keys when CN runs SBS. |
| $pk_{san}^{gNB}, sk_{san}^{gNB}$ | keys when gNB runs SBS. |
| $m$ | plaintext message in SBS. |
| $T$ | tag or ring in TRS |
| $\ell$ | number of the blocks in message $m$ |
| $\mathsf{ADM}$ | index of the admissible modifications |
| $\mathsf{MOD}$ | modification |
| $x_i, y_i$ | integers randomly selected by Signer |
| $r, s$ | scalars randomly selected by Sanitizer |
| $\mathsf{dt}$ | data sent by User to Signer |
| $\mathsf{st}$ | a state stored by User |
| $c$ | ciphertext in PKE |
| $x \leftarrow S$ | uniformly and randomly choose |
| $\mu, \eta$ | signature generated by SPSEQ |
| $\lambda$ | security parameter |
| $\tau$ | timestamp |
| $\mathrm{n}(\cdot)$ | a function is negligible |
| $\approx$ | computationally indistinguishable |

TABLE 1. NOTATION USED IN OUR PROPOSED SBS SCHEME AND PGUS PROTOCOL

| Abbreviations | Definition |
|---|---|
| UE | User Equipment |
| gNB | Next Generation NodeB |
| CN | Core Network |
| PID | User pseudo identity |
| SUCI | Subscription Concealed Identifier |
| SUPI | Subscription Permanent Identifier |
| AKA | Authentication and Key Agreement |
| UPF | User Plane Function |
| AMF | Access Management Function |
| ACK | Acknowledgement |
| UPF | User Plane Function |
| SMF | Session Management Function |
| UDM | Unified Data Management |
| PCF | Policy Control Function |
| AUSF | Authentication Server Function |

TABLE 2. ABBREVIATIONS AND DEFINITIONS ABOUT 5G

## 2. Other preliminaries

### 2.1. Bilinear Group

Let $\mathbb{G}_1 = \langle P \rangle, \mathbb{G}_2 = \langle \hat{P} \rangle$, and $\mathbb{G}_T$ be groups of prime order $p$. A bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently computable non-degenerate bilinear map, where it holds for all $(A, \hat{B}, a, b) \in \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{Z}_p^2$ that $e(A^a, \hat{B}^b) = e(A, \hat{B})^{ab}$ (*bilinearity*), and $e(P, \hat{P}) = g_T \neq 1_{\mathbb{G}_T}$ (*non-degeneracy*). If $\mathbb{G}_1 = \mathbb{G}_2$, then $e$ is symmetric (Type-1) and asymmetric (Type-2 and Type-3) otherwise. Compared with Type-2 pairings, Type-3 pairings have no efficiently computable isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$, and are the optimal choice considering the efficiency and security trade-off [9]. If $\mathbb{G}$ is an (additive) group, then we write $\mathbb{G}^*$ to denote $\mathbb{G} \backslash \{0_{\mathbb{G}}\}$. We use A bilinear group generator algorithm $\mathsf{BGGen}$ that takes a security parameter $1^\lambda$ and output a bilinear group: $\mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P, \hat{P}) \leftarrow \mathsf{BGGen}(1^\lambda)$, where $\log_2 p = \lambda$.

### 2.2. Public Key Encryption (Full version)

Here, we recall the definitions of a public key encryption (PKE) scheme.

**Definition 1 (PKE)**. A public key encryption scheme $\mathsf{PKE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ consists of three PPT algorithms:

- $\mathsf{KGen}(1^\lambda)$ is a probabilistic algorithm which on input

the security parameter $1^\lambda$ and it generates a key pair $(pk, sk)$.
- $\mathsf{Enc}(pk, m)$: is a probabilistic algorithm which on input a encryption key pk and a message $m \in \{0,1\}^*$, and outputs a ciphertext $c$.
- $\mathsf{Dec}(sk, c)$ : is a probabilistic algorithm which on input a decryption key $sk$, a ciphertext $c$ and outputs a plaintext message $m$.

The PKE should be *correct*, which means for all $\lambda$ and $(pk, sk) \leftarrow \mathsf{KGen}(1^\lambda)$ and $m \in \{0,1\}^*$, and all $c \leftarrow \mathsf{Enc}(pk, m)$, it holds that $m = \mathsf{Dec}(sk, c) = 1$.

The instantiation of PKE is flexible according to the requirements of scenarios. In our generic construction of SBS, the security level of PKE only influences the ADM (index of admissible blocks). If ADM contains some sensitive information vulnerable to linkability attacks in the specific use cases, we suggest instantiating the PKE scheme with IND-CCA secure scheme, such as using Cramer-Shoup scheme, or using Fujisaki-Okamoto transform [13] to El-Gamal encryption scheme. Otherwise, 2048-bits RSA is sufficient.

## 2.3. Traceable Ring Signature (Full version)

The Traceable Ring Signature (TRS) scheme was introduced by Fujisaki et al. [14]. It provides a balance of anonymity and accountability, and also provides a novel "double-spending" tracing mechanism. In the main body of this work, we ignore the "issue" in the tag, which has no effect on this primitive.

**Definition 2 (TRS)** A Traceable Ring Signature scheme TRS is a tuple of 4 PPT algorithms, defined as follow.

- $\mathsf{KGen}(1^\lambda)$ is a probabilistic algorithm which on input a security parameter $\lambda$ and outputs a key pair $(sk, pk)$. *Completeness* means that a signature produced by an honest signer is always able to be accepted by an honest verifier, and *Public Traceability* captures the tracing function.is the ring size, and $i \in [N]$.
- $\mathsf{Verify}(T, \sigma, m)$ is a deterministic algorithm that takes tag $T = (issue, pk_N)$, message $m \in \{0,1\}^*$, and signature $\sigma$, then outputs a bit $b$, $b = 1$ if accepting this signature or $b = 0$ if not accepting it.
- $\mathsf{Trace}(T, (m, \sigma), (m', \sigma'))$ is a deterministic algorithm that takes tag $T = (issue, pk_N)$, and two message-signature pairs, $\{(m, \sigma), (m', \sigma')\}$, and outputs a string $\xi \in \{\texttt{"indep"}, \texttt{"linked"}\}$, or $pk$, where $pk \in [pk_N]$.

Fujisaki et al. [14] stress that tag-linkability and exculpability imply *unforgeability*, so we omit the definition of unforgeability and refer the interested readers to [14]. A TRS scheme must hold the *correctness* as follows.

- **Correctness** A TRS scheme is correct if it satisfies *Completeness* and *Public Traceability*. *Completeness* means that a signature produced by an honest signer is always ablel to be accepeted by an honest verifier, and *Public Traceability* captures the tracing funtion.

  A TRS scheme is *complete*, if for all $(pk_i, sk_i) \leftarrow \mathsf{KGen}(1^\lambda)$ for $i \in [N]$, all $T = ((pk_i)_N, issue)$ for some $issue$, all $m$ and all $\sigma \leftarrow \mathsf{Sign}(sk_i, T, m)$, it holds that $\mathsf{Verify}(T, \sigma, m) = 1$.

  A TRS scheme is *public traceable*, if for all $m, m', issue$, for $(pk_i, sk_i) \leftarrow \mathsf{KGen}(1^\lambda)$ for $i \in [N], T = (pk_N, issue)$, $\sigma \leftarrow \mathsf{Sign}(sk_i, T, m)$ and $\sigma' \leftarrow \mathsf{Sign}(sk_{i'}, T, m')$, it holds that with an overwhelming probability that:

$$\mathsf{Trace}(T, m, \sigma, m', \sigma') = \begin{cases} \texttt{"indep"}, \text{ if } i \neq i', \\ \texttt{"linked"}, \text{ else if } m = m', \\ pk_i, \text{ otherwise.} \end{cases}$$

Then, we use the security definitions of [12], which formally propose that a TRS has three security requirements: *Tag-Linkability, Anonymity and Exculpability*. The definitions of those oracles are easy and can be found in [12], [14].

Let $\mathcal{A}$ be an adversary modelled as a probabilistic algorithm. We use the definition of [4]; in the following context, the adversary is given a list of a-priority target public keys $T = (pk_1, ..., pk_N)$ and can append other public keys to the global public-key list.

- **Tag-Linkability.** Tag-linkability means that it should be infeasible for an adversary to create $(n+1)$ signatures having access to $n$ pairs of public and secret keys. So this security requirements hold true against malicious PKI. A TRS scheme is said to be tag-linkable if for all PPT adversary $\mathcal{A}(1^\lambda)$, if it has access to $n$ pairs of keys in the tag $T = (pk_1, ..., pk_n)$,

$$\Pr\left[\begin{array}{l} (\{(m_1, \sigma_1), \ldots, (m_{n+1}, \sigma_{n+1})\}) \leftarrow \mathcal{A}; \\ T \leftarrow \mathcal{A}(T); \\ \mathsf{Verify}(T, m_i, h_i) = 1, \forall i \in [n+1]; \\ \mathsf{Trace}\,(T, m_i, h_i, m_j, \sigma_j) = \texttt{"indep"}, \\ \forall i, j \in [n+1], s.t., i \neq j; \end{array}\right] \leq \mathrm{n}(\lambda)$$

- **Exculpability.** The requirement of Exculpability is that $\mathsf{Trace}(T, (m, \sigma), (m', \sigma')) = pk$ happens only if both message-signature pairs are corresponding to the same signer with respect to $pk$. The formal definition is given as follows.

$$\Pr\left[\begin{array}{l} (pk^\dagger, sk^\dagger) \leftarrow \mathsf{KGen}(1^\lambda); \\ (T, m, \sigma), (T, m', \sigma') \leftarrow \mathcal{A}^{TRS.\mathcal{O}^{Sign}}(pk^\dagger); \\ \mathsf{Trace}(T, (m, \sigma), (m', \sigma')) = pk^\dagger, \end{array}\right] \leq \mathrm{n}(\lambda)$$

- **Anonymity.** As long as a signer does not sign on two different messages with respect to the same tag, the identity of the signer is indistinguishable from any of the possible ring members. The formal definition is given as follows.

$$\Pr\left[\begin{array}{l} (pk_0^\dagger, sk_0^\dagger), (pk_1^\dagger, sk_1^\dagger) \leftarrow \mathsf{KGen}(1^\lambda); \\ b \leftarrow \{0,1\}; \\ \mathbb{O} := \{TRS.\mathcal{O}^{Sign}, TRS.\mathcal{O}^{bLoRSign}\} \\ b' \leftarrow \mathcal{A}^{\mathbb{O}}(pk_0^\dagger, pk_1^\dagger); \end{array}\right] - \frac{1}{2} \leq \mathrm{n}(\lambda)$$

## 2.4. Structure-Preserving Signatures on Equivalence Classes (Full version)

For a relation $\mathcal{R}$, let $[M]_{\mathcal{R}} = \{N|\mathcal{R}(M,N)\}$. If $\mathcal{R}$ is an equivalence relation, then $[M]_{\mathcal{R}}$ denotes the equivalence class of which $M$ is a representative. We refer readers to [15], which introduces the definition of equivalence relation $\mathcal{R}$ and class-hiding groups $[M]_{\mathcal{R}}$. In simple terms, it should be hard to distinguish elements from the same equivalence class from randomly sampled group elements. We note that equivalence relation $\mathcal{R}$ is class-hiding if and only if the Decisional Diffie–Hellman assumption (DDH) assumption holds in $\mathbb{G}_i$ [15].

We recall that in an SPSEQ scheme, one can sign vectors of group elements and it's possible to jointly randomize messages and signatures. The messages space can be a projective equivalence class on a bilinear group, i.e. $\vec{M} \in \mathbb{G}_i^{\ell}$. ($\ell > 1$) To be specific, it allows us to sign representatives of the equivalence classes $[\vec{M}]_{\mathcal{R}}$, such that a representative and its corresponding signature can be adapted to give a fresh signature of a random representative in the same class $[\vec{M}^{\rho}]_{\mathcal{R}}$. In the main body of our work, we use the instantiation of [16]. Here, we discuss the abstract model and the security model of such a signature scheme, as introduced in [15].

**Definition 3 (SPSEQ)** A Structure-preserving signatures on equivalence classes scheme SPSEQ for equivalence relation $\mathcal{R}$ over $\mathbb{G}_i$ consists of the following PPT algorithms:

- $\mathsf{KGen}(\mathsf{BG}, 1^{\ell})$ is a probabilistic algorithm which on input a bilinear group BG and a vector length $\ell > 1$ in unary outputs a key pair $(sk, pk)$.
- $\mathsf{Sign}(\vec{M}, \mathsf{sk})$ is a probabilistic algorithm which on input a representative $\vec{M} \in (\mathbb{G}_i^*)^{\ell}$ of an equivalence class $[\vec{M}]_{\mathcal{R}}$ and a secret key $sk$ outputs a signature $\sigma$.
- $\mathsf{ChgRep}(\vec{M}, \sigma, \rho, \mathsf{pk})$ is a probabilistic algorithm which on input a representative $\vec{M} \in (\mathbb{G}_i^*)^{\ell}$ of an equivalence class $[\vec{M}]_{\mathcal{R}}$, a signature $\sigma$ for $\vec{M}$, a scalar $\rho$ and a public key $pk$ returns an updated signature $\sigma'$ that is valid for the representative $\vec{M}' = \rho \cdot \vec{M}$. It could also be expressed as $[\vec{M}^{\rho}]_{\mathcal{R}}$=$[\vec{M}]_{\mathcal{R}}$.
- $\mathsf{Verify}(\vec{M}, \sigma, \mathsf{pk})$ is a deterministic algorithm which given a representative $\vec{M} \in (\mathbb{G}_i^*)^{\ell}$, a signature $\sigma$ and a public key $pk$, it returns 1 if $\sigma$ is valid for equivalence class $[\vec{M}]_{\mathcal{R}}$ under $pk$ and 0 otherwise.

The SPSEQ scheme satisfies *correctness*, *EUF-CMA security* and *Perfect Signature Adaption* as defined below w.r.t semi-honest adversaries in the random oracle model. Here, we introduce the properties including *correctness*, *EUF-CMA security*, and *Perfect Signature Adaption*.

- **Correctness.** An SPSEQ scheme over $\mathbb{G}_i$ is said to be correct if for all $\ell > 1$, all bilinear groups BG, all key pairs $(sk, pk) \in [\mathsf{KGen}(\mathsf{BG}, 1^{\ell})]$ and all messages $\vec{M} \in (\mathbb{G}_i^*)^{\ell}$ and scalars $\mu \in \mathbb{Z}_p^*$ we have:
$$\Pr\left[\mathsf{Verify}(\vec{M}, \mathsf{Sign}(\vec{M}, sk), pk) = 1\right] = 1$$
and $1 =$
$$\Pr\left[\mathsf{Verify}(\mu \cdot \vec{M}, \mathsf{ChgRep}(\vec{M}, \mathsf{Sign}(\vec{M}, sk), \mu, pk), pk) = 1\right]$$
- **EUF-CMA.** An SPSEQ signature scheme is said to be existentially unforgeable under chosen message attacks (EUF-CMA) if for all $\ell > 1$, for all $n \in \mathrm{poly}(\lambda)$, and for all PPT adversaries $\mathcal{A}$ who have access to the signing oracle,
$$\Pr\left[\begin{array}{c} (pk, sk) \leftarrow \mathsf{KGen}(\mathsf{BG}, 1^{\ell}); \\ (\vec{M}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}^{Sign}}(pk) \\ 1 = \mathsf{Verify}(pk, \vec{M}^*, \sigma^*) \\ \forall \vec{M} \in \mathcal{Q} : [\vec{M}^*]_{\mathcal{R}} \neq [\vec{M}]_{\mathcal{R}} \end{array}\right] \leq n(\lambda).$$
- **Perfect Signature Adaptation.** An EQS scheme is said to perfectly adapt signatures if for all valid tuples $(\mathsf{sk}, \mathsf{pk}, \vec{M}, \sigma, \rho)$, such as $\mathsf{Verify}(\mathsf{pk}, \vec{M}, \sigma) = 1$, $\vec{M} \in \mathbb{G}_1^{\ell}$ for some $\ell > 1$, and $\rho \leftarrow \mathbb{Z}_q^*$ it holds that
$$\mathsf{ChgRep}(\mathsf{pk}, \vec{M}, \sigma, \rho) \approx \mathsf{Sign}(\mathsf{sk}, \vec{M}^{\rho})$$
This means two representations are identically distributed.

## 2.5. Commitment (Full version)

Com allows one party to commit to a chosen value while keeping it hidden, with the ability to reveal the committed value later, we use the instantiation of [1].

**Definition 4 (Com)** A commitment scheme Com is a tuple of 3 PPT algorithms, defined as follows.

- $\mathsf{ck} \leftarrow \mathsf{KGen}(1^{\lambda})$: Given a security parameter $\lambda$, generates a public parameter ck that is implicitly passed as input to the other algorithms.
- $(c, \delta) \leftarrow \mathsf{Com}_{ck}(m, r)$: Given the public parameter ck, a message $m$, and a randomness $r$, outputs a commitment $c$ together with an opening information $\delta$.
- $m/\perp \leftarrow \mathsf{Decom}_{ck}(c, m, \delta)$: Given the public parameter ck, a commitment $c$, the message $m$, and an opening information $\delta$, outputs the message $m$ if the opening verification is successful, or $\perp$ if the verification fails.

The commitment scheme Com satisfies *hiding* and *binding* properties. Besides, some additional properties are demanded in the UC-secure commitment scheme, including extractability and equivocability. We introduce the interested readers to [1] for details.

- **Hiding.** The commitment conceals the committed value, providing no information about it to the receiver until the sender chooses to reveal it. For all PPT adversaries $\mathcal{A}$ who are given the commitment $c$, then $\mathcal{A}$ cannot distinguish any different messages $m_0$ and $m_1$ in polynomial time.
$$\begin{array}{c} |\Pr[\mathcal{A}(\mathsf{Com}_{ck}(m_0, r)) = 1] \\ - \Pr[\mathcal{A}(\mathsf{Com}_{ck}(m_1, r)) = 1]| \leq n(\lambda) \end{array}$$
- **Binding.** Once the sender has committed to a value, they cannot change it; the commitment is bound to that specific value, preventing the sender from revealing a

different value later. For all PPT adversaries $\mathcal{A}$, it is difficult for $\mathcal{A}$ to find two different messages $m, m'$ and the corresponding open information $\delta, \delta'$, which are related to the same commitment $c$.

$$\Pr\left[\begin{array}{c}(c, m, \delta, m', \delta') \leftarrow \mathcal{A}(1^\lambda, \mathsf{ck}) \\ m \neq m' \\ \mathsf{Decom}ck(c, m, \delta) = m \\ \mathsf{Decom}ck(c, m', \delta') = m'\end{array}\right] \leq n(\lambda)$$

## 2.6. Non-Interactive Zero-Knowledge Proof (Full version)

The NIZK scheme ZK [10] enables a prover to convince a verifier of a statement's validity without revealing additional information. The ZK operates under a common reference string (CRS) model and uses a relation $\mathcal{R}$, where a statement-witness pair $(x, w)$ satisfies $\mathcal{R}(x, w)$ if $x$ is valid.

**Definition 5 (ZK)** A Non-Interactive Zero-Knowledge Proof (NIZK) system ZK for a relation $\mathcal{R}$ consists of the following PPT algorithms:

- $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{K}_{\mathsf{crs}}(\lambda)$: A probabilistic algorithm that, given a security parameter $\lambda$, outputs a common reference string (CRS) crs and a trapdoor td.
- $\pi \leftarrow \mathsf{P}(\mathsf{crs}, x, w)$: A probabilistic algorithm that, given crs, a statement $x$, and a witness $w$ with $(x, w) \in \mathcal{R}$, outputs a proof $\pi$. (Prove)
- $0, 1 \leftarrow \mathsf{V}(\mathsf{crs}, x, \pi)$: A probabilistic algorithm that, given crs, a statement $x$, and a proof $\pi$, returns 1 (accept) if $\pi$ is valid, otherwise 0. (Verify)
- $\pi \leftarrow \mathsf{S}(\mathsf{crs}, \mathsf{td}, x)$: A probabilistic algorithm that, given crs, trapdoor td, and statement $x$, outputs a simulated proof $\pi$. (Simulation)

The ZK scheme also satisfies *completeness*, *zero-knowledge*, and *soundness* as defined in the literature, ensuring proof validity without revealing the witness.

- **Comleteness.** A NIZK proof system ZK is said to complete if, for all $(x, w)$ pairs such that $\mathcal{R}(x, w) = 1$, an honest prover can generate a proof that the verifier will accept with overwhelming probability. For $\forall (x, w) \in \mathcal{R}$, it holds:

$$\Pr\left[\mathsf{V}(\mathsf{crs}, x, \pi) = 1 : \pi \leftarrow \mathsf{P}(\mathsf{crs}, x, w)\right] \geq 1 - n(\lambda)$$

- **Zero-Knowledge.** A NIZK proof system ZK is zero-knowledge if, for any statement $x$ and witness $w$ such that $(x, w) \in \mathcal{R}$, there exists a simulator S that can generate a simulated proof $\pi$ that is indistinguishable from a real proof, without knowing the witness.

$$\forall (x, w) \in \mathcal{R}, \quad \pi \approx \mathsf{S}(\mathsf{crs}, \mathsf{td}, x)$$

- **Soundness.** A NIZK proof system ZK satisfies soundness if, for any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ that produces a false statement $x \notin \mathcal{L}_{\mathcal{R}}$, the probability that V accepts a forged proof $\pi$ is negligible.

$$\Pr\left[\mathsf{V}(\mathsf{crs}, x, \pi) = 1 : x \notin \mathcal{L}_{\mathcal{R}}, \pi \leftarrow \mathcal{A}(\mathsf{crs}, x)\right] \leq n(\lambda)$$

## 2.7. Sanitizable Signature (SS) (Full version)

The sanitizable signature scheme introduced in [2], [5], [7], [11] provides a mechanism for selective message modification while maintaining signature validity. SS achieves this by allowing a designated sanitiser to alter predefined parts of a signed message without invalidating the original signature, ensuring controlled flexibility. This is accomplished through a combination of traditional digital signatures and a sanitization algorithm, which enables both the message's integrity and authenticity to be preserved, even after selective modification. The scheme is particularly useful in contexts where sensitive information must be redacted or updated post-signing while still guaranteeing security properties like unforgeability and accountability.

**Definition 6** (Sanitizable Signature). *An Sanitizable Signature (*SS*) scheme is defined as a tuple of six algorithms :*{KGen, Sign, Sanit, Verify, Prove, Judge}. *Specifically, a Sanitizable Signature scheme comprises the following probabilistic polynomial-time (PPT) algorithms:*

- $\underline{(\mathsf{pk}_{\mathsf{sig}}, \mathsf{sk}_{\mathsf{sig}}), (\mathsf{pk}_{\mathsf{san}}, \mathsf{sk}_{\mathsf{san}}), \leftarrow \mathsf{KGen}(1^\lambda, 1^\ell)}$: The key generation algorithm inputs the security parameter $\lambda$ and the upper limit of the message's blocks $ell$, and outputs the public and private key pairs of the signing and sanitizing steps respectively.
- $\underline{\sigma \leftarrow \mathsf{Sign}(m, \mathsf{ADM}, \mathsf{sk}_{\mathsf{sig}}, \mathsf{pk}_{\mathsf{san}})}$ : The signing algorithm inputs the message, the index of admissible modification ADM, the signing private key $\mathsf{sk}_{\mathsf{sig}}$ as well as the sanitizing public key $\mathsf{pk}_{\mathsf{san}}$, and outputs a signature $\sigma$.
- $\underline{(m', \sigma') \leftarrow \mathsf{Sanit}(m, \mathsf{MOD}, \sigma, \mathsf{pk}_{\mathsf{sig}}, \mathsf{sk}_{\mathsf{san}})}$ : The Sanitization algorithm inputs a message $m \in \{0, 1\}^*$, a modification instruction $\mathsf{MOD} \subset \mathbb{N} \times \{0, 1\}^\ell$ of the modifications to $m$, a signature $\sigma$, a public signing key $\mathsf{pk}_{\mathsf{sig}}$ and a private sanitizing key $\mathsf{sk}_{\mathsf{san}}$, then outputs a sanitized message $m'$ and a corresponding sanitized signature $\sigma'$.
- $\underline{b \leftarrow \mathsf{Verify}(\mathsf{pk}_{\mathsf{sig}}, \mathsf{pk}_{\mathsf{san}}, m, \sigma)}$ : The verification algorithm inputs a message $m$, a signature $\sigma$, a signing public key $\mathsf{pk}_{\mathsf{sig}}$, a sanitizing public key $\mathsf{pk}_{\mathsf{san}}$, then outputs a bit $b \in \{0, 1\}$, outputs 1 if validity holds, and 0 otherwise.
- $\underline{\pi \leftarrow \mathsf{Prove}(m, \sigma, \mathsf{sk}_{\mathsf{sig}}, \mathsf{pk}_{\mathsf{san}})}$ : The proof algorithm takes as input a message $m$, a signature $\sigma$, a signer private key $\mathsf{sk}_{\mathsf{sig}}$, and a sanitizing public key $\mathsf{sk}_{\mathsf{sig}}$, then outputs a proof $\pi$.
- $\underline{d \leftarrow \mathsf{Judge}(m, \sigma, \mathsf{pk}_{\mathsf{sig}}, \mathsf{pk}_{\mathsf{san}})}$ : The judge algorithm takes as input a message $m$, a signature $\sigma$, signing and sanitizing public keys $\mathsf{pk}_{\mathsf{sig}}, \mathsf{pk}_{\mathsf{san}}$, and proof $\pi$, then outputs a decision $d \in \{Sig, San\}$ indicating whether the message-signature pair was created by the signer or the sanitizer.

The initial construction of sanitizable signatures was based on Chameleon Hash functions, as proposed by Brzuska et al. [5]. This work identified immutability, transparency, and signer/sanitizer accountability as fundamental security properties of sanitizable signatures. Their construction is highly encapsulated, with accountability being handled internally through chameleon hashes, which cannot resist linkability attacks [6]. Fleischhacker et al. [11] proposed the linkability issues related to Chameleon Hashes, and turn to use re-randomized keys to achieve unlinkability. In their scheme, accountability is realized independently via the Prove and Judge algorithms, which internally invoke two Non-Interactive Zero-Knowledge Proof Systems. However, [11] relies on a signature scheme with perfectly re-randomizable keys, as well as a deterministic signature scheme, both of which require the plaintext message when signing. Bultel et al. [7] extended the existing works and added invisibility. Their construction in is based on the Structure-Preserving Signatures on Equivalence Classes (SPSEQ) and Verifiable Ring Signature (VRS), and accountability is independently realized by VRS.

## 3. Oracles

Here we introduce the oracles related to our proposed SBS.

Firstly, we introduce the signing oracles $\mathcal{O}^{Sign}$. Our Sign Oracle encapsulates the Extract, Sign and Derive algorithms, and its output is the signature generated by the interactive protocol between User and Signer.

$\underline{\mathcal{O}^{Sign}(\mathsf{ADM}, m, \mathsf{pk_{san}})}$
$(\mathsf{dt}, \mathsf{st}) \leftarrow_{User} \mathsf{Extract}(\mathsf{ADM}, m, \mathsf{pk_{san}})$
$\sigma_{inner} \leftarrow_{Signer} \mathsf{Sign}(\mathsf{sk}^{\dagger}_{\mathsf{sig}}, \mathsf{pk_{san}}, \mathsf{dt})$
$\sigma \leftarrow_{User} \mathsf{Derive}(\mathsf{st}, \sigma_{inner})$
$L := L \| \{\mathsf{pk_{sig}}^{\dagger}, \mathsf{pk_{san}}, \mathsf{ADM}, m, \sigma\}$
return $\sigma$.

We refer to the definition of $\mathcal{O}^{bLoRADM}$ oracle from [3], [8], but we make some modifications.

$\underline{\mathcal{O}^{bLoRADM}(\mathsf{ADM}_0, \mathsf{ADM}_1, m, \mathsf{pk_{san}})}$
if $(|\mathsf{ADM}_0| = |\mathsf{ADM}_1| = |m|) \wedge (\mathsf{pk_{san}} = \mathsf{pk_{san}}^{\dagger} \vee \mathsf{ADM}_0 = \mathsf{ADM}_1)$
then $(\mathsf{dt}, \mathsf{st}) \leftarrow_{User} \mathsf{Extract}(\mathsf{ADM}_b, m, \mathsf{pk_{san}})$
$\sigma_{inner} \leftarrow_{Signer} \mathsf{Sign}(\mathsf{sk}^{\dagger}_{\mathsf{sig}}, \mathsf{pk_{san}}, \mathsf{dt})$
$\sigma \leftarrow_{User} \mathsf{Derive}(\mathsf{st}, \sigma_{inner})$
if $\mathsf{pk_{san}} = \mathsf{pk_{san}}^{\dagger}$ then
$R := \{R \| (m, \sigma, \mathsf{ADM}_0 \circ \mathsf{ADM}_1)\}$
end if, return $\sigma$.
end if, return $\perp$.

We refer to the definiton of transparency experiment in [5], which contains a "Sanit/Sign" box, inside this box is a hidden random bit $b$, which can be either 0 or 1. The adversary can input a message and its modification details to this box. Depending on the value of $b$, the box either generates a sanitized signature (if $b = 0$) or creates a new signature for the modified message from scratch (if $b = 1$). Here, we make some modifications and design $\mathcal{O}^{bSanit/Sign}$ oracle.

$\underline{\mathcal{O}^{bSanit/Sign}(m, \mathsf{ADM}, \mathsf{MOD})}$
If $\mathsf{MOD} \notin \mathsf{ADM}$,return $\perp$.
$(\mathsf{dt}, \mathsf{st}) \leftarrow_{User} \mathsf{Extract}(\mathsf{ADM}, m, \mathsf{pk_{san}}^{\dagger})$
$\sigma_{inner} \leftarrow_{Signer} \mathsf{Sign}(\mathsf{sk}^{\dagger}_{\mathsf{sig}}, \mathsf{pk_{san}}^{\dagger}, \mathsf{dt})$
$\sigma \leftarrow_{User} \mathsf{Derive}(\mathsf{st}, \sigma_{inner})$
If $b = 0$,
$(\mathsf{dt}', \mathsf{st}') \leftarrow_{User} \mathsf{Extract}(\mathsf{ADM}, \mathsf{MOD}(m), \mathsf{pk_{san}}^{\dagger})$
$\sigma'_{inner} \leftarrow_{Signer} \mathsf{Sign}(\mathsf{sk}^{\dagger}_{\mathsf{sig}}, \mathsf{pk_{san}}^{\dagger}, \mathsf{dt}')$
$\sigma' \leftarrow_{User} \mathsf{Derive}(\mathsf{st}', \sigma'_{inner})$
else if $b = 1$,
$\sigma' \leftarrow \mathsf{Sanit}(m, \mathsf{MOD}, \sigma, \mathsf{pk_{sig}}^{\dagger}, \mathsf{sk_{san}}^{\dagger})$
return $\sigma'$.

Finally, we use two kinds of sanitizing oracles. The common form $\mathcal{O}^{San}(m, \mathsf{MOD}, \sigma, \mathsf{pk_{sig}})$ is defined similar to signing oracles, it returns a sanitizable signature $\sigma'$ and updates $L$. But the special form $\mathcal{O}^{San'}(m, \mathsf{MOD}, \sigma, \mathsf{pk_{sig}})$ will update $R$ if ($\exists \mathsf{ADM}$ s.t. $(m, \sigma, \mathsf{ADM}) \in \mathcal{R} \wedge \mathsf{MOD} \in \mathsf{ADM}$). We refer the reader to the definition of Strong Invisibility from Beck et al. [3], which contains a full version of $\mathcal{O}^{San'}$. We also refer the reader to the definition of unlinkability experiment in [6], which contains the "left-or-right sanitize" oracle $\mathcal{O}^{bLoRSanit}(m_0, \mathsf{MOD}_0, \sigma_0, m_1, \mathsf{MOD}_1, \sigma_1)$, which on inputs two message-modification-signature tuples and outputs a sanitized signature produced from one of the tuples, then updates $L$.

## 4. Definitions of All Security Properties of SBS

**Definition 7 (Immutability)** A scheme $\Pi$ is immutable if for all PPT adversaries $\mathcal{A}$, it satisfies that $\Pr\left[ExpImm_{\mathcal{A},\Pi}(1^{\lambda}) = 1\right] \leq \mathrm{n}(\lambda)$, and the experiment is defined as follows.

$\underline{ExpImm_{\mathcal{A},\Pi}(1^{\lambda})}$
$(\mathsf{pk_{sig}}^{\dagger}, \mathsf{sk_{sig}}^{\dagger}) \leftarrow \mathsf{KGen_{sig}}(1^{\lambda})$
$(\mathsf{pk_{san}}^{*}, m^{*}, \sigma^{*}) \leftarrow \mathcal{A}^{\mathcal{O}^{Sign}}(\mathsf{pk_{sig}}^{\dagger})$
Parse $L$ as $\left\{\left(\mathsf{pk}_{\mathsf{sig},i}, \mathsf{pk}_{\mathsf{san},i}, m_i, \mathsf{ADM}_i, h_i\right)\right\}_{i=1}^{[|L|]}$
If $\mathsf{Verify}(m^{*}, \sigma^{*}, \mathsf{pk_{sig}}^{*}, \mathsf{pk_{san}}^{\dagger})$,
and for all $i \in [|L|]$
$\mathsf{pk_{san}} \neq \mathsf{pk}_{\mathsf{san},i}$ ,or
$m^{*} \neq \mathsf{MOD}_i(m_i)$ for some $\mathsf{MOD}_i \notin \mathsf{ADM}_i$
return 1;
else, return 0.

**Definition 8 (Transparency)** A scheme $\Pi$ is transparent if for all PPT adversaries $\mathcal{A}$, it cannot predict the random bit $b$ significantly better than by guessing, which implies that$\Pr\left[ExpTran^{0}_{\mathcal{A},\Pi}(1^{\lambda}) = 1\right]$-$\Pr\left[ExpTran^{1}_{\mathcal{A},\Pi}(1^{\lambda}) = 1\right] \leq \mathrm{n}(\lambda)$. And the experiment is defined as follows.

$\underline{ExpTran^{b}_{\mathcal{A},\Pi}(1^{\lambda})}$
$(\mathsf{pk_{sig}}^{\dagger}, \mathsf{sk_{sig}}^{\dagger}) \leftarrow \mathsf{KGen_{sig}}(1^{\lambda})$
$(\mathsf{pk_{san}}^{\dagger}, \mathsf{sk_{san}}^{\dagger}) \leftarrow \mathsf{KGen_{san}}(1^{\lambda})$
$b' \leftarrow \mathcal{A}^{\mathcal{O}^{Sign}, \mathcal{O}^{San}, \mathcal{O}^{bSanit/Sign}}(\mathsf{pk_{sig}}^{\dagger}, \mathsf{pk_{san}}^{\dagger})$
return $b'$.

**Definition 9 (Unlinkability)** A scheme $\Pi$ is unlinkable if for all PPT adversaries $\mathcal{A}$, it cannot predict the random bit $b$ significantly better than by guessing, which implies that $\Pr\left[ExpUnl^0_{\mathcal{A},\Pi}(1^\lambda)=1\right]$-$\Pr\left[ExpUnl^1_{\mathcal{A},\Pi}(1^\lambda)=1\right] \leq \mathrm{n}(\lambda)$. And the experiment is defined as follows.

$\underline{ExpUnl^b_{\mathcal{A},\Pi}(1^\lambda)}$
$(\mathsf{pk_{sig}}^\dagger, \mathsf{sk_{sig}}^\dagger) \leftarrow \mathsf{KGen_{sig}}(1^\lambda)$
$(\mathsf{pk_{san}}^\dagger, \mathsf{sk_{san}}^\dagger) \leftarrow \mathsf{KGen_{san}}(1^\lambda)$
$b' \leftarrow \mathcal{A}^{\mathcal{O}^{Sign},\mathcal{O}^{San},\mathcal{O}^{bLoRSanit}}(\mathsf{pk_{sig}}^\dagger, \mathsf{pk_{san}}^\dagger)$
return $b'$.

**Definition 10 (Invisibility)** A scheme $\Pi$ is invisible if for all PPT adversaries $\mathcal{A}$, it cannot predict the random bit $b$ significantly better than by guessing, which implies that $\Pr\left[ExpInv^0_{\mathcal{A},\Pi}(1^\lambda)=1\right]$-$\Pr\left[ExpInv^1_{\mathcal{A},\Pi}(1^\lambda)=1\right] \leq \mathrm{n}(\lambda)$. And the experiment is defined as follows.

$\underline{ExpInv^b_{\mathcal{A},\Pi}(1^\lambda)}$
$(\mathsf{pk_{sig}}^\dagger, \mathsf{sk_{sig}}^\dagger) \leftarrow \mathsf{KGen_{sig}}(1^\lambda)$
$(\mathsf{pk_{san}}^\dagger, \mathsf{sk_{san}}^\dagger) \leftarrow \mathsf{KGen_{san}}(1^\lambda)$
$b' \leftarrow \mathcal{A}^{\mathcal{O}^{San'},\mathcal{O}^{bLoRADM}}(\mathsf{pk_{sig}}^\dagger, \mathsf{pk_{san}}^\dagger)$
return $b'$;

**Definition 11 (Responsibility)** A scheme $\Pi$ is responsible if for all PPT adversaries $\mathcal{A}$ , and even for malicious Signer, it satisfies that $\Pr\left[ExpSigRespon_{\mathcal{A},\Pi}(1^\lambda)=1\right] \leq \mathrm{n}(\lambda)$, and the experiment is defined as follows.

$\underline{ExpSigRespon_{\mathcal{A},\Pi}(1^\lambda)}$
$(\mathsf{pk_{san}}^\dagger, \mathsf{sk_{san}}^\dagger) \leftarrow \mathsf{KGen_{san}}(1^\lambda)$
$(\mathsf{pk_{sig}}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}^{San}}(\mathsf{pk_{san}}^\dagger)$
Parse $L$ as $\left\{\left(\mathsf{pk}_{\mathsf{sig},i}, \mathsf{pk}_{\mathsf{san},i}, m_i, \mathsf{ADM}_i, h_i\right)\right\}_{i=1}^{[|L|]}$
$b_0 := \mathsf{Verify}(m^*, \sigma^*, \mathsf{pk_{sig}}^*, \mathsf{pk_{san}}^\dagger)$,
$b_1 := \left((\mathsf{pk_{sig}}^*, m^*, \sigma^*) \notin \left\{\left(\mathsf{pk}_{\mathsf{sig},i}, m_i, h_i\right)\right\}_{i=1}^{[|L|]}\right)$,
$b_2 := (\mathsf{Trace}(\sigma_{true}, \sigma^*, \mathsf{pk_{sig}}^*, \mathsf{pk_{san}}^\dagger) = \texttt{"indep"}$
Return $b_0 \wedge b_1 \wedge b_2$

**Definition 12 (Exculpability)** A scheme $\Pi$ is exculpable if for all PPT adversaries $\mathcal{A}$, and even for malicious Sanitizer, it satisfies that $\Pr\left[ExpSigExculp_{\mathcal{A},\Pi}(1^\lambda)=1\right] \leq \mathrm{n}(\lambda)$ and the experiment is defined as follows.

$\underline{ExpSigExculp_{\mathcal{A},\Pi}(1^\lambda)}$
$(\mathsf{pk_{sig}}^\dagger, \mathsf{sk_{sig}}^\dagger) \leftarrow \mathsf{KGen_{sig}}(1^\lambda)$
$(\mathsf{pk_{san}}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}^{Sign}}_{sanitizer}(\mathsf{pk}^\dagger_{\mathsf{sig}})$
Parse $L$ as $\left\{\left(\mathsf{pk}_{\mathsf{sig},i}, \mathsf{pk}_{\mathsf{san},i}, m_i, \mathsf{ADM}_i, h_i\right)\right\}_{i=1}^{[|L|]}$
$b_0 := \mathsf{Verify}(m^*, \sigma^*, \mathsf{pk_{sig}}^\dagger, \mathsf{pk_{san}}^*)$,
$b_1 := \left((\mathsf{pk_{san}}^*, m^*, \sigma^*) \notin \left\{\left(\mathsf{pk}_{\mathsf{san},i}, m_i, h_i\right)\right\}_{i=1}^{[|L|]}\right)$,
$b_2 := (\mathsf{Trace}(\sigma_{true}, \sigma^*, \mathsf{pk_{sig}}^\dagger, \mathsf{pk_{san}}^*) = \mathsf{pk_{sig}}^\dagger)$
Return $b_0 \wedge b_1 \wedge b_2$

**Definition 13 (EUF-CMA)** A scheme $\Pi$ is EUF-CMA secure, if for all PPT adversaries $\mathcal{A}$, it satisfies that $\Pr\left[ExpEUFCMA_{\mathcal{A},\Pi}(1^\lambda)=1\right] \leq \mathrm{n}(\lambda)$ and the experiment is defined as follows.

$\underline{ExpEUFCMA_{\mathcal{A},\Pi}(1^\lambda)}$
$(\mathsf{pk}^\dagger_{\mathsf{sig}}, \mathsf{sk}^\dagger_{\mathsf{sig}}) \leftarrow \mathsf{KGen}_{sig}(1^\lambda)$
$(\mathsf{pk}^\dagger_{\mathsf{san}}, \mathsf{sk}^\dagger_{\mathsf{san}}) \leftarrow \mathsf{KGen_{san}}(1^\lambda)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}^{Sign},\mathcal{O}^{Sanit}}(\mathsf{pk}^\dagger_{\mathsf{sig}}, \mathsf{pk}^\dagger_{\mathsf{san}})$
For $i = 1, 2, \ldots, k$, the signing oracle $\mathcal{O}^{Sign}$ takes $(\mathsf{ADM}_i, m_i, \mathsf{pk}^\dagger_{\mathsf{san},i})$ as input, and answers $\sigma_i$.
For $j = k+1, \ldots, n$, the sanitizing oracle $\mathcal{O}^{Sanit}$ takes $(m_j, \mathsf{MOD}_j, \sigma_j, \mathsf{pk}^\dagger_{\mathsf{sig},j})$ as input, and answers $(m'_j, \sigma'_j)$.
$b_0 := \mathsf{Verify}(m^*, \sigma^*, \mathsf{pk}^\dagger_{\mathsf{sig}}, \mathsf{pk}^\dagger_{\mathsf{san}})$
$b_1 := \{(pk_{san}, m^*) \neq (pk_{san,i}, m_i) \text{ for all } i = 1, 2, \ldots, k\}$
$b_2 := \{(pk_{sig}, m^*) \neq (pk_{sig,j}, m'_j) \text{ for all } j = k+1, \ldots, n\}$
Return $b_0 \wedge b_1 \wedge b_2$

# 5. Security Proof of SBS

We first demonstrate that our $\Pi_{SS}$ fully inherits four security properties of scheme $\Pi_1$ in [7]: immutability, invisibility, transparency, and unlinkability. There are only two differences between $\Pi_1$ and $\Pi_{SS}$: 1) The adversary of $\Pi_{SS}$ doesn't have access to the Prove oracle, since $\Pi_{SS}$ removes the Prove and Judge algorithms. So the adversary in $\pi_{SS}$ has fewer resources than in $\Pi_1$. 2) $\pi_{SS}$ has decoupled the $\Pi_1$.Sign algorithm into an interactive protocol between the User and Signer, consisting of the $\Pi_{SS}$.Extract, $\Pi_{SS}$.Sign, and $\Pi_{SS}$.Derive algorithms. However, when designing the security experiments and oracles corresponding to these four properties, $\Pi_{SBS}$ encapsulates its Extract, Sign, Derive algorithms into its signing oracle $\mathcal{O}^{Sign}$.

Therefore, if the probability that a PPT adversary wins in these four experiments of $\Pi_1$'s security property is negligible, then the probability that a PPT adversary wins in these four $\Pi_{SS}$'s experiments is also negligible. Thus, if $\Pi_1$ holds these security properties, $\Pi_{SS}$ holds them as well. Our conclusion of $\Pi_{SS}$ is given in Corollary 1.

## 5.1. Reduction of Properties (Proof of Theorem 1)

Next, we prove the following lemmas:

**Lemma 1** Let $\Pi_{SS}$ be immutable, then the construction $\Pi_{SBS}$ is immutable.

**Proof** Assume a PPT adversary $\mathcal{A}$ that wins the immutability game of $\Pi_{SBS}$ with non-negligible probability. Then, we show how to construct an adversary $\mathcal{B}$ that breaks the immutability of $\Pi_{SS}$.

The algorithm $\mathcal{B}$ receives $\mathsf{spksig}^\dagger$ as input, then it generates the TRS key pairs for signing part, $(\mathsf{tpksig}^\dagger, \mathsf{tsksig}^\dagger) \leftarrow \mathsf{TRS.KGen}(1^\lambda)$, and it lets $\mathsf{pk_{sig}}^\dagger := (\mathsf{spksig}^\dagger, \mathsf{tpksig}^\dagger)$. Let $\mathcal{A}$ runs that $(\mathsf{pk_{san}}^*, m^*, \sigma^*) \leftarrow A(\mathsf{pk_{sig}}^\dagger)$. Then, $\mathcal{B}$ simulates the $\Pi_{SBS}$'s Sign oracle $\Pi_{SBS}.\mathcal{O}^{Sign}$ gets $(\mathsf{ADM}, m, \mathsf{pk_{san}})$ as input, parses and submits $(0\|\mathsf{ADM}, \mathsf{pk_{san}}\|m, \mathsf{spksan})$ to $\Pi_{SS}$'s Sign oracle $\Pi_{SS}.\mathcal{O}^{Sign}$ as input, and receives a signature $\sigma_{SS}$. Then it parses $\sigma_{SS}$ as $(\pi_{SS}, ...)$ , and runs $\sigma_{TRS} \leftarrow \mathsf{TRS.Sign}(\mathsf{tsksig}, (\mathsf{tpksig}, \mathsf{tpksan}), \mathsf{pk_{sig}}^\dagger\|\mathsf{pk_{san}}\|\pi_{SS})$.
Next, it returns $\sigma := (\sigma_{SS}, \sigma_{TRS})$ to $\mathcal{A}$. Finally,

$\mathcal{B}$ receives $(\mathsf{pk_{san}}^*, m^*, \sigma^*)$, parses and return $(\mathsf{spksan}^*, \mathsf{pk_{san}}^* \| m^*, \sigma^*_{ss})$.

Because we assume that $\mathcal{A}$ wins the $\Pi_{SBS}$'s immutability game, it means that $ExpImm_{\mathcal{A},\Pi_{SBS}}(\lambda)$ returns 1, then the two conditions hold: i) $\Pi_{SBS}.\mathsf{Verify}(\mathsf{pk_{sig}}\dagger, \mathsf{pk_{san}}^*, m^*, \sigma^*) = 1$. ii) $\forall i \in [|L|], \mathsf{MOD} \in \mathsf{ADM}_i, s.t. \mathsf{pk_{san}}^* \neq \mathsf{pk_{san}}_{,i} \vee m^* \neq \mathsf{MOD}(m_i)$. We use the first condition to reduce that $\Pi_{SS}.\mathsf{Verify}(\mathsf{spksig}^\dagger, \mathsf{spksan}^*, \mathsf{pk_{san}}^*, \sigma^*_{SS}) = 1$, and we use the second condition to reduce that $\mathsf{pk_{san}}^* \| m^* \neq \mathsf{MOD}(\mathsf{pk_{san}}_{,i} \| m_i)$. Then, $ExpImm_{\mathcal{B},\Pi_{SS}}(\lambda)$ returns 1. This contradicts the immutability of $\Pi_{SS}$, so we have thus shown our construction $\Pi_{SBS}$ is immutable. □

**Lemma 2** Let $\Pi_{SS}$ be invisible, then the construction $\Pi_{SBS}$ is invisible.

**Proof** Assume a PPT adversary $\mathcal{A}$ that wins the invisibility game of $\Pi_{SBS}$ with non-negligible probability. Then, we show how to construct an adversary $\mathcal{B}$ that breaks the invisibility of $\Pi_{SS}$.

The algorithm $\mathcal{B}$ receives $\mathsf{spksig}^\dagger$ as input, then it generates two TRS key pairs, $(\mathsf{tpksig}^\dagger, \mathsf{tsksig}^\dagger), (\mathsf{tpksan}^\dagger, \mathsf{tsksan}^\dagger) \leftarrow \mathsf{TRS.KGen}(1^\lambda)$. Then it lets $\mathsf{pk_{sig}}^\dagger := (\mathsf{spksig}^\dagger, \mathsf{tpksig}^\dagger)$, $\mathsf{pk_{san}}^\dagger := (\mathsf{spksan}^\dagger, \mathsf{tpksan}^\dagger)$. Then it lets $\mathcal{A}$ to guess the bit, $b' \leftarrow \mathcal{A}(\mathsf{pk_{sig}}^\dagger, \mathsf{pk_{san}}^\dagger)$. Then, $\mathcal{B}$ simulates two $\Pi_{SBS}$'s oracles $\Pi_{SBS}.\mathcal{O}^{San'}$ and $\Pi_{SBS}.\mathcal{O}^{bLoRADM}$.

$\mathcal{B}$ simulates $\mathcal{O}^{San'}$ as follows. On input a tuple $(m_i, \mathsf{MOD}_i, h_i, \mathsf{pk_{sig}}_{,i})$, if $(\mathsf{pk_{sig}}_{,i} = \mathsf{pk_{sig}}^\dagger) \wedge \neg(\exists (m_i, h_i, \mathsf{ADM}_i) \in R \wedge \mathsf{MOD}_i \in \mathsf{ADM}_i)$, then the oracle returns $\perp$. Else, it parses and sets $(\mathsf{pk_{san}}^\dagger \| m_i, \mathsf{pk_{san}}^\dagger \| \mathsf{MOD}_i(m_i), \sigma_{SS,i}, \mathsf{spksig}_{,i})$, send it to $\Pi.\mathcal{O}^{San'}$ oracle as input, then it gets a signature $\sigma'_{SS,i}$. Then it parses $\sigma'_{SS,i}$ as $(\pi'_{SS,i},...)$, and runs $\sigma'_{TRS,i} \leftarrow \mathsf{TRS.Sign}(\mathsf{tsksan}^\dagger, (\mathsf{tpksig}_i, \mathsf{tpksan}^\dagger), \mathsf{pk_{sig}}^\dagger \| \mathsf{pk_{san}}^\dagger \| \pi'_{SS,i})$. Next, it returns $h_i := (\sigma'_{SS,i}, \sigma'_{TRS,i})$. If $(\mathsf{pk_{sig}}_{,i} = \mathsf{pk_{sig}}^\dagger) \wedge (\exists (m_i, h_i, \mathsf{ADM}'_i) \in R \wedge \mathsf{MOD}_i \in \mathsf{ADM}'_i$, then the oracle sets $R \leftarrow R \| (\mathsf{MOD_i}(m_i), h'_i, \mathsf{ADM}')$

$\mathcal{B}$ simulates $\mathcal{O}^{bLoRADM}$ as follows. On inout a tuple $(\mathsf{pk_{san}}, m, \mathsf{ADM}_0, \mathsf{ADM}_1)$,if $\neg(|m| = |\mathsf{ADM}_0| = |\mathsf{ADM}_1|)$ or $\mathsf{pk_{san}}^\dagger \neq \mathsf{pk_{san}} \wedge \mathsf{ADM}_0 \neq \mathsf{ADM}_1$, then the oracle returns $\perp$. Otherwise, it parses and sets $(\mathsf{spksan}, \mathsf{pk_{san}} \| m, 0 \| \mathsf{ADM}_0 \| \mathsf{ADM}_1)$ and send it to $\Pi.\mathcal{O}^{bLoRADM}$ oracle as input, then it gets a signature $\sigma_{SS,b}$. Then it parses $\sigma_{SS,b}$ as $(\pi_{SS,b}, ...)$, and runs $\sigma_{TRS,b} \leftarrow \mathsf{TRS.Sign}(\mathsf{tsksig}^\dagger, (\mathsf{tpksig}^\dagger, \mathsf{tpksan}), \mathsf{pk_{sig}}^\dagger \| \mathsf{pk_{san}} \| \pi_{SS,b})$. Next, it returns $\sigma_b := (\sigma_{SS,b}, \sigma_{TRS,b})$. If $\mathsf{pk_{san}}^\dagger = \mathsf{pk_{san}}$, it sets $R \leftarrow R \| (m, \sigma_b, \mathsf{ADM}_0 \circ \mathsf{ADM}_1)$

As it is clear, $\mathcal{B}$ can handle any oracle query and never aborts. So, $\mathcal{B}$ simulates all oracles perfectly for $\mathcal{A}$ who is able, and $\mathcal{B}$ can wins the invisibility game of $\Pi_{SS}$, this contradicts the invisibility of $\Pi_{SS}$. We thus show our construction $\Pi_{SBS}$ is invisible. □

**Lemma 3** Let $\Pi_{SS}$ be unlinkable, and let TRS be unforgeable, then the construction $\Pi_{SBS}$ is unlinkable.

**Proof** We use the definition of hybrid unlinkability experiments from [7] : i) $\mathsf{Hyb}_0^b$ : is identical to $ExpUnl^b_{\mathcal{A},\Pi_{SBS}}$. ii) $\mathsf{Hyb}_1^b$ : is identical to $\mathsf{Hyb}_0^b$ except for the following change. Let $(m_0, \mathsf{MOD}_0, \sigma_0, m_1, \mathsf{MOD}_1, \sigma_1)$ be a query from $\mathcal{A}$ to $\mathcal{O}^{bLoRsanit}$. Suppose that $\Pi_{SBS}.\mathsf{Verify}(\mathsf{pk_{sig}}^\dagger, \mathsf{pk_{san}}^\dagger, m_i, h_i) = 1$ for all $i \in \{0, 1\}$. Then if for some $i \in \{0, 1\}$, there is no $\mathsf{ADM}_i$ which satisfies $(\mathsf{pk_{sig}}^\dagger, \mathsf{pk_{san}}^\dagger, m_i, \mathsf{ADM}_i, \sigma_{SS,i}) \in L$, the challenger aborts.

Now we prove that if TRS is unforgeable, then $|\Pr\left[\mathsf{Hyb}_0^b = 1\right]$-$\Pr\left[\mathsf{Hyb}_1^b = 1\right]| \leq \mathrm{n}(\lambda)$. Assume a PPT adversary $\mathcal{A}$ that makes the challenger in $\mathsf{Hyb}_1^b$ to abort with non-negligible probability. Then, we show how to construct an adversary $\mathcal{B}$ that breaks the unforgeability of TRS. $\mathcal{B}$ receives $\mathsf{tpksig}, \mathsf{tpksan}$ as input, then it use $\Pi_{SS}.\mathsf{KGen}$ to generates $(\mathsf{spksig}, \mathsf{ssksig}), (\mathsf{spksan}, \mathsf{ssksan})$, then it lets $\mathsf{pk_{sig}}$ , $\mathsf{pk_{san}}$. Let $\mathcal{A}$ runs that $b' \leftarrow \mathcal{A}(\mathsf{pk_{sig}}, \mathsf{pk_{san}})$.

Then, $\mathcal{B}$ simulates three $\Pi_{SBS}$'s oracles $\Pi_{SBS}.\mathcal{O}^{Sign}$,$\Pi_{SBS}.\mathcal{O}^{San}$,$\Pi_{SBS}.\mathcal{O}^{bLoRSanit}$. i) $\mathcal{B}$ simulates $\mathcal{O}^{Sign}$ as follows. $\mathcal{B}$ simulates the $\mathcal{O}^{Sign}$ oracle honestly except that it generates $\sigma_{TRS}$ by using the TRS's oracle $\mathcal{O}^{TRS.Sign}$ on the input $(\{\mathsf{tpksig}, \mathsf{tpksan}\}, \mathsf{pk_{sig}} \| \mathsf{pk_{san}} \| \pi_{SS})$. ii) $\mathcal{B}$ simulates $\mathcal{O}^{San}$ as follows. $\mathcal{B}$ simulates the $\mathcal{O}^{San}$ oracle honestly except that it generates $\sigma_{TRS}$ by using the TRS's oracle $\mathcal{O}^{TRS.Sign}$ on the input $(\{\mathsf{tpksig}, \mathsf{tpksan}\}, \mathsf{pk_{sig}} \| \mathsf{pk_{san}} \| \pi'_{SS})$. iii) $\mathcal{B}$ simulates $\mathcal{O}^{bLoRSanit}$ as follows. Let $(m_0, \mathsf{MOD}_0, \sigma_0, m_1, \mathsf{MOD}_1, \sigma_1)$ be a query from $\mathcal{A}$ to $\mathcal{O}^{bLoRsanit}$. Suppose that $\Pi_{SBS}.\mathsf{Verify}(\mathsf{pk_{sig}}^\dagger, \mathsf{pk_{san}}^\dagger, m_i, h_i) = 1$ for all $i \in \{0, 1\}$. It means that $\mathsf{TRS.Verify}(\{\mathsf{tpksan}, \mathsf{tpksig}\}, \mathsf{pk_{sig}}^\dagger \| \mathsf{pk_{san}}^\dagger \| \pi_{SS,i}, \sigma_{TRS,i}) = 1$. Suppose that for some $i \in \{0, 1\}$, there is no $\mathsf{ADM}_i$ which satisfies $(\mathsf{pk_{sig}}^\dagger, \mathsf{pk_{san}}^\dagger, m_i, \mathsf{ADM}_i, h_i) \in L$. Then $\mathcal{B}$ aborts, and outputs $(\{\mathsf{tpksan}, \mathsf{tpksig}\}, \mathsf{pk_{sig}}^\dagger \| \mathsf{pk_{san}}^\dagger \| \pi_{SS,i}, \sigma_{TRS,i})$ as a forgery. Otherwise, $\mathcal{B}$ simulates the $\mathcal{O}^{bLoRSanit}$ oracle honestly except that it generates $\sigma_{TRS}$ by using the TRS's oracle $\mathcal{O}^{TRS.Sign}$ on the input $(\{\mathsf{tpksig}, \mathsf{tpksan}\}, \mathsf{pk}^\dagger_{\mathsf{sig}} \| \mathsf{pk_{san}}^\dagger \| \pi_{SS,b})$. Therefore, if $\mathcal{B}$ aborts, it contradicts the unforgeability of TRS. So we prove that if TRS is unforgeable, then $|\Pr\left[\mathsf{Hyb}_0^b = 1\right]$-$\Pr\left[\mathsf{Hyb}_1^b = 1\right]| \leq \mathrm{n}(\lambda)$. Combining this with the conclusion of [7] that if $\Pi_{SS}$ is weakly unlinkable, then $|\Pr\left[\mathsf{Hyb}_1^0 = 1\right]$-$\Pr\left[\mathsf{Hyb}_1^1 = 1\right]| \leq \mathrm{n}(\lambda)$, we get that our construction $\Pi_{SBS}$ is unlinkable. □

**Lemma 4** Let $\Pi_{SS}$ be transparent, and TRS is anonymous, then $\Pi_{SBS}$ is transparent.

**Proof** We use the definition of a hybrid transperency experiment in [7] : i) $\mathsf{Hyb}_0^b$: is identical to $ExpTran^b_{\mathcal{A},\Pi_{SBS}}$. ii)$\mathsf{Hyb}_1$ : is identical to $ExpTran^0_{\mathcal{A},\Pi_{SBS}}$ except for the following change. Let $(m_i, \mathsf{MOD}_i, \mathsf{ADM}_i)$ be a query from $\mathcal{A}$ to the oracle $\mathcal{O}^{bSanit/Sign}$ then returns $h_i = (\sigma_{SS,i}, \sigma_{TRS,i})$ including $\sigma_{TRS,i} \leftarrow \mathsf{TRS.Sign}(\mathsf{tsksig}, \{\mathsf{tpksig}, \mathsf{tpksan}\}, \mathsf{pk_{sig}} \| \mathsf{pk_{san}} \| \pi_{SS,i})$, which means that $\sigma_{TRS,i}$ is generated by the signer.

Then, we prove that if TRS is anonymous, then $|\Pr\left[\mathsf{Hyb}_1 = 1\right]$-$\Pr\left[\mathsf{Hyb}_0^0 = 1\right]| \leq \mathrm{n}(\lambda)$. Assume a PPT adversary $\mathcal{A}$ that sucessfully differs the two experiments, we construct a PPT adversary $\mathcal{B}$ that breaks the anonymity of TRS. $\mathcal{B}$ receives $(\mathsf{tpksig}^{\dagger}, \mathsf{tpksan}^{\dagger})$ as input, then generates signing and sanitizing key pairs $(\mathsf{spksig}^{\dagger}, \mathsf{ssksig}^{\dagger}, \mathsf{spksan}^{\dagger}, \mathsf{ssksan}^{\dagger})$ to set $\mathsf{pk}_{\mathsf{sig}}^{\dagger}, \mathsf{pk}_{\mathsf{san}}^{\dagger}$, then $B$ simulates the $\Pi_{SS}$'s oracles honestly but the TRS's oracles are replaced by the anonymity challenger of TRS.

Take $\mathcal{B}$ simulates the $\Pi_{SBS}.\mathcal{O}^{bSanit/Sign}$ as an example. On input a tuple $(m_i, \mathsf{MOD}_i, \mathsf{ADM}_i)$, $\mathcal{B}$ uses $\mathsf{Hyb}_0^0$ to compute $\sigma_{SS,i}$, parses $\sigma_{SS,i}$ into $(\pi_{SS,i}, ...)$ and sends $(\{\mathsf{tpksig}^{\dagger}, \mathsf{tpksan}^{\dagger}\}, \mathsf{pk}_{\mathsf{sig}} \| \mathsf{pk}_{\mathsf{san}}) \| \pi_{SS,i}$ to the $\mathcal{O}^{bSanit/Sign}$ oracle, then gets the answer $\sigma_{TRS,i}$, then it outputs $h_i = (\sigma_{SS,i}, \sigma_{TRS,i})$. Therefore, $\mathcal{B}$ can perfectly simulates either $\mathsf{Hyb}_0^0$ or $\mathsf{Hyb}_1$, if the probabilities that $\mathsf{Hyb}_0^0(1^\lambda)$ and $\mathsf{Hyb}_1(1^\lambda)$ output 1 differ in non-negligible probability, then $\mathcal{B}$ just breaks the anonymity of TRS, which is a contradiction.
Combining this with the conclusion of [7] that if $\Pi_{SS}$ is transparent, then $|\Pr[\mathsf{Hyb}_1 = 1]$-$\Pr[\mathsf{Hyb}_0^1 = 1]| \leq \mathrm{n}(\lambda)$, we get that $|\Pr[\mathsf{Hyb}_0^0 = 1]$-$\Pr[\mathsf{Hyb}_0^1 = 1]| \leq \mathrm{n}(\lambda)$, which shows that our construction $\Pi_{SBS}$ is transparent. □

**Lemma 5** Let TRS be tag-linkable, then $\Pi_{SBS}$ is responsible.

**Proof** We firstly consider Case 1, $\sigma_{true}$ originates from the signer. Assume a PPT adversary $\mathcal{A}$ that wins the responsibility game of $\Pi_{SBS}$ with non-negligible probability. Then, we show how to construct an adversary $\mathcal{B}$ that breaks the tag-linkability of TRS when $n = 1$. The algorithm $\mathcal{B}$ receives $\mathsf{tpksan}^{\dagger}$ as input, then it generates sanitizing key pairs $(\mathsf{spksan}^{\dagger}, \mathsf{ssksan}^{\dagger}) \leftarrow \Pi_{SS}.\mathsf{KGen}_{\mathsf{San}}$ and it lets $\mathsf{pk}_{\mathsf{san}}^{\dagger} = (\mathsf{spksan}^{\dagger}, \mathsf{tpksan}^{\dagger})$. Let $\mathcal{A}$ runs that $(\mathsf{pk_{sig}}^*, \sigma^*) \leftarrow \mathcal{A}(\mathsf{pk_{san}}^{\dagger})$.
Then, $\mathcal{B}$ simulates the $\Pi_{SBS}$'s sanitizing oracle $\Pi_{SBS}.\mathcal{O}^{San}$ as follows. On input a tuple $(\mathsf{pk}_{\mathsf{sig},i}, m_i, \mathsf{MOD}_i, h_i)$, the $\mathcal{O}^{San}$ parses and runs $\sigma'_{SS,i} \leftarrow \Pi_{SS}.\mathsf{Sanit}(\mathsf{ssksan}^{\dagger}, \mathsf{spksig}_i, m'_i, \mathsf{MOD}'_i, \sigma_{SS,i})$, and parses $\sigma'_{SS,i} = (\pi'_{SS,i}, ...)$, then sends $(\{\mathsf{tpksig}_i, \mathsf{tpksan}^{\dagger}\}, \mathsf{pk}_{\mathsf{sig},i} \| \mathsf{pk_{san}}^{\dagger} \| \pi'_{SS,i})$ to the TRS's signing oracle $\mathcal{O}^{TRS.Sign}$, and gets $\sigma'_{TRS,i}$. Next, it returns $h'_i := (\sigma'_{SS,i}, \sigma'_{TRS,i})$ to $\mathcal{A}$.
$\mathcal{B}$ gets $(\mathsf{pk_{sig}}^*, \sigma^*)$ from $\mathcal{A}$, and parses $\sigma^* = (\sigma^*_{SS}, \sigma^*_{TRS})$, then parses $\sigma^*_{SS} = (\pi^*_{SS}, ...)$ and sets the tag $L^{**} = (\mathsf{tpksig}^*, \mathsf{tpksan}^{\dagger})$,and $\pi^{**} = \mathsf{pk_{sig}}^* \| \mathsf{pk_{san}}^{\dagger} \| \pi^*_{SS}$, and $\sigma^{**} = \sigma^*_{TRS}$. Finally, $\mathcal{B}$ returns $(L^{**}, \pi^{**}, \sigma^{**})$. Assume that $\mathcal{A}$ wins the responsibility experiment of $\Pi_{SBS}$, then for all $i$ the following holds: i) it can reduce that $(\mathsf{pk_{sig}}^*, \mathsf{pk_{san}}^{\dagger}, m^*, \sigma^*_{SS}, \sigma^*_{TRS}) \neq (\mathsf{pk}_{\mathsf{sig},i}, \mathsf{pk_{san}}^{\dagger}, \mathsf{MOD}_i(m_i), \sigma'_{SS,i}, \sigma'_{TRS,i})$. It means that the adversary' role is malicious signer, and it has access to Sanitizing box. ii) $\Pi_{SBS}.\mathsf{Verify}(\mathsf{pk_{sig}}^*, \mathsf{pk_{san}}^{\dagger}, m^*, \sigma^*) = 1$, which means that $\mathsf{TRS.Verify}(L^{**}, \pi^{**}, \sigma^{**}) = 1$. iii) $\Pi_{SBS}.\mathsf{Trace}(\mathsf{pk_{sig}}^*, \mathsf{pk_{san}}^{\dagger}, \sigma_{true}, \sigma^*)$="`indep`", which implies that $\mathsf{TRS.Trace}(L^{**}, (\pi_{true}, \sigma_{true}), (\pi^{**}, \sigma^{**}))$= "`indep`". We review the adversarially-chosen-key-and-sub-ring attack in [4], and note that the adversary can append other public keys to the global public-key list, however, $\mathcal{B}$ only has been given one apriori target public key $\mathsf{tpksan}^{\dagger}$, so it satisfies the tag-linkability when $n = 1$. Therefore, if $\mathcal{A}$ breaks the responsibility of $\Pi_{SBS}$, it contradicts the tag-linkability of TRS when $n = 1$. The proof of tag-linkability is quite the same in Case 2 so it is omitted here, if $\sigma_{true}$ originates from the sanitizer, we can let $\mathcal{B}$ runs the SBS's signing oracle. We note all the private keys whose corresponding public keys of TRS are stored in the tag $T$ will hold tag-linkability. □.

**Lemma 6** Let TRS be exculpable, then $\Pi_{SBS}$ is exculpable.

**Proof** Assume a PPT adversary $\mathcal{A}$ that wins the exculpability game of $\Pi_{SBS}$ with non-negligible probability. Then, we show how to construct an adversary $\mathcal{B}$ that breaks the exculpability of TRS. The algorithm $\mathcal{B}$ receives $\mathsf{tpksig}^{\dagger}$ as input, then it generates signing key pairs $(\mathsf{spksig}^{\dagger}, \mathsf{ssksig}^{\dagger}) \leftarrow \Pi_{SS}.\mathsf{KGen}_{\mathsf{Sig}}$ and it lets $\mathsf{pk}_{\mathsf{sig}}^{\dagger} = (\mathsf{spksig}^{\dagger}, \mathsf{tpksig}^{\dagger})$. Let $\mathcal{A}$ runs that $(\mathsf{pk_{san}}^*, \sigma^*) \leftarrow \mathcal{A}(\mathsf{pk_{sig}}^{\dagger})$.
Then, $\mathcal{B}$ simulates the $\Pi_{SBS}$'s signing oracle $\Pi_{SBS}.\mathcal{O}^{Sign}$ as follows. On input a tuple $(\mathsf{pk}_{\mathsf{san},i}, m_i, \mathsf{ADM}_i)$, the $\mathcal{O}^{Sig}$ parses and sends $(\mathsf{ssksig}^{\dagger}, \mathsf{spksan}_i, \mathsf{pk}_{\mathsf{san}i} \| m_i, 0 \| \mathsf{ADM}_i)$ to the $\Pi_{SS}$'s sign oracle $\Pi_{SS}.\mathcal{O}^{Sign}$ that answers $\sigma_{SS,i}$, and parses $\sigma_{SS,i} = (\pi_{SS,i}, ...)$, then sends $(\{\mathsf{tpksig}^{\dagger}, \mathsf{tpksan}_i\}, \mathsf{pk_{sig}}^{\dagger} \| \mathsf{pk}_{\mathsf{san},i} \| \pi_{SS,i})$ to the TRS's signing oracle $\mathcal{O}^{TRS.Sign}$, and gets $\sigma_{TRS,i}$. Next, it returns $h_i := (\sigma_{SS,i}, \sigma_{TRS,i})$ to $\mathcal{A}$.
$\mathcal{B}$ gets $(\mathsf{pk_{san}}^*, \sigma^*)$ from $\mathcal{A}$, and parses $\sigma^* = (\sigma^*_{SS}, \sigma^*_{TRS})$, then parses $\sigma^*_{SS} = (\pi^*_{SS}, ...)$ and sets the tag $L^{**} = (\mathsf{tpksig}^{\dagger}, \mathsf{tpksan}^*)$,and $\pi^{**} = \mathsf{pk_{sig}}^{\dagger} \| \mathsf{pk_{san}}^* \| \pi^*_{SS}$, and $\sigma^{**} = \sigma^*_{TRS}$. Finally, $\mathcal{B}$ returns $(L^{**}, \pi^{**}, \sigma^{**})$. Assume that $\mathcal{A}$ wins the exculpability experiment of $\Pi_{SBS}$, then for all $i$ the following holds: i) it can reduce that $(\mathsf{pk_{sig}}^{\dagger}, \mathsf{pk_{san}}^*, m^*, \sigma^*_{SS}, \sigma^*_{TRS}) \neq (\mathsf{pk_{sig}}^{\dagger}, \mathsf{pk}_{\mathsf{san},i}, m_i, \sigma_{SS,i}, \sigma_{TRS,i})$. It means that the adversary's role is a malicious sanitiser, and it has access to the Signing box. ii) $\Pi_{SBS}.\mathsf{Verify}(\mathsf{pk_{sig}}^{\dagger}, \mathsf{pk_{san}}^*, m^*, \sigma^*) = 1$, which means that $\mathsf{TRS.Verify}(L^{**}, \pi^{**}, \sigma^{**}) = 1$. iii) $\Pi_{SBS}.\mathsf{Trace}(\mathsf{pk_{sig}}^{\dagger}, \mathsf{pk_{san}}^*, \sigma_{true}, \sigma^*) = \mathsf{pk_{sig}}^{\dagger}$, which implies that $\mathsf{TRS.Trace}(L^{**}, (\pi_{true}, \sigma_{true}), (\pi^{**}, \sigma^{**})) = \mathsf{tpksig}^{\dagger}$.
Therefore, if $\mathcal{A}$ breaks the exculpability of $\Pi_{SBS}$, it contradicts the exculpability of TRS. The proof of exculpability is quite the same in Case 2 so it is omitted here, if $\sigma_{true}$ originates from the sanitizer, we can let $\mathcal{B}$ runs the SBS's sanitizing oracle. We note all the private keys whose corresponding public keys of TRS are stored in the tag $T$ will hold exculpability. □

Finally, we introduce the corollary of [7], and by combining all the six lemmas and Corollary 1, Theorem 1 can be proven.

**Corollary 1** ( [7]) We assume that $\Pi_{SS}$ is in the random oracle model. If SPSEQ is EUF-CMA secure and perfectly adapts signatures, and TRS is anonymous, then the construction $\Pi_{SS}$ is unforgeable and transparent; if PKE is IND-CCA secure, then $\Pi_{SS}$ is invisible; if equivalence class relation $\mathcal{R}$ is class-hiding, SPSEQ perfectly adapt signatures, and PKE is correct and IND-CCA secure, and

TRS is unforgeable, then $\Pi_{\mathsf{SS}}$ is unlinkable; if TRS is tag-linkability, then $\Pi_{\mathsf{SS}}$ is responsible; if TRS is exculpable, then $\Pi_{\mathsf{SS}}$ is exculpable; if SPSEQ is EUF-CMA secure, then $\Pi_{\mathsf{SS}}$ is immutable.

## 5.2. EUF-CMA of SBS (Proof of Theorem 2)

Next, we prove that our proposed SBS signature scheme holds EUF-CMA security, which is crucial to the universal composability framework of PGUS.

**Proof:** We prove that a SBS scheme, which holds exculpability and responsibility, is also EUF-CMA secure. We show this by contraposition, constructing different attackers: (1) adversary $\mathcal{A}_{exc}$ for exculpability of SBS; (2) adversary $\mathcal{A}_{res}$ for responsibility of SBS; (3) adversary $\mathcal{A}_{eufcma}$ for EUF-CMA. We analyse the reduction between $\mathcal{A}_{exc}$, $\mathcal{A}_{res}$ and the successful attacker $\mathcal{A}_{eufcma}$ which wins the EUF-CMA experiment of SBS.

Firstly, we describe the adversary $\mathcal{A}_{exc}$, which acts as a dishonest sanitizer. $\mathcal{A}_{exc}$ receives a signing public key $\mathsf{pk_{sig}}^{\dagger}$ as input, and it runs the sanitizing key generation algorithm $\mathsf{KGen_{san}}$ to get a key pair $(\mathsf{pk_{san}}, \mathsf{sk_{san}})$, it begins to simulate adversary $\mathcal{A}_{eufcma}$ on $(\mathsf{pk_{sig}}^{\dagger}, \mathsf{pk_{san}})$. For each subsequent signing request, the adversary $\mathcal{A}_{exc}$ will forward the requests to its own oracle, and sends the reply to $\mathcal{A}_{eufcma}$. For each subsequent sanitizing request, the adversary $\mathcal{A}_{exc}$ keeps the same with the honest sanitizer, because $\mathcal{A}_{exc}$ knows sanitizing private key $\mathsf{sk_{san}}$. When the adversary $\mathcal{A}_{eufcma}$ outputs the forgery attempt $(m^*, \sigma^*)$, the adversary $\mathcal{A}_{exc}$ outputs $(\mathsf{pk_{san}}, m^*, \sigma^*)$ and stops.

Secondly, we describe the adversary $\mathcal{A}_{res}$, which acts as a dishonest signer. $\mathcal{A}_{res}$ receives a sanitizing public key $\mathsf{pk_{san}}^{\dagger}$ as input, and it runs the signing key generation algorithm $\mathsf{KGen_{sig}}$ to get a key pair $(\mathsf{pk_{sig}}, \mathsf{sk_{sig}})$, it begins to simulate adversary $\mathcal{A}_{eufcma}$ on $(\mathsf{pk_{sig}}, \mathsf{pk_{san}}^{\dagger})$. For each subsequent sanitizing request, the adversary $\mathcal{A}_{res}$ will forward the requests to its own oracle, and sends the reply to $\mathcal{A}_{eufcma}$. For each subsequent signing request, the adversary $\mathcal{A}_{res}$ keeps the same with the honest signer, because $\mathcal{A}_{res}$ knows signing private key $\mathsf{sk_{sig}}$. When the adversary $\mathcal{A}_{eufcma}$ outputs the forgery attempt $(m^*, \sigma^*)$, the adversary $\mathcal{A}_{res}$ outputs $(\mathsf{pk_{sig}}, m^*, \sigma^*)$ and stops.

Then, we assume that **bad** defines the case that $\mathcal{A}_{eufcma}$ in the simulation of $\mathcal{A}_{exc}$ or $\mathcal{A}_{res}$ outputs $(m^*, \sigma^*)$ which make the SBS.Verify algorithms return 1 in the experiments of exculpability or responsibility, but the message $m^*$ has never been submitted or answered to the signing oracle $\mathcal{O}^{Sign}$ or sanitizing oracle $\mathcal{O}^{Sanit}$. (The simulations of $\mathcal{A}_{exc}$ or $\mathcal{A}_{res}$ is the same, so **bad** event can represent both cases.) The probability that **bad** happens is equal to the probability that $\mathcal{A}_{eufcma}$ wins the EUF-CMA experiment (gets 1 in non-negligible probability).

We note that under the condition of **bad**, the probability for $ExpSigExculp_{\mathcal{A}_{exc},\Pi_{SBS}}(1^{\lambda}) = 1$, that SBS.Trace returns $\mathsf{pk_{sig}}^{\dagger}$, is exactly the probability for $ExpSigRespon_{\mathcal{A}_{res},\Pi_{SBS}}(1^{\lambda}) = 0$, that SBS.Trace returns `"indep"`. The only difference between these events is, in the exculpability experiment we require $(\mathsf{pk_{san}}, m^*)$ or $(\mathsf{pk_{sig}}, m^*)$ to be different from all queries/answers from the signing/sanitizing oracle. This difference should be ignored because the two conditions are all satisfied in the **bad** event. Also, **bad** avoids the case that SBS.Trace returns $\perp$, which avoiding the case that TRS.Trace returns `"linked"`, then:

$$\Pr\left[ExpSigExculp_{\mathcal{A}_{exc},\Pi_{SBS}}(1^{\lambda}) = 1\right] + \Pr\left[ExpSigRespon_{\mathcal{A}_{res},\Pi_{SBS}}(1^{\lambda}) = 1\right] \geq \Pr\left[ExpSigExculp_{\mathcal{A}_{exc},\Pi_{SBS}}(1^{\lambda}) = 1 | \mathbf{bad}\right] \cdot \Pr\left[\mathbf{bad}\right] + \Pr\left[ExpSigRespon_{\mathcal{A}_{res},\Pi_{SBS}}(1^{\lambda}) = 1 | \mathbf{bad}\right] \cdot \Pr\left[\mathbf{bad}\right] = \Pr\left[ExpSigExculp_{\mathcal{A}_{exc},\Pi_{SBS}}(1^{\lambda}) = 1 | \mathbf{bad}\right] \cdot \Pr\left[\mathbf{bad}\right] + (1 - \Pr\left[ExpSigExculp_{\mathcal{A}_{exc},\Pi_{SBS}}(1^{\lambda}) = 1 | \mathbf{bad}\right]) \cdot \Pr\left[\mathbf{bad}\right] = \Pr\left[\mathbf{bad}\right].$$

Therefore, if the bad event happens in non-negligible probability, then it would break exculpability and responsibility of SBS for at least one of them. Since SBS holds all the properties mentioned in Theorem 1, it should hold EUF-CMA security. ☐

# 6. Security Proof in UC Framework (Proof of Theorem 3)

In UC framework, $\mathcal{A}$ is the adversary interacts with real parties, and we will construct a simulator Sim that interfaces between adversary $\mathcal{A}$ and ideal functionality $\mathcal{F}_{Reg}$. We construct an ideal-world adversary Sim that runs a black-box simulation of the real-world adversary $\mathcal{A}$ by simulating the PGUS protocol execution and relaying messages between $\mathcal{A}$ and the environment $\mathcal{Z}$. Next, we consider a sequence of hybrid games to prove Theorem 3.

**Game $G_0$:** This is the original real game that corresponds to the real world in the model. This game executes the real protocol between sender $\mathcal{P}_i$ and receiver $\mathcal{P}_j$. The environment $\mathcal{Z}$ chooses the input for the honest sender $\mathcal{P}_i$, and $\mathcal{Z}$ receives the output of the honest sender. In our framework, there is an adversary $\mathcal{A}$ that aims to attack the real protocol in the real world by corrupting some parties and listening to all flows from parties. In that case, $\mathcal{A}$ can read the corrupted party's current inner state and fully control it, and $\mathcal{Z}$ can control adversary $\mathcal{A}$ and see all communication messages from all parties and also all of $\mathcal{A}$'s interactions with other parties.

**Game $G_1$:** We consider that the adversary $\mathcal{A}$ has controlled the gNB, and CN doesn't trust the gNB. Upon the simulator Sim receiving message $\mathbf{M}_1$ from the gNB, Sim firstly verifies the zero-knowledge proof $\pi_{ZK_{gNB}}$. After that, Sim uses SBS.Sign algorithm to generate a inner signature $\sigma_{inner}$ on dt, $\sigma_{inner} \leftarrow \mathsf{SBS.Sign}(sk_{sig}^{CN}, pk_{sig}^{gNB}, \mathsf{dt})$, then Sim sends the $\sigma_{inner}$ to gNB via $\mathbf{M}_2$.

**Lemma 7:** Game $G_0$ and Game $G_1$ are computationally indistinguishable for arbitrary environment $\mathcal{Z}$.

**Proof:** In Game $G_1$, the simulator Sim gets the $sk_{sig}^{CN}$ and parses it into $\mathsf{ssksig}, \mathsf{tsksig}$. After Sim receives the $\mathbf{M}_1$ from the corrupted gNB, checks this message and parses the message into $\mathsf{dt} \| com_{gNB} \| \pi_{ZK_{gNB}}$, and parses dt into $(X_1, ..., X_\ell), (Y_1, ..., Y_\ell)$. It use the ZK.Verify to check the validity of gNB identity. If the verification is successful, $\mathsf{Sim}$ runs $\mu \leftarrow \mathsf{SPSEQ.Sign}(\mathsf{ssksig}, (X_1, \ldots, X_\ell))$ and runs $\eta \leftarrow \mathsf{SPSEQ.Sign}(\mathsf{ssksig}, (Y_1, \ldots, Y_\ell))$. Then it generates the traceable ring signature $\sigma_{TRS} \leftarrow \mathsf{TRS.Sign}(\mathsf{tsksig}, (\mathsf{tpksig}, \mathsf{tpksan}), pk_{sig}^{CN} \| pk_{san}^{gNB} \| \mu \| \eta)$. Finally, the $\mathsf{Sim}$ can simulate the generation of inner signature $\sigma_{inner}$ and send it via $\mathbf{M}_2$. For any environment $\mathcal{Z}$, we consider the possible scenarios in which the adversary wins the indinstinguishability game, we use **bad** event to define these cases. We observe that the simulator reveals verified results after some party $\mathcal{P}_i'$ open commitment to the message. The **bad** happens if sender $\mathcal{P}_i$ successfully generates a $H_{gid}$ (which implies a valid $\pi_{ZK_{gNB}}$). This happens with a negligible probability because the commitment generated by the NIZK has the *sound* and *binding* properties. Also, due to the EUF-CMA security of the $\mathsf{SPSEQ}$ [15] and $\mathsf{TRS}$ [14] schemes, the simulated signature output $\sigma_{inner}$ is indistinguishable from a signature generated by CN in the real world. Therefore, two games $G_0$ and $G_1$ are computationally indistinguishable in a view of $\mathcal{Z}$. □

**Game** $G_2$**:** We consider that the adversary $\mathcal{A}$ has controlled CN, and gNB doesn't trust the CN. In the setup phase of this game, the simulator $\mathsf{Sim}$ chooses the common reference string $crs$, trapdoor $td$ and uses $\mathsf{SBS.Extract}$ algorithm to generate $(\mathsf{dt}, \mathsf{st})$ on the ADM and certificate $C_G$. After that, $\mathsf{Sim}$ randomly generate $u \leftarrow_\$ \{0,1\}^n$. Because $\mathsf{Sim}$ does not know the gNB's pseudo-identity $gid$, it generates the commitment $com_{gNB} = \mathsf{com}_{ck'}(0, u)$. Next, $\mathsf{Sim}$ generates the proof by using the simulator of NIZK: $\pi_{ZK_{gNB}} \leftarrow \mathsf{ZK.S}(td, com_{gNB}, crs, 0)$. Then, $\mathsf{Sim}$ sends $\mathsf{dt} \| com_{gNB} \| \pi_{ZK_{gNB}}$ to the CN via $\mathbf{M}_1$. In the later phase of this game, upon $\mathsf{Sim}$ receiving the message $\mathbf{M}_2$ from the CN, $\mathsf{Sim}$ uses $\mathsf{SBS.Derive}$ algorithm to derive the fixed signature $\sigma_{fix}$, and then $\mathsf{Sim}$ uses $\mathsf{SBS.Verify}$ algorithm to verify the sanitizable blind signature.

**Lemma 8:** Game $G_1$ and Game $G_2$ are computationally indistinguishable for arbitrary environment $\mathcal{Z}$.

**Proof:** In Game $G_2$, the simulator $\mathsf{Sim}$ simulates the $\mathbf{M}_1$ without knowing the pseduo-identifier $gid$, then, after it receives $\mathbf{M}_2 = \sigma_{inner}$, it use verification of $\mathsf{SBS}$ to judge the legitimacy of corrupted CN. For any environment $\mathcal{Z}$, we consider two possible scenarios in which the adversary wins the indinstinguishability game, we use **bad** event to define this kind of cases. (1) We observe that the simulator reveals verified results after some party $\mathcal{P}_i$ open commitment to the message. The **bad** happens if the receiver $\mathcal{P}_j$ successfully get value $H_{gid}$ or $C_{gNB}$. This happens with a negligible probability because the NIZK holds *zero-knowledge* property and the commitment holds *hiding* property, and also because $\mathsf{dt}$ does not contain any information about gNB's identity or $\mathsf{ADM}$. The data $\mathsf{dt}$ is computed by $x_i, y_i$ which are randomly chosen by gNB's in $\mathsf{SBS.Extract}$ algorithm. (2) The **bad** happens if the some sender $\mathcal{P}_i$ successfully generate a valid inner signature $\sigma_{inner}$, which means gNB can derive a valid fixed signature $\sigma_{fix}$, then it successfully passes the gNB's verification process $\mathsf{SBS.Verify}$. This happens with a negligible probability because the sanitizable blind signature $\mathsf{SBS}$ holds EUF-CMA security. Hence, from the proof above, the **bad** happens only with a negligible probability, and two games $G_0$ and $G_1$ are computationally indistinguishable in view of $\mathcal{Z}$. □

**Game** $G_3$**:** This game corresponds to the ideal world in the CRS model, there exists an ideal functionaility $\mathcal{F}_{Reg}$ and an honest task. Parties in the ideal world simply pass inputs from environment $\mathcal{Z}$ to the ideal world function and vice versa. In this game, the "ideal process adversary" simulator $\mathsf{Sim}$ proceeds following functions:

- **Initialisation step**: $\mathsf{Sim}$ chooses the $crs$, its $td$, and $ck$.
- **Simulating the communication with** $\mathcal{Z}$: Every input value that $\mathsf{Sim}$ receives from $\mathcal{Z}$ is written on $\mathcal{A}$'s input tape (as if coming from $\mathcal{Z}$) and vice versa.
- **Simulating the first round when sender** $\mathcal{P}_i$ **is honest**: Upon receiving the receipt message ($\mathsf{receipt}$, $sid$, $sgid$, $\mathcal{P}_i$, $\mathcal{P}_j$) from $\mathcal{F}_{Reg}$, $\mathsf{Sim}$ computes $\mathbf{M}_1$ like a honest party and sends ($\mathsf{message}_1$, $sid$, $sgid$, $\mathbf{M}_1$) to $\mathcal{P}_j$.
- **Simulating the second round when sender** $\mathcal{P}_i$ **is honest**: Upon receiving the receipt message ($\mathsf{receipt}$, $sid$, $sgid$, $\mathcal{P}_j$, $\mathcal{P}_i$) from $\mathcal{F}_{Reg}$, $\mathsf{Sim}$ computes $\mathbf{M}_2$ by $\mathsf{Com}_{ck'}(0, u)$ for randomly chosen $r$, and runs the NIZK simulator to compute the proof $\pi_{ZK_{gNB}}$ (with using the trapdoor $td$), then compute the $\mathsf{dt}$ use $\mathsf{SBS.Extract}$ algorithm, and sends ($\mathsf{message}_2$, $sid$, $sgid$, $\mathbf{M}_2$) to $\mathcal{P}_j$.
- **Simulating the commit phase when committer** $\hat{\mathcal{P}}_i$ **is corrupted and the receiver** $\mathcal{P}_j$ **is honest**: After receiving ($\mathsf{message}_k$, $sid$, $sgid$, $\mathbf{M}_k$) from $\hat{P}_i$ controlled by $\mathcal{A}$ in the round $k$, $\mathsf{Sim}$ runs the extractor of NIZK and compute $\mathbf{M}_k'$, and sends ($\mathsf{message}_k$, $sid$, $sgid$, $\mathbf{M}_k'$) to $\mathcal{F}_{Reg}$.

By the construction of the $\mathsf{Sim}$'s functions, since the parties in Game $G_3$ and the ideal functionality $\mathcal{F}_{Reg}$ run identical executions, the ideal functionality $\mathcal{F}_{Reg}$ and the game $G_3$ are computationally indistinguishable for any environment $\mathcal{Z}$. Furthermore, combining Lemma 1 and Lemma 2, we can conclude that, for any environment $\mathcal{Z}$, the probability distribution of outputting 1 after interacting with the hybrid adversary $\mathcal{A}$ and an instance of the *System Setup and Registration* phase of PGUS-AKA protocol is polynomially indistinguishable from the probability distribution of outputting 1 after interacting with the ideal adversary $\mathsf{Sim}$ and the ideal functionality $\mathcal{F}_{Reg}$. Theorem 1 is thus proved.

# References


[1] Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, Janno Siim, and Michal Zajkac. Dl-extractable uc-commitment schemes. In *Applied Cryptography and Network Security: 17th International Conference, ACNS 2019, Bogota, Colombia, June 5–7, 2019, Proceedings 17*, pages 385–405. Springer, 2019.

[2] Giuseppe Ateniese, Daniel H Chou, Breno De Medeiros, and Gene Tsudik. Sanitizable signatures. In *Computer Security–ESORICS 2005: 10th European Symposium on Research in Computer Security, Milan, Italy, September 12-14, 2005. Proceedings 10*, pages 159–177. Springer, 2005.

[3] Michael Till Beck, Jan Camenisch, David Derler, Stephan Krenn, Henrich C Pöhls, Kai Samelin, and Daniel Slamanig. Practical strongly invisible and strongly accountable sanitizable signatures. *Cryptology ePrint Archive*, 2017.

[4] Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*, pages 60–79. Springer, 2006.

[5] Christina Brzuska, Marc Fischlin, Tobias Freudenreich, Anja Lehmann, Marcus Page, Jakob Schelbert, Dominique Schröder, and Florian Volk. Security of sanitizable signatures revisited. In *Public Key Cryptography–PKC 2009: 12th International Conference on Practice and Theory in Public Key Cryptography, Irvine, CA, USA, March 18-20, 2009. Proceedings 12*, pages 317–336. Springer, 2009.

[6] Christina Brzuska, Marc Fischlin, Anja Lehmann, and Dominique Schröder. Unlinkability of sanitizable signatures. In *Public Key Cryptography–PKC 2010: 13th International Conference on Practice and Theory in Public Key Cryptography, Paris, France, May 26-28, 2010. Proceedings 13*, pages 444–461. Springer, 2010.

[7] Xavier Bultel, Pascal Lafourcade, Russell WF Lai, Giulio Malavolta, Dominique Schröder, and Sri Aravinda Krishnan Thyagarajan. Efficient invisible and unlinkable sanitizable signatures. In *Public-Key Cryptography–PKC 2019: 22nd IACR International Conference on Practice and Theory of Public-Key Cryptography, Beijing, China, April 14-17, 2019, Proceedings, Part I 22*, pages 159–189. Springer, 2019.

[8] Jan Camenisch, David Derler, Stephan Krenn, Henrich C Pöhls, Kai Samelin, and Daniel Slamanig. Chameleon-hashes with ephemeral trapdoors: And applications to invisible sanitizable signatures. In *Public-Key Cryptography–PKC 2017: 20th IACR International Conference on Practice and Theory in Public-Key Cryptography, Amsterdam, The Netherlands, March 28-31, 2017, Proceedings, Part II 20*, pages 152–182. Springer, 2017.

[9] Sanjit Chatterjee and Alfred Menezes. On cryptographic protocols employing asymmetric pairings—the role of $\psi$ revisited. *Discrete Applied Mathematics*, 159(13):1311–1322, 2011.

[10] Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Non-interactive zero-knowledge proof systems. In *Advances in Cryptology—CRYPTO'87: Proceedings 7*, pages 52–72. Springer, 1988.

[11] Nils Fleischhacker, Johannes Krupp, Giulio Malavolta, Jonas Schneider, Dominique Schröder, and Mark Simkin. Efficient unlinkable sanitizable signatures from signatures with re-randomizable keys. *Cryptology ePrint Archive*, 2015.

[12] Eiichiro Fujisaki. Sub-linear size traceable ring signatures without random oracles. *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, 95(1):151–166, 2012.

[13] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Annual international cryptology conference*, pages 537–554. Springer, 1999.

[14] Eiichiro Fujisaki and Koutarou Suzuki. Traceable ring signature. In *International Workshop on Public Key Cryptography*, pages 181–200. Springer, 2007.

[15] Christian Hanser and Daniel Slamanig. Structure-preserving signatures on equivalence classes and their application to anonymous credentials. In *Advances in Cryptology–ASIACRYPT 2014: 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, ROC, December 7-11, 2014. Proceedings, Part I 20*, pages 491–511. Springer, 2014.

[16] Mojtaba Khalili, Daniel Slamanig, and Mohammad Dakhilalian. Structure-preserving signatures on equivalence classes from standard assumptions. In *Advances in Cryptology–ASIACRYPT 2019: 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8–12, 2019, Proceedings, Part III 25*, pages 63–93. Springer, 2019.