Unlinkable Policy-Based Sanitizable Signatures

Ismail Afia[0000−0002−7669−8762] and Riham AlTawy[0000−0002−4734−3700]

University of Victoria, Victoria, BC V8P5C2, Canada
iafia@uvic.ca raltawy@uvic.ca

Abstract. In CT-RSA 2020, P3S was proposed as the first policy-based sanitizable signature scheme, allowing the signer to designate future message sanitizers by defining an access policy relative to their attributes rather than their keys. However, since P3S utilizes a policy-based chameleon hash (PCH), it does not achieve unlinkability, a required notion in privacy-preserving applications. Moreover, P3S requires running a procedure to share the secret trapdoor information for PCH with each new sanitizer before sanitizing a new message. We further observe that to maintain transparency in P3S's multiple sanitizers setting, the signature size should grow linearly with the number of sanitizers. In this work, we propose an unlinkable policy-based sanitizable signature scheme (UP3S) where we employ a rerandomizable digital signature scheme and a traceable attribute-based signature scheme as its building blocks. Compared to P3S, UP3S achieves unlinkability, does not require new secrets to be shared with future sanitizers prior to sanitizing each message, and has a fixed signature size for a given sanitization policy. We define and formally prove the security notions of the generic scheme, propose an instantiation of UP3S utilizing the Pointcheval-Sanders rerandomizable signature scheme and DTABS traceable attribute-based signature scheme, and analyze its efficiency. Finally, we compare UP3S with P3S in terms of the features of the procedures, scalability, and security models.

Keywords: sanitizable signature · attribute-based signatures · rerandomizable signature · policy-based signatures.

1 Introduction

Sanitizable signature schemes allow the signer of a message to designate a semi-trusted entity called the sanitizer to alter a signed message in a controlled way, and yet the original signature of the message is verified successfully [1]. The original signer of the message controls the modification process by defining which blocks of the message are allowed to be modified. Sanitizable signature schemes enabled numerous applications where the modification of a signed message is required without interaction with its signer. Such applications include outsourced databases, multicast transmissions, secure routing, privacy-preserving document disclosure, and privacy-preserving dissemination of patient data in healthcare applications [1, 10, 30].

The standard security notions of sanitizable signatures include unforgeability, immutability, privacy, accountability, and transparency. Additionally, unlinkability has been presented by Brzuska *et al.* as a required security notion for privacy-preserving applications [10, 1]. Intuitively, unlinkability ensures that associating different sanitized signatures with a source original message, i.e., linking the two sanitized versions of the same message, is not feasible. Hence, concluding combined information about the original message is prevented. For instance, in healthcare applications, if we have two sanitized message signature pairs of a specific patient's medical records where one of the messages contains only the personal information of the patient and the other is an anonymized version of the same patient's health records, linking both message signatures may lead to the reconstruction of the full medical records of the patient. Consequently, within the literature on sanitizable signature schemes [14, 16, 2, 18], constructing unlinkable ones has been the objective of the works in [12, 23, 27, 13]. Broadly, in the literature, several sanitizable signature schemes have been presented, which are classified by Bilzhause *et al.* [6] into four major categories as follows, i) schemes that provide additional security properties such as non-interactive public accountability [11] and invisibility [13, 14], ii) schemes that support multiple signers and sanitizers [16, 17, 9, 26], iii) schemes that limit the sanitizer ability to alter admissible blocks to signer chosen values [15, 20], and iv) schemes that allow the sanitization of encrypted data [2, 18].

Sanitizable signature schemes are usually defined in a single-signer single-sanitizer setting where the sanitizer is known in advance to the signer before the signature generation process. Conversely, trapdoor sanitizable signature schemes enable the signer to grant sanitization rights to sanitizer(s) after signature generation [16, 17]. However such schemes often require interaction between the sanitizer and the signer after signature generation to obtain trapdoor information [26]. To tackle the aforementioned limitation, Samelin and Slamanig proposed the first policy-based sanitizable signature scheme (P3S) where sanitization rights are assigned to any sanitizer that fulfills a predefined access policy [35]. Hence, sanitization is enabled based on possible sanitizer(s) attributes determined by the signer rather than sanitizers' public keys that may be unknown to the signer at the time of signing. Accordingly, sanitizers are not required to be known to the signer before signature generation. P3S employs a policy-based chameleon hash (PCH) [19] and a dynamic-group-signature similar primitive as its building blocks [4]. PCH allows sharing of the encryption of the trapdoor information of a chameleon hash function with possible sanitizers of a given message using an attribute-based encryption algorithm (ABE) where the sanitization policy controls who can decrypt the trapdoor [33]. On the other hand, P3S accountability is achieved by group signature similar primitive in which the signer/sanitizer of a given message provides the encryption of their public key in addition to a non-interactive zero-knowledge (NIZK) proof that the encryption hides either the signer or the sanitizer public key. Nevertheless, the use of PCH in P3S facilitates linking two signatures together because the message hash is not changed with each sanitization process. Moreover, for each new message,

the P3S setup has to be executed where the encryption of the PCH trapdoor secret key is shared with all sanitizers which do not lead to the most efficient instantiation. We also observe that to maintain the transparency security notion where it is infeasible to distinguish a freshly signed message from a sanitized one, the size of the group signature in P3S should grow linearly with the number of sanitizers which may further affect the system's scalability (see section 7.1).
Our Contributions. We present an Unlinkable Policy-based Signature Scheme (UP3S) that allows a signer to grant the sanitization rights of a specific document to sanitizers satisfying a predefined policy. UP3S ensures that the generated sanitized versions of such documents are unlinkable where it is infeasible to associate them with the same original document. We design UP3S such that for a given sanitization policy, the system setup is run once for the sanitization of all future messages and the signature size is fixed. We define the unlinkability, unforgeability, immutability, transparency, privacy, and accountability security notions for the generic UP3S, and prove that it achieves them. Moreover, we instantiate UP3S with Pointcheval-Sanders rerandomizable signature scheme [31] and DTABS [24], and analyze its performance. Finally, We compare it with P3S in terms of the schemes' properties, scalability, and security.

2 Preliminaries and Building Blocks

Let $i \in \mathbb{I}$ denote an identity i from the identity universe $\mathbb{I}$ and $\mathbb{S} \subseteq \mathbb{U}$ denote an attribute set $\mathbb{S}$ from the attribute universe $\mathbb{U}$. Let $\lambda \in \mathbb{N}$ denote our security parameter, then a function $\epsilon(\lambda) : \mathbb{N} \rightarrow [0, 1]$ denotes the negligible function if for any $c \in \mathbb{N}$, $c > 0$ there exists $\lambda_c \in \mathbb{N}$ s.t. $\epsilon(\lambda) < \lambda^{-c}$ for all $\lambda > \lambda_c$. For a message $m = (m_1, m_2, \ldots, m_l) \in \mathbb{M}^l$, let m_i denotes a message block, and the variable $adm = (\{A \subseteq \{1, \ldots, l\}\}, l)$ specifies the set of indices A of the modifiable blocks over m which contains l blocks each of size n bits. We use $\mathsf{Adm}(m) = 1$ if adm is valid with respect to m, i.e., it contains a subset A of $\{0, ..., l\}$ and m contains exactly l blocks. Let m_{adm} denote the list of blocks in m which are admissible with respect to adm. We denote the list of blocks in m which are not admissible under adm with $m_{\not\in adm}$. The function $m' \leftarrow \mathsf{MoD}(m, adm, mod)$ is used to modify the message m by applying the modifications mod on the admissible block(s) adm and outputs the modified message m', where mod is a set that contains the tuple (i, m_i, m'_i) for $i \in adm$. Furthermore, we write $\mathsf{CheckMoD}(m, adm) = 1$, if adm is valid with respect to m, i.e., $\mathsf{Adm}(m) = 1$ and the blocks indices to be modified in a message m are contained in adm as admissible. For each message m, there is one signer and one or more sanitizer(s) who can sanitize m by running $m' \leftarrow \mathsf{MoD}(m, adm, mod)$ and generating a valid sanitized signature on m' depending on their attributes. Finally, to denote that an attribute set $\mathbb{S}$ satisfies a monotone access structure predicate Υ (see Def. 1), we use $\Upsilon(\mathbb{S}) = 1$.

Definition 1. *(Access Structure [5]). Let $\mathbb{U}$ denote the universe of attributes. A collection $\mathbb{A} \in 2^{\mathbb{U}} \setminus \{0\}$ of a non-empty set is an access structure on $\mathbb{U}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the*

unauthorized sets. The collection $\mathbb{A}$ *is called monotone if* $\forall B, C \in \mathbb{A}$: *if* $B \in \mathbb{A}$ *and* $B \subseteq C$*, then* $C \in \mathbb{A}$.

2.1 Rerandomizable Digital Signature (RDS) Scheme

RDS schemes are digital signature algorithms that allow rerandomizing a signature such that the rerandomized version of the signature is still verifiable under the verification key of the signer [36]. An important property of RDS schemes is that the rerandomized signatures produced using the same signing key on the same message are indistinguishable from a freshly signed one [31]. An RDS scheme is a tuple of five polynomial-time algorithms, RDS = {ppGenRDS, KeyGenRDS, SignRDS, RandomizeRDS, VerifyRDS} which are defined as follows.

- ppGenRDS. This algorithm outputs the public parameters of the scheme, $pp_{RDS} \leftarrow \mathsf{ppGenRDS}(1^{\lambda})$.

- KeyGenRDS. This procedure generates the signer's secret and public key pair, $(sk_{RDS}, pk_{RDS}) \leftarrow \mathsf{KeyGenRDS}(pp_{RDS})$.

- SignRDS. This procedure generates a digital signature σ_{RDS} on a message m, $\sigma_{RDS} \leftarrow \mathsf{SignRDS}(sk_{RDS}, m)$.

- VerifyRDS. This algorithm verifies the (rerandomized) signature σ_{RDS} over m, $\{\top, \bot\} \leftarrow \mathsf{VerifyRDS}(pk_{RDS}, m, \sigma_{RDS})$.

- RandomizeRDS. This procedure rerandomizes the digital signature σ_{RDS} on a message m and outputs σ'_{RDS}, $\sigma'_{RDS} \leftarrow \mathsf{RandomizeRDS}(m, \sigma_{RDS})$.

RDS schemes ensure both existential unforgeability under chosen message attacks (EUF-CMA) and unlinkability. The formal definition of both security notions, their associated experiments, and security oracles, are given in [31, 36] and in Appendix A.

2.2 Traceable Attribute-based Signatures (TABS)

Attribute-based signature (ABS) schemes are probabilistic digital signature schemes in which the produced signature attests a specific claim predicate (Υ) regarding the attributes that the signer possesses rather than the identity of the signer [28]. ABS schemes ensure privacy where the signer's identity is anonymous among all the users who possess a set of attributes that satisfy the claim predicate specified in the signature. Such schemes utilize a trusted entity called the Attribute Authority (AA) to authenticate users' identities and issue their corresponding attributes. Traceable ABS (TABS) schemes are a variant of ABS schemes where tracing a signature to its original signer is supported [21]. In such schemes, tracing could be performed by AA or another tracing authority (TA) [21, 22]. A TABS scheme is a tuple of eight polynomial-time algorithms, TABS = {ppGenTABS, TAKeyGenTABS, AAKeyGenTABS, SignerKeyGenTABS, SignTABS, VerifyTABS, TraceTABS, JudgeTABS} which are specified as follows.

- ppGenTABS. This algorithm outputs the public parameters of the scheme pp_{TABS} which also defines both the identity universe $\mathbb{I}$ and the attribute universe $\mathbb{U}$, $pp_{TABS} \leftarrow \mathsf{ppGenTABS}(1^{\lambda})$.

- TAKeyGenTABS. This algorithm is run by the TA and outputs a tracing key tsk^{TA}_{TABS} for the tracing authority, $tsk^{TA}_{TABS} \leftarrow \mathsf{TAKeyGenTABS}(pp_{TABS})$.

- AAKeyGenTABS. This algorithm is run by the AA to generate its public key and master secret key pair, $(pk_{TABS}, msk^{AA}_{TABS}) \leftarrow \mathsf{AAKeyGenTABS}(pp_{TABS})$.

- SignerKeyGenTABS. This algorithm is run by the AA, on the attribute set $\mathbb{S}_i \subset \mathbb{U}$ and $i \in \mathbb{I}$ for a specific user and the AA master secret key msk^{AA}_{TABS}. It outputs the user's secret key, $sk^{User,i}_{TABS} \leftarrow \mathsf{SignerKeyGenTABS}(pp_{TABS}, msk^{AA}_{TABS}, i, \mathbb{S}_i)$.

- SignTABS. This algorithm is run by the signer on a message $m \in \{0,1\}^*$ for a claim predicate Υ where the user possesses a set of attributes $\mathbb{S}'_i \subseteq \mathbb{S}_i$ satisfying the claim predicate Υ, i.e., $\Upsilon(\mathbb{S}'_i) = 1$. It outputs a signature, $\sigma_{TABS} \leftarrow \mathsf{SignTABS}(pp_{TABS}, sk^{User,i}_{TABS}, m, \Upsilon)$.

- VerifyTABS. This algorithm verifies the signature σ_{TABS} over m with respect to a claim predicate Υ, $\{\top, \bot\} \leftarrow \mathsf{VerifyTABS}(pp_{TABS}, pk_{TABS}, m, \sigma_{TABS}, \Upsilon)$.

- TraceTABS. The TA runs this algorithm to trace a signature tuple $(m, \sigma_{TABS}, \Upsilon)$ to its actual signer. It outputs the identity of the signer along with NIZK proof π, attesting to this claim, $(i, \pi) \leftarrow \mathsf{TraceTABS}(tsk^{TA}_{TABS}, m, \sigma_{TABS}, \Upsilon)$.

- JudgeTABS. This algorithm outputs true if it verifies that π proves that i is the identity of the signer who produced σ_{TABS} on m,
$\{\top, \bot\} \leftarrow \mathsf{JudgeTABS}(pp_{TABS}, pk_{TABS}, m, \sigma_{TABS}, \Upsilon, i, \pi)$.

The security definitions for unforgeability, privacy, traceability, and non-frameability which are the security notions required to prove the security of UP3S are defined in [24] and are also given in Appendix B.

3 UP3S Black-box Construction

The main idea behind the proposed construction is that after signers and sanitizers acquire their respective secret keys, the signer of a given message defines a policy Υ that controls the sanitization rights of such a message, i.e., any sanitizer who possesses an attribute-set satisfying Υ is able to generate a sanitized version of such a message without interaction with the AA or the signer. Formally, UP3S scheme is a tuple of nine polynomial-time algorithms, UP3S = {ParGenUP3S, SetupUP3S, KGenSignUP3S, KGenSanUP3S, SignUP3S, SanitizeUP3S, VerifyUP3S, ProveUP3S, JudgeUP3S}. The specifications of the aforementioned algorithms are as follows:

- ParGenUP3S. This algorithm returns the scheme's public parameters which become implicit input for all UP3S algorithms. It also defines the identity universe $\mathbb{I}$ and the attribute universe $\mathbb{U}$. $pp_{\mathsf{UP3S}} \leftarrow \mathsf{ParGenUP3S}(1^\lambda)$

- SetupUP3S. This algorithm outputs the global public key pk_{UP3S} and the master secret key sk_{UP3S} of the scheme. $(pk_{\mathsf{UP3S}}, sk_{\mathsf{UP3S}}) \leftarrow \mathsf{SetupUP3S}(pp_{\mathsf{UP3S}})$

- KGenSignUP3S. This algorithm generates the public-secret key pairs of a signer with identity $i_{Sign} \in \mathcal{I}$ who holds an attribute-set $\mathbb{S}_{Sign} \subset \mathbb{U}$.
$(pk^{Sign}_{\mathsf{UP3S}}, sk^{Sign}_{\mathsf{UP3S}}) \leftarrow \mathsf{KGenSignUP3S}(sk_{\mathsf{UP3S}}, i_{Sign}, \mathbb{S}_{Sign})$

- KGenSanUP3S. This algorithm generates the secret key of a sanitizer with identity $i_{San} \in \mathcal{I}$ who holds an attribute-set $\mathbb{S}_{San} \subset \mathbb{U}$.
$sk^{San}_{\mathsf{UP3S}} \leftarrow \mathsf{KGenSanUP3S}(sk_{\mathsf{UP3S}}, i_{San}, \mathbb{S}_{San,i})$

- SignUP3S. This algorithm generates a sanitizable signature σ_m using the signer's key $sk^{Sign}_{\mathsf{UP3S}}$ on a message m, given the set of indices of the modifiable blocks adm, a predicate Υ, and the attribute-set of possible future sanitizer(s) $\mathbb{S}_{PSan} \subseteq \mathbb{U}$. $(m, \sigma_m, adm, \Upsilon) \leftarrow \mathsf{SignUP3S}(pk_{\mathsf{UP3S}}, sk^{Sign}_{\mathsf{UP3S}}, m, adm, \mathbb{S}_{PSan})$

- VerifyUP3S. This algorithm verifies a signature σ_m on a message m, a set of indices of the modifiable blocks adm and a predicate Υ, using the scheme's public key pk_{UP3S} and the signer's public key $pk^{Sign}_{\mathsf{UP3S}}$.
$\{\top, \bot\} \leftarrow \mathsf{VerifyUP3S}(pk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}}, m, \sigma_m, adm, \Upsilon)$

- SanitizeUP3S. This algorithm generates a sanitized signature σ'_m using the sanitizer secret key sk^{San}_{UP3S} on a signature σ_m, the original message m which is modified to $m' \leftarrow MoD(m, adm, mod)$, the set of indices of the modifiable blocks adm, a predicate Υ, the scheme public key pk_{UP3S}, and the signer public key $pk^{Sign}_{\mathsf{UP3S}}$.
$(m', \sigma'_m, adm, \Upsilon) \leftarrow \mathsf{SanitizeUP3S}(pk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}}, sk^{San}_{\mathsf{UP3S}}, m, \sigma_m, adm, mod, \Upsilon)$

- ProveUP3S. This algorithm outputs the identity i of the signer (resp. sanitizer) of a specific message signature tuple $(pk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}}, m, \sigma_m, adm, \Upsilon)$ along with a NIZK proof π which proves that i is the signer who generated σ_m on m.
$\{i, \pi\} \leftarrow \mathsf{ProveUP3S}(pk_{\mathsf{UP3S}}, sk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}}, m, \sigma_m, adm, \Upsilon)$

- JudgeUP3S. This algorithm verifies the proof π on a specific message signature tuple $(m, \sigma_m, adm, \Upsilon)$ and an identity i.
$\{\top, \bot\} \leftarrow \mathsf{JudgeUP3S}(pk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}}, m, \sigma_m, adm, \Upsilon, i, \pi)$

UP3S **Correctness**. For the correctness of UP3S, we require that, for all $\lambda \in \mathbb{N}$, for all $pp_{\mathsf{UP3S}} \leftarrow \mathsf{ParGenUP3S}(1^\lambda)$, for all $(pk_{\mathsf{UP3S}}, sk_{\mathsf{UP3S}}) \leftarrow \mathsf{SetupUP3S}\ (pp_{\mathsf{UP3S}})$, for all $i_{Sign} \in \mathbb{I}$, for all $\mathbb{S}_{Sign} \subseteq \mathbb{U}$, for all $(sk^{Sign}_{\mathsf{UP3S}}) \leftarrow \mathsf{KGenSignUP3S}\ (sk_{\mathsf{UP3S}}, i_{Sign}, \mathbb{S}_{Sign})$, for all $l \in \mathbb{N}$, for all $m \in \mathbb{M}^l$, for all $\mathbb{S}_{PSan} \in \mathbb{U}$, for all $\Upsilon \in 2^{\mathbb{U}} \mid \Upsilon(\mathbb{S}_{Sign}) = 1$, for all $adm = (\{A \subseteq \{1, \ldots, l\}\}, l)$ such that $\mathsf{Adm}(m) = 1$, for all

$(m, \sigma_m, adm, \Upsilon) \leftarrow \mathsf{SignUP3S}$ $(pk_{\mathsf{UP3S}}, sk^{Sign}_{\mathsf{UP3S}}, m, adm, \mathbb{S}_{PSan})$, for all $i_{San} \in \mathbb{I}$, for all $\mathbb{S}_{San} \subseteq \mathbb{U} \mid \Upsilon(\mathbb{S}_{San}) = 1$, for all $sk^{San}_{\mathsf{UP3S}} \leftarrow \mathsf{KGenSanUP3S}$ $(sk_{\mathsf{UP3S}}, i_{San}, \mathbb{S}_{San,i})$, for all $mod = \{mod_i\} \mid mod_i = (i, m_i, m'_i) \forall i \in adm$, for all $m' \leftarrow \mathsf{MoD}(m, adm, mod) \mid \mathsf{CheckMoD}(m, adm) = 1$, and for all $(m', \sigma'_m, adm, \Upsilon) \leftarrow \mathsf{SanitizeUP3S}(pk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}}, sk^{San}_{\mathsf{UP3S}}, m, \sigma_m, adm, mod, \Upsilon)$, we have $\top = \mathsf{VerifyUP3S}(pk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}}, m, \sigma_m, adm, \Upsilon)$ and $\top = \mathsf{VerifyUP3S}(pk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}}, m', \sigma'_m, adm, \Upsilon)$.
Furthermore, for all $\{i, \pi\} \leftarrow \mathsf{ProveUP3S}(pk_{\mathsf{UP3S}}, sk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}}, m, \sigma_m, adm, \Upsilon)$, and $\{i', \pi'\} \leftarrow \mathsf{ProveUP3S}(pk_{\mathsf{UP3S}}, sk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}}, m', \sigma'_m, adm, \Upsilon)$, we have $\top = \mathsf{JudgeUP3S}(pk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}}, m, \sigma_m, adm, \Upsilon, i, \pi)$ and $\top = \mathsf{JudgeUP3S}(pk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}}, m', \sigma'_m, adm, \Upsilon, i', \pi')$.

3.1 UP3S Security Definitions

In what follows, we define the required security notion of UP3S. We use the same notations as in [35, 8, 10] for ease of readability. The oracles used in the security experiments are defined in Fig. 1. All the security experiments are initialized by running the following setup and key generation procedures.

$$
\begin{aligned}
pp_{\mathsf{UP3S}} &\leftarrow \mathsf{ParGenUP3S}(1^{\lambda}) \\
(pk_{\mathsf{UP3S}}, sk_{\mathsf{UP3S}}) &\leftarrow \mathsf{SetupUP3S}() \\
(sk^{Sign}_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}}) &\leftarrow \mathsf{KGenSignUP3S}(sk_{\mathsf{UP3S}}, i_{Sign}, \mathbb{S}_{Sign,i}) \\
(sk^{San}_{\mathsf{UP3S}}) &\leftarrow \mathsf{KGenSanUP3S}(sk_{\mathsf{UP3S}}, i_{San}, \mathbb{S}_{San,i})
\end{aligned}
$$

The $\mathcal{O}$SignUP3S, $\mathcal{O}$SanitUP3S, $\mathcal{O}$ProveUP3S, $\mathcal{O}$LoRSanitUP3S, $\mathcal{O}$LoRSignSanitUP3S, and $\mathcal{O}$Sign-or-SanitUP3S oracles are implicitly initialized with the secrets $sk^{Sign}_{\mathsf{UP3S}}$, sk^{San}_{UP3S}, sk_{UP3S}, sk^{San}_{UP3S}, $(sk^{Sign}_{\mathsf{UP3S}}, sk^{San}_{\mathsf{UP3S}})$, and $(sk^{Sign}_{\mathsf{UP3S}}, sk^{San}_{\mathsf{UP3S}})$, respectively. Moreover, $\mathcal{O}$LoRSanitUP3S, $\mathcal{O}$LoRSignSanitUP3S, and $\mathcal{O}$Sign-or-SanitUP3S oracles are further initialized with a secret bit b that is randomly chosen in the experiments, thus we pass it as a secret input after it gets selected. Note that the attribute sets $\mathbb{S}_{Sign,i}$ and $\mathbb{S}_{San,i}$ used in the experiments initialization are selected such that $\Upsilon(\mathbb{S}_{Sign,i}) = 1$ and $\Upsilon(\mathbb{S}_{San,i}) = 1$ for those oracles queried by the adversary with any Υ.
Unlinkability. Unlinkability is defined using the experiment in Fig 2, where the adversary has access to left-or-right sanitization oracle $\mathcal{O}$LoRSanitUP3S (see Fig. 1) among other oracles. The adversary inputs two sanitizable messages-signature pairs $\{(m_0, \sigma_{m0}), (m_1, \sigma_{m1})\}$ along with their modifications to $\mathcal{O}$LoRSanitUP3S, the oracle is initialized with a secret random bit '$b \in \{0, 1\}$'. Depending on 'b', the oracle outputs a sanitized signature of either the left or right input message signature pair. The adversary wins if it could determine which pair is used in the sanitization process with probability better than the random guess. The adversary is restricted to inputting two messages such that their modified outputs are the same $m'_0 = m'_1$ to prevent linking a sanitized message to its original source. To achieve such a restriction, the adversary must input two messages with identical fixed parts, $m_{0_{\lfloor adm}} = m_{1_{\lfloor adm}}$ and the two messages' admissible blocks indices must be the same, i.e., $adm_0 = adm_1$. To match

$\mathcal{O}\mathsf{SignUP3S}(pk'_{\mathsf{UP3S}}, m, adm, \mathbb{S}_{PSan})$

if $pk'_{\mathsf{UP3S}} = pk_{\mathsf{UP3S}}$
  $(m, \sigma_m, adm, \Upsilon) \leftarrow \mathsf{SignUP3S}(pk_{\mathsf{UP3S}}, sk^{Sign}_{\mathsf{UP3S}}, m, adm, \mathbb{S}_{PSan})$
  $\mathcal{M} = \mathcal{M} \cup \{m, adm, \sigma_m, \Upsilon\}$
  $\mathcal{S} = \mathcal{S} \cup \{\mathbb{S}_{PSan}\}$
  return $(m, \sigma_m, adm, \Upsilon)$
return 0

$\mathcal{O}\mathsf{SanitUP3S}(pk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}}, m, \sigma_m, adm, mod, \Upsilon)$

$(m', \sigma'_m, adm, \Upsilon) \leftarrow \mathsf{SanitizeUP3S}(pk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}}, sk^{San}_{\mathsf{UP3S}}, m, \sigma_m, adm, mod, \Upsilon)$
$\mathcal{L} = \mathcal{L} \cup \{m', \sigma'_m, adm, \Upsilon\}$
return $(m', \sigma'_m, adm, \Upsilon)$

$\mathcal{O}\mathsf{ProveUP3S}(pk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}}, m, \sigma_m, adm, \Upsilon)$

if $(m, \sigma_m) \notin \mathcal{T}$
  return $(i, \pi) \leftarrow \mathsf{ProveUP3S}(pk_{\mathsf{UP3S}}, sk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}}, m, \sigma_m, adm, \Upsilon)$
return 0

$\mathcal{O}\mathsf{LoRSanitUP3S}(pk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}}, m_0, \sigma_{m0}, adm_0, \Upsilon_0, mod_0, m_1, \sigma_{m1}, adm_1, \Upsilon_1, mod_1)$

if $\mathsf{MoD}(m_0, adm_0, mod_0) = \mathsf{MoD}(m_1, adm_1, mod_1) \wedge \Upsilon_0 = \Upsilon_1 \wedge adm_0 = adm_1$
  if $\mathsf{VerifyUP3S}(pk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}}, m_0, \sigma_{m0}, adm_0, \Upsilon_0) \wedge \mathsf{VerifyUP3S}(pk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}}, m_1, \sigma_{m1}, adm_1, \Upsilon_1)$
    return $(m'_b, \sigma'_{mb}, adm_b, \Upsilon_b) \leftarrow \mathsf{SanitizeUP3S}(pk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}}, sk^{San}_{\mathsf{UP3S}}, m_b, \sigma_{mb}, adm_b, mod_b, \Upsilon_b)$
return 0

$\mathcal{O}\mathsf{LoRSignSanitUP3S}(m_0, adm_0, \Upsilon_0, mod_0, \mathbb{S}_{PSan,0}, m_1, adm_1, \Upsilon_1, mod_1, \mathbb{S}_{PSan,1})$

if $\mathsf{MoD}(m_0, adm_0, mod_0) = \mathsf{MoD}(m_1, adm_1, mod_1) \wedge \Upsilon_0 = \Upsilon_1 \wedge adm_0 = adm_1 \wedge \mathbb{S}_{PSan,0} = \mathbb{S}_{PSan,1}$
  $(m_b, \sigma_{mb}, adm_b, \Upsilon_b) \leftarrow \mathsf{SignUP3S}(pk_{\mathsf{UP3S}}, sk^{Sign}_{\mathsf{UP3S}}, m_b, adm_b, \mathbb{S}_{PSan,b})$
  return $(m'_b, \sigma'_{mb}, adm_b, \Upsilon_b) \leftarrow \mathsf{SanitizeUP3S}(pk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}}, sk^{San}_{\mathsf{UP3S}}, m_b, \sigma_{mb}, adm_b, mod_b, \Upsilon_b)$
return 0

$\mathcal{O}\mathsf{Sign\text{-}or\text{-}SanitUP3S}(m, adm, \mathbb{S}_{PSan}, mod)$

if $b = 0$
  $m' \leftarrow \mathsf{MoD}(m, mod, adm)$
  $(m', \sigma'_m, adm, \Upsilon) \leftarrow \mathsf{SignUP3S}(pk_{\mathsf{UP3S}}, sk^{Sign}_{\mathsf{UP3S}}, m', adm, \mathbb{S}_{PSan})$
else $(m, \sigma_m, adm, \Upsilon) \leftarrow \mathsf{SignUP3S}(pk_{\mathsf{UP3S}}, sk^{Sign}_{\mathsf{UP3S}}, m, adm, \mathbb{S}_{PSan})$
  $(m', \sigma'_m, adm, \Upsilon) \leftarrow \mathsf{SanitizeUP3S}(pk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}}, sk^{San}_{\mathsf{UP3S}}, m, \sigma_m, adm, mod, \Upsilon)$
$\mathcal{T} = \mathcal{T} \cup \{m'_b, \sigma'_{mb}\}$
return $(m', \sigma'_m, adm, \Upsilon)$

Fig. 1: UP3S security experiments oracles

UP3S's policy-based expressiveness, we further restrict the adversary to input two messages that could be sanitized under the same predicate, i.e., $\Upsilon_1 = \Upsilon_2$. Note that, unlike group signature schemes where unlinkability is defined as the infeasibility to link two messages and their signatures to the same signer [3], in sanitizable signature, unlinkability is defined as the infeasibility to link signatures of two or more sanitized versions of a message to the same source message [10].

Definition 2. *(Unlinkability)* UP3S *scheme is unlinkable if for any PPT adversary* $\mathcal{A}$, $\left|\mathsf{Pr}[\boldsymbol{Exp}^{Unlinkability}_{\mathcal{A},\mathsf{UP3S}}(\lambda) = 1] - \frac{1}{2}\right| \leq \epsilon(\lambda)$, *where the unlinkability experiment is described in Fig. 2.*

$\mathbf{Exp}^{Unlinkability}_{\mathcal{A},\mathsf{UP3S}}(\lambda)$

$b \xleftarrow{\$} \{0,1\}$

$a \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{SignUP3S}(.),\mathcal{O}\mathsf{SanitUP3S}(.),\mathcal{O}\mathsf{ProveUP3S}(.),\mathcal{O}\mathsf{LoRSanitUP3S}(.,b)}(pk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}})$

if $a = b$

  return 1

return 0

Fig. 2: UP3S unlinkability experiment.

Transparency. This notion requires that no adversary can distinguish between sanitizable signatures created by the signer or the sanitizer. Transparency is modeled by the experiment in Fig. 3 in which adversary $\mathcal{A}$ has access to $\mathcal{O}$SignUP3S, $\mathcal{O}$SanitUP3S , and $\mathcal{O}$ProveUP3S. At the end, $\mathcal{A}$ queries $\mathcal{O}$Sign-or-SanitUP3S with a message m, a modification mod, possible sanitizers attribute set $\mathbb{S}_{PSan}$ and the set of indices of the modifiable blocks adm. $\mathcal{O}$Sign-or-SanitUP3S which is initialized by a secret random bit b, outputs the signature tuple $(m', \sigma'_m, adm, \Upsilon)$ as follows.

- For $b = 0$, $m' \leftarrow \mathsf{MoD}(m, adm, mod)$, $\mathcal{O}$Sign-or-SanitUP3S runs the signing algorithm to create $(m', \sigma'_m, adm, \Upsilon) \leftarrow \mathsf{SignUP3S}(\ pk_{\mathsf{UP3S}}\ , sk^{Sign}_{\mathsf{UP3S}}, m', adm, \mathbb{S}_{PSan})$ and outputs the message signature pair $(m', \sigma'_m, adm, \Upsilon)$.
- For $b = 1$, $\mathcal{O}$Sign-or-SanitUP3S runs the signing algorithm to create $(m, \sigma_m, adm, \Upsilon) \leftarrow \mathsf{SignUP3S}(pk_{\mathsf{UP3S}}, sk^{Sign}_{\mathsf{UP3S}}, m, adm, \mathbb{S}_{PSan})$, further sanitizes the message $m' \leftarrow \mathsf{MoD}(m, adm, mod)$ and returns $(m', \sigma'_m, adm, \Upsilon) \leftarrow \mathsf{SanitizeUP3S}(pk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}}, sk^{San}_{\mathsf{UP3S}}, m, \sigma_m, adm, mod, \Upsilon)$.

$\mathcal{A}$ wins if it can guess b with probability better than the random guess. Note that access to $\mathcal{O}$ProveUP3S oracle is restricted to (m, σ_m) pairs that have never been queried to $\mathcal{O}$Sign-or-SanitUP3S oracle.

Definition 3. *(Transparency)* UP3S *is transparent if for any PPT adversary* $\mathcal{A}$*,* $\left|\mathsf{Pr}[\boldsymbol{Exp}^{Transparency}_{\mathcal{A},\mathsf{UP3S}}(\lambda) = 1] - \frac{1}{2}\right| \leq \epsilon(\lambda)$*, where the transparency experiment is defined in Fig 3*

$\mathbf{Exp}^{Transparency}_{\mathcal{A},\mathsf{UP3S}}(\lambda)$

$b \xleftarrow{\$} \{0,1\}, \mathcal{T} = \{\}$

$a \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{SignUP3S}(.),\mathcal{O}\mathsf{SanitUP3S}(.),\mathcal{O}\mathsf{ProveUP3S}(.),\mathcal{O}\mathsf{Sign\text{-}or\text{-}SanitUP3S}(.,b)}(pk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}})$

if $a = b$

  return 1

else return 0

Fig. 3: UP3S transparency experiment.

Immutability. This security notion implies that no adversary with no access to the signer's secret key $sk^{Sign}_{\mathsf{UP3S}}$ can alter inadmissible blocks. In UP3S, we extend the immutability definition to capture adversarial changes in the predefined signing predicate Υ, i.e., no adversary can change the signing predicate defined by the original signer of a message. Immutability is modeled by the security experiment defined in Fig. 4 in which adversary $\mathcal{A}$ has access to $\mathcal{O}$SignUP3S, and $\mathcal{O}$SanitUP3S oracles. The signing oracle $\mathcal{O}$SignUP3S is initialized with $sk^{Sign}_{\mathsf{UP3S}}$ for the attribute set $\mathbb{S}_{Sign}$. $\mathcal{A}$ queries $\mathcal{O}$SignUP3S by $m_i, adm_i, \mathbb{S}_{PSan,i}$ for $i = 1, 2, ..., q$, the signing oracle outputs the signature tuple $(m_i, \sigma_{mi}, adm_i, \Upsilon_i)$ where the predicate Υ_i is satisfied by $\mathbb{S}'_{Sign} \subseteq \mathbb{S}_{Sign}$ and by $\mathbb{S}_{PSan,i}$. On the other hand, The sanitization oracle $\mathcal{O}$SanitUP3S is initialized with sk^{San}_{UP3S} for the attribute set $\mathbb{S}_{San}$. $\mathcal{A}$ queries $\mathcal{O}$SanitUP3S by $(m_j, \sigma_{mj}, adm_j, mod_j, \Upsilon_j)$ for $j = 1, 2, ..., p$, the sanitization oracle outputs the signature tuple $(m'_j, \sigma'_{mj}, adm_j, \Upsilon_j)$. The adversary wins if it could generate a verifiable $(\sigma^*_m, m^*, adm^*, \Upsilon^*)$ such that for all $i = 1, 2, ..., q$ (resp. $j = 1, 2, ..., p$), m^* is not valid a modification of any m_i (resp. m_j) under adm_i (resp. adm_j) where $\mathsf{CheckMoD}(m_i, adm_i) = 1$ (resp. $\mathsf{CheckMoD}(m_j, adm_j) = 1$), or m^* is a valid a modification of any m_i (resp. m_j) under adm_i (resp. adm_j) where $\mathsf{CheckMoD}(m_i, adm_i) = 1$ (resp. $\mathsf{CheckMoD}(m_j, adm_j) = 1$) and $\Upsilon^* \neq \Upsilon_i$ (resp. $\Upsilon^* \neq \Upsilon_j$). Note that $\mathcal{A}$ is allowed to query $\mathcal{O}$SanitUP3S oracle to simulate multiple sanitization cases where a sanitized message could be further sanitized by a different sanitizer. The definition considers adversaries who are valid sanitizers trying to alter inadmissible blocks thus the adversary may access some sanitization key $sk^{San,\mathcal{A}}_{\mathsf{UP3S}}$ for a predefined attribute set.

Definition 4. *(Immutability)* UP3S *is an immutable sanitizable signature scheme if for any PPT adversary* $\mathcal{A}$, $\mathsf{Pr}[\boldsymbol{Exp}^{Immutability}_{\mathcal{A},\mathsf{UP3S}}(\lambda) = 1] \leq \epsilon(\lambda)$, *where the immutability experiment is defined in Fig. 4.*

$\mathbf{Exp}^{Immutability}_{\mathcal{A},\mathsf{UP3S}}(\lambda)$

$\mathcal{M} = \mathcal{L} = \{\}$

$(m^*, \sigma^*_m, adm^*, \Upsilon^*) \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{SignUP3S}(.),\mathcal{O}\mathsf{SanitUP3S}(.)}(pk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}})$

if $\mathsf{VerifyUP3S}(pk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}}, m^*, \sigma^*_m, adm^*, \Upsilon^*)$

$(\forall\{m_i, adm_i, \Upsilon_i\} \in \mathcal{M} \wedge \forall\{m_j, adm_j, \Upsilon_j\} \in \mathcal{L})$

if $(m^* \notin \{\mathsf{MoD}(m_i, adm_i, .)|\mathsf{CheckMoD}(m_i, adm_i) = 1\} \wedge$

$m^* \notin \{\mathsf{MoD}(m_j, adm_j, .)|\mathsf{CheckMoD}(m_j, adm_j) = 1\})$

return 1

elseif $(m^* \in \{\mathsf{MoD}(m_i, adm_i, .)|\mathsf{CheckMoD}(m_i, adm_i) = 1\} \wedge \Upsilon^* \neq \Upsilon_i) \vee$

$(m^* \in \{\mathsf{MoD}(m_j, adm_j, .)|\mathsf{CheckMoD}(m_j, adm_j) = 1\} \wedge \Upsilon^* \neq \Upsilon_j))$

return 1

return 0

Fig. 4: UP3S immutability experiment.

Accountability. This security notion implies that if a signer (resp. sanitizer) did not sign (resp. sanitize) a message, then a malicious sanitizer (resp. signer) should not be able to convince the judge to accuse the signer (resp. sanitizer). Accountability is modeled by the security experiment defined in Fig. 5, in which adversary $\mathcal{A}$ has access to either sk^{San}_{UP3S} (resp. $sk^{Sign}_{\mathsf{UP3S}}$), in addition to two oracles $\mathcal{O}\mathsf{SanitUP3S}$ (resp. $\mathcal{O}\mathsf{SignUP3S}$) and $\mathcal{O}\mathsf{ProveUP3S}$. $\mathcal{A}$ can query $\mathcal{O}\mathsf{SanitUP3S}$ (resp. $\mathcal{O}\mathsf{SignUP3S}$) with $(m_i, \sigma_{mi}, adm_i, mod_i, \Upsilon_i)$ (resp. m_i) to get $(m'_i, \sigma'_{mi}, adm_i, \Upsilon_i)$ (resp. $(m_i, \sigma_{mi}, adm_i, \Upsilon_i)$) for $i = \{1, 2, ..., q\}$. The adversary wins if it outputs a verifiable message signature pair $(m^*, \sigma^*_m, adm^*, \Upsilon^*)$ where $m^* \notin \{m_1, \ldots, m_q\}$ and the output $\mathcal{O}\mathsf{ProveUP3S}$ oracle on the input of $(pk^{Sign}_{\mathsf{UP3S}}, m^*, \sigma^*_m, adm^*, \Upsilon^*)$ is falsely traced back to i_{Sign} if $\mathcal{A}$ has access to sk^{San}_{UP3S}), or to i_{San} if $\mathcal{A}$ has access to $sk^{Sign}_{\mathsf{UP3S}}$, and such a result is verified by the $\mathsf{JudgeUP3S}$ algorithm.

Definition 5. *(Accountability)* $\mathsf{UP3S}$ *ensures accountability if for any PPT adversary* $\mathcal{A}$, $\mathsf{Pr}[\boldsymbol{Exp}^{Accountability}_{\mathcal{A},\mathsf{UP3S}}(\lambda) = 1] \leq \epsilon(\lambda)$, *where the accountability experiment is defined in Fig. 5*

$\mathbf{Exp}^{Accountability}_{\mathcal{A},\mathsf{UP3S}}(\lambda)$

$\mathcal{M} = 0, \mathcal{L} = 0$
if $\mathcal{A}$ has $sk^{Sign}_{\mathsf{UP3S}}$
  $(m^*, \sigma^*_m, adm^*, \Upsilon^*) \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{SanitUP3S}(.),\mathcal{O}\mathsf{ProveUP3S}(.)}(pk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}})$
  $(i^*, \pi^*) \leftarrow \mathsf{ProveUP3S}(pk_{\mathsf{UP3S}}, sk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}}, m^*, \sigma^*_m, adm^*, \Upsilon^*)$
  if $\mathsf{VerifyUP3S}(pk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}}, m^*, \sigma^*_m, adm^*, \Upsilon^*) \wedge i^* \neq i_{Sign} \wedge (m^*, \sigma^*_m) \notin (\mathcal{M} \cup \mathcal{L}) \wedge$
    $\top \leftarrow \mathsf{JudgeUP3S}(pk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}}, m^*, \sigma^*_m, adm^*, \Upsilon^*, i^*, \pi^*)$
    return 1
  else return 0
if $\mathcal{A}$ has sk^{San}_{UP3S}
  $(m^*, \sigma^*_m, adm^*, \Upsilon^*) \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{SignUP3S}(.),\mathcal{O}\mathsf{ProveUP3S}(.)}(pk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}})$
  $(i^*, \pi^*) \leftarrow \mathsf{ProveUP3S}(pk_{\mathsf{UP3S}}, sk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}}, m^*, \sigma^*_m, adm^*, \Upsilon^*)$
  if $\mathsf{VerifyUP3S}(pk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}}, m^*, \sigma^*_m, adm^*, \Upsilon^*) \wedge i^* \neq i_{San} \wedge (m^*, \sigma^*_m) \notin (\mathcal{M} \cup \mathcal{L}) \wedge$
    $\top \leftarrow \mathsf{JudgeUP3S}(pk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}}, m^*, \sigma^*_m, adm^*, \Upsilon^*, i^*, \pi^*)$
  return 1
return 0

Fig. 5: $\mathsf{UP3S}$ accountability experiment.

Privacy. This notion implies that it is infeasible to use sanitized signatures to recover information about the sanitized parts of the message. Privacy is defined using an experiment where the adversary inputs two message-modifications tuples (m_0, adm_0, mod_0) and (m_1, adm_1, mod_1) to $\mathcal{O}\mathsf{LoRSignSanitUP3S}$ oracle which is initialized with a secret random bit 'b'. Depending on 'b', the oracle

outputs a sanitized signature of either the left or right input message modification tuple. The adversary wins if it could determine which pair is used in the sanitization process with probability better than the random guess. Similar to $\mathcal{O}$LoRSanitUP3S, the adversary must input two messages with identical fixed parts, $m_{0_{!adm}} = m_{1_{!adm}}$, the two messages' admissible policies must be the same, i.e., $adm_0 = adm_1$, and the two messages have to be signed under the same attribute-set of possible future sanitizers, $\mathbb{S}_{PSan,0} = \mathbb{S}_{PSan,1}$.

Definition 6. *(Privacy)* UP3S *scheme is private if for any PPT adversary* $\mathcal{A}$*,* $|\mathsf{Pr}[\boldsymbol{Exp}^{Privacy}_{\mathcal{A},\mathsf{UP3S}}(\lambda) = 1] - \frac{1}{2}| \leq \epsilon(\lambda)$*, where the privacy experiment is defined in Fig. 6.*

$\mathbf{Exp}^{Privacy}_{\mathcal{A},\mathsf{UP3S}}(\lambda)$

$b \xleftarrow{\$} \{0,1\}$

$a \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{SignUP3S}(.),\mathcal{O}\mathsf{SanitUP3S}(.),\mathcal{O}\mathsf{ProveUP3S}(.),\mathcal{O}\mathsf{LoRSignSanitUP3S}(.,b)}(pk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}})$

if $a = b$

  return 1

return 0

Fig. 6: UP3S privacy experiment.

Unforgeability This notion implies that an adversary with no access to either the signer or the sanitizer secret keys cannot generate a verifiable signature under honestly generated keys. This also includes the case where the adversary does not possess the required attribute set by the claim predicate to generate such signatures. This must hold even if the adversary has access to additional message signature pairs and the public keys. Unforgeability is modeled by the experiment depicted in Fig. 7 in which adversary $\mathcal{A}$ has access to three oracles $\mathcal{O}$SignUP3S, $\mathcal{O}$SanitUP3S , $\mathcal{O}$ProveUP3S and possesses a set of attributes $\mathbb{S}_{\mathcal{A}}$. $\mathcal{A}$ wins if it outputs a verifiable tuple $(m^*, \sigma^*_m, adm^*, \Upsilon^*)$ that has never been queried to $\mathcal{O}$SignUP3S nor $\mathcal{O}$SanitUP3S oracles and the claim predicate Υ^* is not satisfied by $\mathbb{S}_{\mathcal{A}}$.

Definition 7. *(Unforgeability)* UP3S *scheme is unforgeability if for any PPT adversary* $\mathcal{A}$*,* $|\mathsf{Pr}[\boldsymbol{Exp}^{Unforgeability}_{\mathcal{A},\mathsf{UP3S}}(\lambda) = 1]| \leq \epsilon(\lambda)$*, where the unforgeability experiment is defined in Fig. 7.*

4 UP3S Generic Construction

In the generic construction for UP3S, we utilize two main building blocks, a TABS scheme, and an RDS scheme. The generic construction of UP3S scheme is depicted in Fig. 8. Once UP3S is initialized, signers and sanitizers can generate their keys using KGenSignUP3S and KGenSanUP3S algorithms. To construct the

$\mathbf{Exp}^{Unforgeability}_{\mathcal{A},\mathsf{UP3S}}(\lambda)$

$\mathcal{M} = \mathcal{L} = \{\}$

$(m^*, \sigma^*_m, adm^*, \Upsilon^*) \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{SignUP3S}(.),\mathcal{O}\mathsf{SanitUP3S}(.),\mathcal{O}\mathsf{ProveUP3S}(.)}(pk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}})$

if $\mathsf{VerifyUP3S}(pk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}}, m^*, \sigma^*_m, adm^*, \Upsilon^*) \wedge (m^*, \sigma^*_m) \notin \mathcal{M} \cup \mathcal{L} \wedge \Upsilon^*(\mathbb{S}_{\mathcal{A}}) \neq 1$

  return 1

return 0

Fig. 7: UP3S unforgeability experiment.

sanitization policy for a given message, the signer uses their own selective set of attributes ($\mathbb{S}'_{Sign} \subseteq \mathbb{S}_{Sign}$), and that of possible future sanitizers $\mathbb{S}_{PSan}$ to construct a monotone access structure (predicate) Υ. The produced predicate Υ must be satisfied by some of the signer attributes ($\mathbb{S}'_{Sign}$) and should be satisfied by the selected attribute sets of future possible sanitizers $\mathbb{S}_{PSan}$ as well, i.e., ($\Upsilon(\mathbb{S}'_{Sign}) = 1$ and $\Upsilon(\mathbb{S}_{PSan}) = 1$). Thus any scheme user who holds an attribute set $\mathbb{S}''$ that satisfies the claim predicate Υ, i.e, $\Upsilon(\mathbb{S}'') = 1$, can sanitize such a message.

Signing. To sign a given message, the signer uses the SignUP3S algorithm, in which a hash function H is applied on the access structure Υ along with the inadmissible part of the message $m_{!adm}$, and the set of indices of the modifiable blocks adm. The output of H is signed using the RDS scheme with the signer key sk^{Sign}_{RDS} to output a signature σ_{fix}. Next, the full message m is anonymously signed using the TABS scheme under the signer key sk^{Sign}_{TABS} to output $(\sigma_{full}, \Upsilon)$, where σ_{full} attests that the message signer possesses a set of attributes satisfying the sanitization policy, i.e., $\Upsilon(\mathbb{S}'_{Sign}) = 1$. Finally, the signer outputs (σ_m, Υ) as the sanitizable signature over m, where $\sigma_m = (\sigma_{fix}, \sigma_{full})$.

Sanitizing. A sanitizer who holds a set of attributes ($\mathbb{S}'_{PSan} \subseteq \mathbb{S}_{PSan}$) that satisfy the message signature claim predicate i.e $\Upsilon(\mathbb{S}'_{PSan}) = 1$, is authorized to sanitize the admissible part(s) of the message m_{adm} according to adm. The sanitizer first applies the set of modification mod over m to generate the modified version of the message m' such that $m' = \mathsf{MoD}(m, , adm, mod)$. Then the sanitizer signs m' anonymously using their TABS scheme sanitizer key sk^{San}_{TABS} under the same claim predicate Υ where $\Upsilon(\mathbb{S}'_{PSan}) = 1$ to evaluate σ'_{full}. Finally, the sanitizer rerandomizes the original signature σ_{fix} to produce σ'_{fix}, and outputs (σ'_m, Υ) as the sanitized signature version, where $\sigma'_m = (\sigma'_{fix}, \sigma'_{full})$.

Verifying and tracing. Verifying a message signature pair is straightforward, where σ'_{fix} and σ'_{full} are separately verified with respect to their corresponding verification keys using the VerifyUP3S algorithm. To trace a message signature pair to its original signer, the tracing function of the underlying TABS scheme is utilized in the ProveUP3S algorithm and then the JudgeUP3S algorithm attests whether the output of ProveUP3S is valid or not.

ppGenUP3S. Given a collision-resistant hash function $H : \{0,1\}^* \rightarrow Z_p^*$, run $pp_{TABS} \leftarrow \mathsf{ppGenTABS}(1^\lambda)$, $pp_{RDS} \leftarrow \mathsf{ppGenRDS}(1^\lambda)$. Set $pp_{\mathsf{UP3S}} = \{H, pp_{TABS}, pp_{RDS}\}$, where pp_{UP3S} becomes an implicit input for all UP3S algorithms.

$$pp_{\mathsf{UP3S}} \leftarrow \mathsf{ppGenUP3S}(1^\lambda)$$

SetupUP3S. Initialize the TABS scheme trusted entities and generate their corresponding keys, $tsk_{TABS}^{TA} \leftarrow \mathsf{TAKeyGenTABS}(pp_{\mathsf{UP3S}})$ and $(pk_{TABS}, msk_{TABS}) \leftarrow \mathsf{AAKeyGenTABS}(pp_{\mathsf{UP3S}})$. Output UP3S public-secret key pair $(pk_{\mathsf{UP3S}}, sk_{\mathsf{UP3S}}) = (pk_{TABS}, (msk_{TABS}, tsk_{TABS}^{TA}))$

$$(pk_{\mathsf{UP3S}}, sk_{\mathsf{UP3S}}) \leftarrow \mathsf{SetupUP3S}(pp_{\mathsf{UP3S}})$$

KGenSignUP3S. Let $(sk_{RDS}^{Sign}, pk_{RDS}^{Sign}) \leftarrow \mathsf{keyGenRDS}(pp_{RDS})$ and $sk_{TABS}^{Sign,i} \leftarrow \mathsf{SignerKeyGenTABS}(pp_{TABS}, msk_{TABS}, i_{Sign}, \mathbb{S}_{Sign,i})$. Output $(sk_{\mathsf{UP3S}}^{Sign}, pk_{\mathsf{UP3S}}^{Sign}) = ((sk_{RDS}^{Sign}, sk_{TABS}^{Sign,i}), pk_{RDS}^{Sign})$.

$$(sk_{\mathsf{UP3S}}^{Sign}, pk_{\mathsf{UP3S}}^{Sign}) \leftarrow \mathsf{KGenSignUP3S}(sk_{\mathsf{UP3S}}, i_{Sign}, \mathbb{S}_{Sign,i})$$

KGenSanUP3S. Let $sk_{TABS}^{San,i} \leftarrow \mathsf{SignerKeyGenTABS}(pp_{TABS}, msk_{TABS}, i_{San}, \mathbb{S}_{San,i})$. Output $sk_{\mathsf{UP3S}}^{San} = sk_{TABS}^{San,i}$.

$$sk_{\mathsf{UP3S}}^{San} \leftarrow \mathsf{KGenSanUP3S}(sk_{\mathsf{UP3S}}, i_{San}, \mathbb{S}_{San,i})$$

SignUP3S. Generate the signing predicate Υ s.t. $\mathbb{S}'_{Sign,i} \subseteq \mathbb{S}_{Sign,i}$ and $\mathbb{S}_{PSan}$, where $\Upsilon(\mathbb{S}'_{Sign,i}) = 1$ and $\Upsilon(\mathbb{S}_{PSan}) = 1$. Generate $\sigma_{fix} \leftarrow \mathsf{SignRDS}(sk_{RDS}^{Sign}, H(pk_{\mathsf{UP3S}}||m_{!adm}||adm||\Upsilon))$ and $\sigma_{full} \leftarrow \mathsf{SignTABS}(pp_{TABS}, sk_{TABS}^{Sign,i}, m, \Upsilon)$. Let $\sigma_m = (\sigma_{fix}, \sigma_{full})$, return $(m, \sigma_m, adm, \Upsilon)$.

$$(m, \sigma_m, adm, \Upsilon) \leftarrow \mathsf{SignUP3S}(pk_{\mathsf{UP3S}}, sk_{\mathsf{UP3S}}^{Sign}, m, adm, \mathbb{S}_{PSan})$$

VerifyUP3S. Check if $\mathsf{Adm}(m) = 1$ and $m_{!adm} \in m$ at the correct positions, otherwise return $\perp$. Let $(\sigma_{fix}, \sigma_{full}) \leftarrow \sigma_m$, if $\mathsf{VerifyRDS}(pk_{RDS}^{Sign}, H(pk_{\mathsf{UP3S}}||m_{!adm}||adm||\Upsilon, \sigma_{fix})) \wedge \mathsf{VerifyTABS}(pp_{TABS}, pk_{TABS}, m, \sigma_{full}, \Upsilon)$ return $\top$. Otherwise, return $\perp$.

$$\{\top, \perp\} \leftarrow \mathsf{VerifyUP3S}(pk_{\mathsf{UP3S}}, pk_{\mathsf{UP3S}}^{Sign}, m, \sigma_m, adm, \Upsilon)$$

SanitizeUP3S. If $\mathsf{VerifyUP3S}(pk_{\mathsf{UP3S}}, pk_{\mathsf{UP3S}}^{Sign}, m, \sigma_m, adm, \Upsilon) = \perp \vee \mathsf{CheckMoD}(m, adm) \neq 1$ return $\perp$. Otherwise, let $(\sigma_{fix}, \sigma_{full}) \leftarrow \sigma_m$, generate $\sigma'_{fix} \leftarrow \mathsf{RerandomizeRDS}(m, \sigma_{fix})$, $m' \leftarrow \mathsf{MoD}(m, adm, mod)$, $\sigma'_{full} \leftarrow \mathsf{SignTABS}(pp_{TABS}, sk_{TABS}^{San,i}, m', \Upsilon)$. Let $\sigma'_m \leftarrow (\sigma'_{fix}, \sigma'_{full})$, return $(m', \sigma'_m, adm, \Upsilon)$.

$$(m', \sigma'_m, adm, \Upsilon) \leftarrow \mathsf{SanitizeUP3S}(pk_{\mathsf{UP3S}}, pk_{\mathsf{UP3S}}^{Sign}, sk_{\mathsf{UP3S}}^{San}, m, \sigma_m, adm, mod, \Upsilon)$$

ProveUP3S. If $\mathsf{VerifyUP3S}(pk_{\mathsf{UP3S}}, pk_{\mathsf{UP3S}}^{Sign}, m, \sigma_m, adm, \Upsilon) = \perp$ return $\perp$. Otherwise, parse σ_{full} from σ_m. Return $(i, \pi) \leftarrow \mathsf{TraceTABS}(tsk_{TABS}^{TA}, m, \sigma_{full}, \Upsilon)$.

$$\{i, \pi\} \leftarrow \mathsf{ProveUP3S}(pk_{\mathsf{UP3S}}, sk_{\mathsf{UP3S}}, pk_{\mathsf{UP3S}}^{Sign}, m, \sigma_m, adm, \Upsilon)$$

JudgeUP3S. If $\mathsf{VerifyUP3S}(pk_{\mathsf{UP3S}}, pk_{\mathsf{UP3S}}^{Sign}, m, \sigma_m, adm, \Upsilon) = \perp$, return $\perp$. Otherwise, parse σ_{full} from σ_m. Return $\{\top, \perp\} \leftarrow \mathsf{JudgeTABS}(pp_{TABS}, pk_{\mathsf{UP3S}}, m, \sigma_{full}, \Upsilon, i, \pi)$.

$$\{\top, \perp\} \leftarrow \mathsf{JudgeUP3S}(pk_{\mathsf{UP3S}}, pk_{\mathsf{UP3S}}^{Sign}, m, \sigma_m, adm, \Upsilon, i, \pi)$$

Fig. 8: UP3S generic construction.

5 UP3S Security

It has been proven in [10] that unlinkable sanitizable signature schemes are private. More precisely, Brzuska *et al.* have shown how to convert an adversary against privacy into an adversary against unlinkability. Accordingly, in what follows we prove that UP3S is unlinkable (implies private), accountable, immutable, transparent, and unforgeable sanitizable signature scheme.

Theorem 1. *Given an unlinkable RDS scheme, then the sanitizable signature scheme in Fig. 8 is unlinkable.*

Proof. In the UP3S unlinkability experiment in Fig. 2, the adversary inputs to $\mathcal{O}$LoRSanitUP3S oracle two valid signature tuples $(m_0, \sigma_{m0}, adm_0, \Upsilon_0, mod_0)$, and $(m_1, \sigma_{m1}, adm_1, \Upsilon_1, mod_1)$ where $adm_0 = adm_1$, $\mathsf{MoD}(m_0, adm_0, mod_0) = \mathsf{MoD}(m_1, adm_1, mod_1)$ and $\Upsilon_0 = \Upsilon_1$. $\mathcal{O}$LoRSanitUP3S oracle outputs $(m'_b, \sigma'_{mb}) \leftarrow \mathsf{SanitizeUP3S}(pk_{\mathsf{UP3S}}, pk^{Sign}_{\mathsf{UP3S}}, sk^{San}_{\mathsf{UP3S}}, m_b, \sigma_{mb}, adm_b, mod_b, \Upsilon_b)$ for $b \xleftarrow{\$} \{0,1\}$. Recall that $\sigma'_{mb} = (\sigma'_{fix,b}, \sigma'_{full,b})$ where $\sigma'_{fix,b}$ is a randomized version of the signer's RDS signature on $H(pk_{\mathsf{UP3S}}||m_{!adm,b}||adm_b||\Upsilon_b)$ and $\sigma'_{full,b}$ is the sanitizer's TABS signature on the modified message $m'_b = MoD(m_b, adm_b, mod_b)$. By contradiction, we let an adversary $\mathcal{A}$ be successful in $\mathbf{Exp}^{Unlinkability}_{\mathcal{A},\mathsf{UP3S}}$, we then show that we can build an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break the unlinkability of the underlying RDS scheme and win in $\mathbf{Exp}^{Unlinkability}_{\mathcal{B},RDS}$ in Fig. A.3. $\mathcal{B}$ first generates $(sk^B_{TABS}, pk^B_{TABS})$ for attribute set $\mathbb{S}_\mathcal{B} = \mathbb{U}$. To simulate $\mathcal{A}$'s oracles calls, $\mathcal{B}$ answers $\mathcal{A}$'s calls to $\mathcal{O}$SignUP3S by constructing the claim predicate Υ such that $\Upsilon(\mathbb{S}_\mathcal{B}) = 1$, calculating $H(pk_{\mathsf{UP3S}}||m_{!adm}||adm||\Upsilon)$, and passes $H(pk_{\mathsf{UP3S}}||m_{!adm}||adm||\Upsilon)$ to $\mathcal{O}$SignRDS(.) to get σ_{fix}, then signs (m) using sk^B_{TABS} to get σ_{full}. To answer $\mathcal{A}$'s calls to $\mathcal{O}$SanitizeUP3S, $\mathcal{B}$ evaluates σ'_{fix} by rerandomizing σ_{fix}, and calculates $m' = \mathsf{MoD}(m, adm, mod)$, then signs (m') using sk^B_{TABS} (where $\Upsilon(\mathbb{S}_\mathcal{B} = 1)$) to get σ'_{full}. For $\mathcal{A}$'s calls to $\mathcal{O}$ProveUP3S, $\mathcal{B}$ simply replies with its own identity for all queries where $\Upsilon(\mathbb{S}_\mathcal{B} = 1)$. When $\mathcal{A}$ inputs $(m_0, \sigma_{m0}, adm_0, \Upsilon_0, mod_0)$, and $(m_1, \sigma_{m1}, adm_1, \Upsilon_1, mod_1)$ to $\mathcal{O}$LoRSanitUP3S, $\mathcal{B}$ forwards $(H(pk_{\mathsf{UP3S}}||m_{!adm,0}||adm_0||\Upsilon_0), \sigma_{fix,0})$ and $(H(pk_{\mathsf{UP3S}}||m_{!adm,1}||adm_1||\Upsilon_1), \sigma_{fix,1})$ to $\mathcal{O}$LoRRDS to obtain the challenge $\sigma'_{fix,b}$. Then $\mathcal{B}$ evaluates $m' = m'_0 \leftarrow \mathsf{MoD}(m_0, adm_0, mod_0) = m'_1 \leftarrow \mathsf{MoD}(m_1, adm_1, mod_1)$ and then signs (m') using sk^B_{TABS} where $\Upsilon(\mathbb{S}_\mathcal{B}) = \Upsilon_0(\mathbb{S}_\mathcal{B}) = \Upsilon_1(\mathbb{S}_\mathcal{B}) = 1$ to obtain σ'_{full}. $\mathcal{B}$ returns $(m', (\sigma'_{fix,b}, \sigma'_{full}), adm, \Upsilon)$ where $adm = adm_0 = adm_1$ to $\mathcal{A}$ as the sanitizer's signature over m_b under adm_b ,and Υ_b. At the end, $\mathcal{A}$ outputs a bit 'a' which $\mathcal{B}$ relays as its answer to its $\mathcal{O}$LoRRDS oracle. Note that both messages m_0 and m_1 have the same modified message m' and since $\mathcal{B}$ signs the same m' for either m_0, or m_1, i.e., $m' = m'_0 \leftarrow \mathsf{MoD}(m_0, adm_0, mod_0) = m'_1 \leftarrow \mathsf{MoD}(m_1, adm_1, mod_1)$ from scratch using the TABS scheme to generate σ'_{full}, $\mathcal{A}$ cannot link the signature σ'_{full} to either m_0 or m_1 (since $m' \neq m_0$ and $m' \neq m_1$). Even if $\mathcal{A}$ is a successful adversary against the privacy of the underlying TABS scheme (see Fig. B.6), it could only deduce the identity of the TABS signer and\or the attributes used

in signing m' but it is not able to link the signature over m' to either m_0 or m_1. Hence, the success of $\mathcal{A}$ in $\mathbf{Exp}^{Unlinkability}_{\mathcal{A},\mathsf{UP3S}}$ implies the success of $\mathcal{B}$ in $\mathbf{Exp}^{Unlinkability}_{\mathcal{B},RDS}$.

Theorem 2. *Given a private TABS scheme, then the sanitizable signature scheme in Fig. 8 is transparent.*

Proof. The privacy of the TABS scheme ensures that the generated signature reveals no information about the signer other than the fact that the signer possesses a set of attributes that satisfies a claim predicate. Hence, such a signature hides both the original signer identity and the attributes used to satisfy the predicate Υ as well. Therefore, by contradiction, we assume that the $\mathsf{UP3S}$ scheme is not a transparent sanitizable signature scheme. We then show that the privacy of the underlying TABS scheme cannot hold. Let an adversary $\mathcal{A}$ be successful in $\mathbf{Exp}^{Transparency}_{\mathcal{A},\mathsf{UP3S}}$, we show how to build an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break the privacy of the underlying TABS scheme and win in $\mathbf{Exp}^{Privacy}_{\mathcal{B},TABS}$ in Fig. B.6. $\mathcal{B}$ simulates $\mathcal{A}$'s $\mathsf{UP3S}$ oracles calls as follows; $\mathcal{B}$ first generates the keys (sk_{RDS}, pk_{RDS}) for the RDS scheme so that it can compute σ_{fix} on $H(pk_{\mathsf{UP3S}}||m_{!adm}||adm||\Upsilon)$. $\mathcal{B}$ answers $\mathcal{A}$'s calls to $\mathcal{O}\mathsf{SignUP3S}$ by constructing the claim predicate Υ, calculating $H(pk_{\mathsf{UP3S}}||m_{!adm}||adm||\Upsilon)$, signs the output using sk_{RDS} to get σ_{fix}, then forwards (m, Υ) to $\mathcal{O}\mathsf{SignTABS}$ oracle to get σ_{full}. To answer $\mathcal{A}$'s calls to $\mathcal{O}\mathsf{SanitizeUP3S}$, $\mathcal{B}$ obtains σ'_{fix} by rerandomizing σ_{fix}, then it calculates $m' = \mathsf{MoD}(m, adm, mod)$ and forwards (m', Υ) to $\mathcal{O}\mathsf{SignTABS}$ oracle to get σ'_{full}. For $\mathcal{A}$ calls to $\mathcal{O}\mathsf{ProveUP3S}$, $\mathcal{B}$ simply forwards $(m, \sigma_{full}, \Upsilon)$ to $\mathcal{O}\mathsf{ProveTABS}$. When $\mathcal{A}$ inputs (m, mod, adm, Υ) to $\mathcal{O}\mathsf{Sign\text{-}or\text{-}SanitUP3S}$, $\mathcal{B}$ signs the message $H(pk_{\mathsf{UP3S}}||m_{!adm}||adm||\Upsilon)$ using its RDS keys thus producing σ_{fix}. Then, $\mathcal{B}$ evaluates $m' \leftarrow \mathsf{MoD}(m, adm, mod)$ and passes the message (m', Υ) to the $\mathcal{O}\mathsf{LoRSignTABS}$ oracle which responds with a challenge TABS signature σ_{full}. $\mathcal{B}$ returns $(m', (\sigma_{fix}, \sigma_{full}), adm, \Upsilon)$ to $\mathcal{A}$ as either the signer or sanitizer signature over m'. At the end, $\mathcal{A}$ outputs a bit 'a' which $\mathcal{B}$ forwards as its answer to the $\mathcal{O}\mathsf{LoRSignTABS}$ oracle.

Theorem 3. *Given an unforgeable RDS, and a collision-resistant hash function, the sanitizable signature scheme in Fig. 8 is immutable.*

Proof. Recall that for an adversary $\mathcal{A}$ against $\mathsf{UP3S}$ immutability to succeed in $\mathbf{Exp}^{Immutability}_{\mathcal{A},\mathsf{UP3S}}$, it has to output a verifiable $(m^*, \sigma^*_m, adm^*, \Upsilon^*)$ such that $m^* \notin \{\mathsf{MoD}(m_i, adm_i, .)|\mathsf{CheckMoD}(m_i, adm_i) = 1\}$ $\forall$ i queries to $\mathcal{O}Sign\mathsf{UP3S}$ and $m^* \notin \{\mathsf{MoD}(m_j, adm_j, .)|\mathsf{CheckMoD}(m_j, adm_j) = 1\}$ $\forall$ j queries to $\mathcal{O}Sanit\mathsf{UP3S}$ or $(m^* \in \{\mathsf{MoD}(m_i, adm_i, .)|\mathsf{CheckMoD}(m_i, adm_i) = 1\} \wedge \Upsilon^* \neq \Upsilon_i)$ $\forall$ i queries to $\mathcal{O}Sign\mathsf{UP3S}$ or $(m^* \in \{\mathsf{MoD}(m_j, adm_j, .)|\mathsf{CheckMoD}(m_j, adm_j) = 1\} \wedge \Upsilon^* \neq \Upsilon_j))$ $\forall$ j queries to $\mathcal{O}Sanit\mathsf{UP3S}$. Given a collision-resistant hash function H, by contradiction, we assume that the $\mathsf{UP3S}$ scheme is not immutable. We show that if we have a successful adversary $\mathcal{A}$ in $\mathbf{Exp}^{Immutability}_{\mathcal{A},\mathsf{UP3S}}$, we can build an adversary $\mathcal{B}$ that wins the unforgeability of the underlying

RDS signature scheme in $\mathbf{Exp}^{EUF-CMA}_{\mathcal{B},RDS}$ in Fig. A.2. $\mathcal{B}$ simulates $\mathcal{A}$'s environment with the help of the RDS signing oracle $\mathcal{O}\mathsf{SignRDS}$ as follows, $\mathcal{B}$ receives a public key pk^{Sign}_{RDS} from its experiment, initializes the TABS scheme, then generates a secret key of the TABS scheme $sk^{\mathcal{B}}_{TABS}$. It then passes to $\mathcal{A}$ both public keys and answers $\mathcal{A}$'s oracle queries as follows. $\mathcal{B}$ answers $\mathcal{A}$'s calls to $\mathcal{O}\mathsf{SignUP3S}$ by constructing the claim predicate Υ, calculating $H(pk_{\mathsf{UP3S}}||m_{!adm}||adm||\Upsilon)$, then passes $H(pk_{\mathsf{UP3S}}||m_{!adm}||adm||\Upsilon)$ to $\mathcal{O}\mathsf{SignRDS}$ to obtain σ_{fix}, and signs $(pk_{\mathsf{UP3S}}, m, \Upsilon)$ using $sk^{\mathcal{B}}_{TABS}$ to generate σ_{full}. To answer $\mathcal{A}$'s calls to $\mathcal{O}\mathsf{SanitizeUP3S}$, $\mathcal{B}$ obtains σ'_{fix} by rerandomizing σ_{fix}, calculates $m' = \mathsf{MoD}(m, adm, mod)$, then signs $(pk_{\mathsf{UP3S}}, m', \Upsilon)$ using its generated $sk^{\mathcal{B}}_{TABS}$ to evaluate σ'_{full}. When $\mathcal{A}$ eventually outputs $(m^*, \sigma^*_m, adm^*, \Upsilon^*)$, $\mathcal{B}$ returns to its RDS unforgeability challenger in $\mathbf{Exp}^{EUF-CMA}_{\mathcal{B},RDS}$ the message $(H(m^*_{!adm}||adm^*||\Upsilon^*))$ and the forgery attempt σ^*_{fix} which is the forged RDS signature on the output of H on the input of $(m^*_{!adm}||adm^*||\Upsilon^*)$. Note that $\mathcal{A}$ succeeds if it outputs a verifiable σ_{fix} under the original signer's public key where $m^* \notin (\{\mathsf{MoD}(m_i, adm_i, .)|\mathsf{CheckMoD}(m_i, adm_i) = 1\} \wedge m^* \notin \{\mathsf{MoD}(m_j, adm_j, .)|\mathsf{CheckMoD}(m_j, adm_j) = 1\})$ or $(m^* \in \{\mathsf{MoD}(m_i, adm_i, .)|\mathsf{CheckMoD}(m_i, adm_i) = 1\} \wedge \Upsilon^* \neq \Upsilon_i) \vee (m^* \in \{\mathsf{MoD}(m_j, adm_j, .)|\mathsf{CheckMoD}(m_j, adm_j) = 1\} \wedge \Upsilon^* \neq \Upsilon_j))$, hence $H(m^*_{!adm}||adm^*||\Upsilon^*)$ was not queried by $\mathcal{B}$ to its RDS signing oracle before in either case which implies a valid forgery by $\mathcal{B}$.

Theorem 4. *Given a non-frameable and traceable TABS scheme, then the sanitizable signature scheme in Fig. 8 achieves accountability.*

Proof. Recall that the non-frameability security property of a TABS scheme ensures that even if all authorities and users of the scheme collude, they cannot produce a signature that traces to an honest user whose secret key has not been learned by the adversary. In other words, any generated signature must be traced back to the entity that holds the secret key used in signing such a message. Moreover, the traceability security property of a TABS scheme ensures that every message signature pair generated could be traced. By contradiction, we let an adversary $\mathcal{A}$ be successful in $\mathbf{Exp}^{Accountability}_{\mathcal{A},\mathsf{UP3S}}$ and show that we can build an adversary $\mathcal{B}$ (resp. $\mathcal{B}'$) which can break the non-frameability (resp. traceability) of the underlying TABS scheme and win in $\mathbf{Exp}^{Non-frameability}_{\mathcal{B},TABS}$ in Fig. B.7 (resp. $\mathbf{Exp}^{Traceability}_{\mathcal{B}',TABS}$ in Fig. B.8). $\mathcal{B}$ simulates $\mathcal{A}$'s oracles as follows. $\mathcal{B}$ first generates keys (sk_{RDS}, pk_{RDS}) for the underlying RDS scheme so $\mathcal{B}$ can compute σ_{fix} on $H(m_{!adm}||adm||\Upsilon)$. When $\mathcal{A}$ queries $\mathcal{O}\mathsf{SignUP3S}$ with $(m_i, adm_i, \mathbb{S}_{PSan})$, $\mathcal{B}$ constructs the claim predicate Υ and uses the RDS key pairs to compute $\sigma_{fix,i}$ on $H(m_{!adm,i}, adm_i, \Upsilon_i)$ and forwards (m_i, Υ_i) to the TABS signing oracle $\mathcal{O}\mathsf{SignTABS}$ to get $\sigma_{full,i}$ on m_i and then forwards the tuple $((\sigma_{fix,i}, \sigma_{full,i}), adm_i, \Upsilon_i)$ to $\mathcal{A}$. When $\mathcal{A}$ queries $\mathcal{O}\mathsf{SanitUP3S}$ with $m_j, \sigma_{m,j}, adm_j, mod_j, \Upsilon_j$, $\mathcal{B}$ rerandomize $\sigma_{fix,j}$ to get $\sigma'_{fix,j}$, then calculates $m'_j \leftarrow \mathsf{MoD}(m_j, adm_j, mod_j)$ and forwards (m'_j, Υ_j) to the TABS signing oracle $\mathcal{O}\mathsf{SignTABS}$ to get $\sigma'_{full,j}$ on m_j and then forwards the tuple $((\sigma'_{fix,j}, \sigma'_{full,j}), adm_j, \Upsilon_j)$ to $\mathcal{A}$. For $\mathcal{O}\mathsf{ProveUP3S}$ queries by $\mathcal{A}$, $\mathcal{B}$ forwards the

queries directly to the Prove oracle of the TABS scheme $\mathcal{O}\mathsf{ProveTABS}$ and relays back the output. At the end $\mathcal{A}$ outputs a tuple $(m^*, \sigma_m^*, adm^*, \Upsilon^*)$, $\mathcal{B}$ forwards $(m^*, \sigma_{full}^*, \Upsilon^*)$ to its non-frameability challenger in $\mathbf{Exp}_{\mathcal{B},TABS}^{Non-frameability}$ experiment in Fig. B.7. On the other hand, $\mathcal{B}'$ could be constructed in a similar way to $\mathcal{B}$. However, when $\mathcal{A}$ outputs a tuple $(m^*, \sigma_m^*, adm^*, \Upsilon^*)$, $\mathcal{B}'$ forwards $(m^*, \sigma_{full}^*, \Upsilon^*)$ to its traceability challenger in $\mathbf{Exp}_{\mathcal{B}',TABS}^{Traceability}$ in Fig. B.8. Therefore, the success of $\mathcal{A}$ in $\mathbf{Exp}_{\mathcal{A},\mathsf{UP3S}}^{Accountability}$ implies the success of $\mathcal{B}$ and $\mathcal{B}'$ in $\mathbf{Exp}_{\mathcal{B},TABS}^{Non-frameability}$ and $\mathbf{Exp}_{\mathcal{B}',TABS}^{Traceability}$, respectively.

Theorem 5. *Given an unforgeable RDS scheme, an unforgeable TABS scheme, and a collision-resistant hash function, the sanitizable signature scheme in Fig. 8 is unforgeable.*

Proof. Recall that the unforgeability security property of a TABS scheme ensures that an adversary cannot generate a valid signature under a predicate where it does not possess the corresponding set of attributes that satisfy such a predicate. Moreover, the unforgeability security property of an RDS scheme ensures that it is infeasible for an adversary who does not have access to the signing keys to output a valid message signature pair. Given a collision-resistant hash function H, by contradiction, we let an adversary $\mathcal{A}$ be successful in $\mathbf{Exp}_{\mathcal{A},\mathsf{UP3S}}^{Unforgeability}(\lambda)$, then we show that we can build an adversary $\mathcal{B}$ (resp. $\mathcal{B}'$) which can break the unforgeability of the underlying TABS scheme (resp. RDS scheme) and win in $\mathbf{Exp}_{\mathcal{B},TABS}^{Unforgeability}(\lambda)$ in Fig. B.5 (resp. $\mathbf{Exp}_{\mathcal{B}',RDS}^{EUF-CMA}(\lambda)$ in Fig. A.2). $\mathcal{B}$ simulates $\mathcal{A}$'s oracles as follows. $\mathcal{B}$ first generates the keys (sk_{RDS}, pk_{RDS}) for the underlying RDS scheme. When $\mathcal{A}$ queries $\mathcal{O}\mathsf{SignUP3S}$ with $(m_i, adm_i, \mathbb{S}_{PSan})$, $\mathcal{B}$ constructs the claim predicate Υ_i and uses the RDS secret key to compute $\sigma_{fix,i}$ on $H(m_{!adm,i}, adm_i, \Upsilon_i)$ and forwards (m_i, Υ_i) to the TABS signing oracle $\mathcal{O}\mathsf{SignTABS}$ to get $\sigma_{full,i}$ on m_i and then answers $\mathcal{A}$ with the tuple $((\sigma_{fix,i}, \sigma_{full,i}), adm_i, \Upsilon_i)$. When $\mathcal{A}$ queries $\mathcal{O}\mathsf{SanitUP3S}$ with $(m_j, \sigma_{m,j}, adm_j, mod_j, \Upsilon_j)$ where $\sigma_{m,j} = (\sigma_{fix,j}, \sigma_{full,j})$, $\mathcal{B}$ rerandomize $\sigma_{fix,j}$ to get $\sigma'_{fix,j}$, then calculates $m'_j \leftarrow \mathsf{MoD}(m_j, adm_j, mod_j), \Upsilon_j)$ and forwards (m'_j, Υ_j) to the TABS signing oracle $\mathcal{O}\mathsf{SignTABS}$ to get $\sigma'_{full,j}$ on m_j and then forwards the tuple $((\sigma'_{fix,j}, \sigma'_{full,j}), adm_j, \Upsilon_j)$ to $\mathcal{A}$. For $\mathcal{O}\mathsf{ProveUP3S}$ queries by $\mathcal{A}$, $\mathcal{B}$ forwards the queries directly to the Prove oracle of the TABS scheme $\mathcal{O}\mathsf{ProveTABS}$ and relays the output back to $\mathcal{A}$. At the end of $\mathbf{Exp}_{\mathcal{A},\mathsf{UP3S}}^{Unforgeability}(\lambda)$, $\mathcal{A}$ outputs a tuple $(m^*, \sigma_m^*, adm^*, \Upsilon^*)$ where $\sigma_m^* = (\sigma_{fix}^*, \sigma_{full}^*)$, and $\mathcal{B}$ forwards $(m^*, \sigma_{full}^*, \Upsilon^*)$ to its unforgeability challenger in $\mathbf{Exp}_{\mathcal{B},TABS}^{Unforgeability}(\lambda)$. On the other hand, an RDS unforgeability adversary $\mathcal{B}'$ is constructed as follows. To simulate $\mathcal{A}$'s oracles, $\mathcal{B}'$ initializes the TABS scheme and generates the secret key $sk_{TABS}^{B'}$ for some identity $i_{B'}$ and a set of attributes $(\mathbb{S}_{B'})$ s.t $\Upsilon(\mathbb{S}_{B'}) = 1$ for any Υ, hence $\mathcal{B}'$ can compute σ_{full} on m for any predicate. When $\mathcal{A}$ queries $\mathcal{O}\mathsf{SignUP3S}$ with $(m_i, adm_i, \mathbb{S}_{PSan})$, $\mathcal{B}'$ constructs the claim predicate Υ_i such that $\Upsilon(\mathbb{S}_{B'}) = 1$ and $\Upsilon(\mathbb{S}_{PSan}) = 1$, computes $H(m_{!adm,i}, adm_i, \Upsilon_i)$ and forwards it to the RDS signing oracle $\mathcal{O}\mathsf{SignRDS}$ to

get $\sigma_{fix,i}$ on m_i. Then $\mathcal{B}'$ uses its own TABS secret key to compute $\sigma_{full,i}$ on m_i, and then forwards the tuple $((\sigma_{fix,i}, \sigma_{full,i}), adm_i, \Upsilon_i)$ to $\mathcal{A}$. When $\mathcal{A}$ queries $\mathcal{O}\mathsf{SanitUP3S}$ with $m_j, \sigma_{m,j}, adm_j, mod_j, \Upsilon_j$, $\mathcal{B}'$ rerandomize $\sigma_{fix,j}$ to get $\sigma'_{fix,j}$, then calculates $m'_j \leftarrow \mathsf{MoD}(m_j, adm_j, mod_j)$ and signs $\sigma'_{full,j}$ using $sk_{TABS}^{B',i_{B'}}$ and then forwards the tuple $((\sigma'_{fix,j}, \sigma'_{full,j}), adm_j, \Upsilon_j)$ to $\mathcal{A}$. For $\mathcal{O}\mathsf{ProveUP3S}$ queries by $\mathcal{A}$, $\mathcal{B}'$ returns its own identity and a valid proof for all queries. At the end, $\mathcal{A}$ outputs a tuple $(m^*, \sigma^*_m, adm^*, \Upsilon^*)$ where $\sigma^*_m = (\sigma^*_{fix}, \sigma^*_{full})$, and $\mathcal{B}'$ forwards (m^*, σ^*_{fix}) to its unforgeability challenger in $\mathbf{Exp}^{EUF-CMA}_{\mathcal{B}',RDS}(\lambda)$. Therefore, the success of $\mathcal{A}$ in $\mathbf{Exp}^{Unforgeability}_{\mathcal{A},\mathsf{UP3S}}$ implies the success of $\mathcal{B}$ and $\mathcal{B}'$ in $\mathbf{Exp}^{Unforgeability}_{\mathcal{B},TABS}(\lambda)$ and $\mathbf{Exp}^{EUF-CMA}_{\mathcal{B}',RDS}(\lambda)$, respectively.

6 Instantiation and Efficiency

We instantiate UP3S with Pointcheval-Sanders (PS) RDS Scheme [31] because of its short signature size and low signing and verification costs. For the TABS scheme, we utilize the DTABS scheme in [24] because in addition to providing all the security properties required by UP3S, DTABS offers minimal trust in the attribute authorities by defining a stronger definition for non-frameability, i.e., when all authorities and users collude, they can not frame an honest user. This stronger notion of non-frameability overcomes the shortcomings in standard ABS schemes where the attribute keys are generated by the attribute authority for the scheme's users (signers and sanitizers in UP3S) and hence, the attribute authority has to be fully trusted. Another advantage of using DTABS is the ability to add multiple attribute authorities to the scheme dynamically, which further supports UP3S's scalability. Both PS and DTABS are instantiated in a type-3 bilinear group setting. We use instantiation 1 of DTABS for its shorter signature size [24]. The hash function H should be chosen such that its output is mapped to $\mathbb{Z}_P^*$, thus the PS scheme is used in a single message signature setting where $m \in \mathbb{Z}_p$ to produce σ_{fix}.

Efficiency. To sign a message, the signer needs to generate a hash, an RDS signature on the output of the hash function, and a TABS signature on the whole message. To sanitize a message, the sanitizer has to modify the message, rerandomize the RDS signature and generate a TABS signature for the modified message. To verify a message signature pair, the verifier verifies both the RDS and TABS signatures. Tracing a signature to its origin requires verifying the sanitizable signature, and running the tracing algorithm of the underlying TABS scheme. Finally, to verify the output of the tracing algorithm, the judge procedure verifies both the sanitizable signature and the proof generated by the tracing algorithm of the TABS scheme. The computation and communication complexities of the instantiated UP3S are as follows. Let $l \times t$ be the size of DTAB's claim-predicate monotone span program [24]. The proposed instantiation produces a total signature (σ_{fix}, σ_{full}) size of $(27.l + 21)$ elements in $\mathbb{G}_1$ + $(22.l + 15)$ elements in $\mathbb{G}_2$ + $(t + 3)$ elements in $\mathbb{Z}_p$, where σ_{fix} is a PS signature of size 2 elements in $\mathbb{G}_1$ and requires two modular exponentiations in G_1 [31],

and σ_{full} is a DTABS signature of size $(27.l + 19)$ elements in $\mathbb{G}_1$ + $(22.l + 15)$ elements $\mathbb{G}_2$ + $(t + 3)$ elements in $\mathbb{Z}_p$ [24] and, costs approximately $(27l + 32)$ modular exponentiation in G_1 + $(38l + 34)$ modular exponentiation in G_2 to produce. Note that the aforementioned signature size and computational cost apply to both signing and sanitizing a given message. Verifying a given UP3S message signature pair costs a total of $(32l+80)$ pairing operations + 1 modular exponentiation in G_1 + 2 modular exponentiation in G_2[1]. On the other hand, to trace a signature to its origin, the tracing authority produces 2 elements in $\mathbb{G}_2$ and performs 2 modular exponentiation in $\mathbb{G}_2$ in addition to the cost of UP3S signature verification. The judge procedure performs 4 pairing operations to verify the proof of the tracing procedure.

7 Comparing UP3S to P3S

In what follows, we provide a comparison between UP3S and P3S with respect to their features and security models. The reader is referred to [35] for the formal definition of P3S and its security notions. Note that comparing the efficiency of the UP3S and P3S schemes is not possible because P3S does not provide an efficiency evaluation for its suggested instantiation. Also, both generic schemes have different building blocks and there are no standard metrics for the associated complexities of the generic building blocks, i.e., PCH and a group signature scheme in P3S compared to TABS and RDS in UP3S.

7.1 Features Comparison

We compare UP3S with P3S in terms of the roles of the scheme's entities, features of its procedures, and scalability. *Signing.* In UP3S, the signer's responsibility is limited only to signature generation and sanitization policy definition, and no interaction is needed from the signer to reveal the identity of the actual signer or sanitizer of a given message signature pair. In P3S, signers act as group managers, where they add new sanitizers to the system in addition to acting as openers for the group signature on the message. In P3S, the signer should know the identity/public key of at least one sanitizer prior to signature generation in order to be able to create the group signature using a NIZK OR proof. However, in UP3S, the signer defines a sanitization policy (signing predicate) which determines possible future sanitizers based on their attributes only, and no need to know the identity/key of any of them prior to signature generation.
Sanitizing. Unlike P3S which uses a policy-based chameleon hash as its core building block, UP3S uses a TABS scheme. Thus, it is not required to share any trapdoor information with every sanitizer before sanitizable signature generation as in the case of P3S. In P3S, Υ is only used as an input to the signing algorithm and could not be verified during the signature verification. In UP3S, Υ is an input

[1] The verification cost of DTABS could be enhanced using batch verification [7] of the underlying Groth-Sahai proof of knowledge [25]

to all its subsequent algorithms, hence any of UP3S algorithms can verify that a message signature pair is generated by a signer\sanitizer who possesses a set of attributes satisfying Υ. Furthermore, in UP3S, the sanitization rights of a given message are solely controlled by the attribute set possessed by any scheme user. Hence, UP3S neither requires a group manager role nor defines an AddSan procedure (Def. 6 in [35]) as in P3S, which is used by the group manager to grant the sanitization rights of a given message to a specific sanitizer.

Scalability. In P3S, the signature size should grow linearly with the number of group members (possible future sanitizers) which is required to achieve transparency in a linkable signature scheme. More precisely, like in group signature schemes, P3S generates a NIZK OR proof that proves that the encrypted public key (identity) of the signature generator for a given message is either the original signer OR a sanitizer. However, since P3S is linkable, assuming a given timeline for signature generation, an observer can link two signatures originating from two different sanitizers to their original message. Thus, using the description of the NIZK in construction 1 in [35] where the anonymity set is always equal to two (the signer identity is always in the set), an adversary can determine with more than the negligible probability if the second message is sanitized or not which contradicts the transparency requirement. In UP3S every message has a specific sanitization policy with no sanitizers identities included in the signature, and whatever the number of sanitizers who are authorized to sanitize a given message, the signature size is fixed per the associated sanitization policy. In P3S all system-wide parameters including secret-public key pairs are initialized from scratch for each message, i.e., a new chameleon hash instance, which may limit the system's scalability. UP3S on the other hand is based on an ABS scheme where once initialized, signers (resp. sanitizers) can sign (resp. sanitize) any message, and sanitization rights are controlled by a predicate defined by the signer only.

Table 1 summarizes the features comparison between UP3S and P3S in terms of the building blocks, if the scheme requires knowing future sanitizers or not, sanitization technique, how sanitization rights are granted, signature size, and if a group manager is needed.

7.2 Security Models Comparison

Our security definitions introduce some modifications to the definitions which are proposed in P3S to capture the roles and features of the underlying building blocks in UP3S. P3S defines nine security properties, namely unforgeability, immutability, privacy, transparency, pseudonymity, signer-accountability, sanitizer-accountability, proof-soundness, and traceability [35]. Besides unlinkability which is not offered by P3S, UP3S defines unforgeability, immutability, privacy, transparency, and accountability as its required notion of security. In what follows, we compare the definitions of the security properties of both schemes.

Unforgeability. Unlike UP3S, P3S uses the concept of groups and defines unforgeability in a way to capture the various cases that arise where groups are used, such as secret signing keys can be re-used across multiple groups and sanitization

Table 1: Comparison between UP3S and P3S.

	UP3S (this work)	P3S [35]
Building blocks	TABS and RDS	PCH and GSS
Unlinkability	yes	no
Future sanitizers	no	at least one
Sanitization technique	ABS	secret key sharing
Sanitization rights	set prior to sig. gen.	granted after sig. gen.
Signature size	fixed*	variable**
Group manager	no	yes

GSS: Group signature scheme

* Per message sanitization policy

** To achieve transparency, the signature should grow linearly with the number of group members (possible future sanitizers of a certain message)

between different groups. On the other hand, UP3S does not use groups, accordingly, the unforgeability experiment (see Fig. 7) is defined with no consideration for forgery cases associated with groups as in P3S.

Immutability. Both P3S and UP3S definitions follow the original definition in [8]. However, in UP3S's immutability experiment (see Fig. 4), we give the adversary access to the sanitization oracle to consider double sanitization cases where a sanitized message could be further sanitized by a different sanitizer who fulfils the sanitization policy.

Privacy. P3S defines a stronger notion of privacy, to capture secret key leakage and bad randomness in key generation use cases. However, since UP3S provides unlinkability and it has been proven in [10] that unlinkability implies privacy, UP3S follows the definition in [10].

Transparency. Both schemes follow Brzuska *et. al* definition of transparency [8]. However, both schemes designed the experiment with different inputs to the oracles due to the difference in the used building blocks.

Pseudonymity. P3S defines pseudonymity as the infeasibility that an adversary can decide which sanitizer actually is responsible for a given signature. P3S modeled such property by an experiment where the adversary input a message signature pair, some modifications, and two possible sanitizers' secret keys to the left-or-right sanitization oracle. The adversary wins if it can decide which sanitizer secret key is used by the left-or-right sanitization oracle (see Fig.8 in [35]). To prove the independence of pseudonymity, the authors assume that the sanitizer's identity is encoded such that it can only be recovered, if both the sanitized and the original signatures are available to the adversary. We find the latter assumption counter-intuitive to the transparency requirement because such an adversary can decide with certainty which of the signatures is freshly signed and which is sanitized. UP3S provides a stronger notion of pseudonymity since it defines unlinkability (see Theorem. 1), where such an assumption can not hold while preserving unlinkability.

Accountability. P3S uses the signer secret key to open a signature and trace it to the identity (the public key) of the signer/sanitizer of a given message. Hence, it defines two types of accountability, signer-accountability, and sanitizer-accountability. Moreover, P3S defines traceability to capture the case when the opening algorithm returns $\perp$. On the other hand, UP3S uses a separate tracing authority to trace a signature back to its actual signer and does not use the signer keys in the tracing process. Hence, UP3S defined one security property, accountability in Fig. 5, which captures the cases of signer-accountability, sanitizer-accountability, and traceability in P3S.
Proof-Soundness. P3S constructs a dynamic-group-signature-like scheme, hence, it introduces proof-soundness to resist signature hijacking in group signatures where an adversary can generate a valid NIZK for an already signed message that traces back to another user [34]. In UP3S, traceability is provided by the underlying TABS scheme, where its traceability-soundness notion (see tracing soundness in [24]) serves the same goal.

8 Conclusion

We have proposed UP3S, an unlinkable policy-based sanitizable signature scheme with a fixed signature length per sanitization policy. Our scheme does not require any interaction between sanitizers and the original signer to enable the sanitization of new messages. We have analyzed the security of UP3S and proved that it is an unlinkable, immutable, transparent, and accountable signature scheme. Moreover, we provided an instantiation of UP3S using the Pointcheval-Sanders rerandomizable signature scheme and DTABS attribute-based signature scheme and analyzed its efficiency. Finally, we compared our proposed scheme with P3S, the only policy-based sanitizable signature scheme in the literature, in terms of features, scalability, and security models.

A RDS Schemes Security

In what follows, we give the formal definitions of the security properties of RDS schemes that are required for proving the security of UP3S.
Existential Unforgeability under Chosen Message Attack (EUF-CMA). This security notion implies that given access to a signing oracle $\mathcal{O}$SignRDS (see Fig. A.1), it is hard for an adversary $\mathcal{A}$ who does not have access to the signing keys to output a valid message signature pair (m^*, σ^*_{RDS}) for which m^* was never queried to the signing oracle [31].

Definition 8. *(RDS EUF-CMA) The RDS scheme is EUF-CMA secure if the for any PPT adversary $\mathcal{A}$, $\mathsf{Pr}[\boldsymbol{Exp}^{EUF-CMA}_{\mathcal{A},RDS}(\lambda) = 1] \leq \epsilon(\lambda)$, where the RDS EUF-CMA experiment is defined in Fig. A.2.*

Unlinkability. This security notion requires that given access to oracles $\mathcal{O}$Sign(.) and $\mathcal{O}$LoRRDS(.) which are defined in Fig. A.1, the adversary $\mathcal{A}$ inputs

$\mathcal{O}\mathsf{SignRDS}(m)$

$(m, \sigma_{RDS}) \leftarrow \mathsf{SignRDS}(sk_{RDS}, m)$
$\mathcal{M} = \mathcal{M} \cup \{m, \sigma_{RDS}\}$
return (m, σ_{RDS})

$\mathcal{O}\mathsf{LoRRDS}(m_0, \sigma_{RDS,0}, m_1, \sigma_{RDS,1})$

if $\mathsf{VerifyRDS}(pk_{RDS}, m_0, \sigma_{RDS,0}) \wedge \mathsf{VerifyRDS}(pk_{RDS}, m_1, \sigma_{RDS,1})$
  $(m_b, \sigma'_{RDS,b}) \leftarrow \mathsf{RandomizeRDS}(m_b, \sigma_{RDS,b})$
  return $(\sigma'_{RDS,b})$
return 0

Fig. A.1: RDS security experiments oracles

$\mathbf{Exp}^{EUF-CMA}_{\mathcal{A},RDS}(\lambda)$

$\mathcal{M} = \{\}$
$pp_{RDS} \leftarrow \mathsf{ParGenRDS}(1^{\lambda})$
$(pk_{RDS}, sk_{RDS}) \leftarrow \mathsf{KeyGenRDS}(pp_{RDS})$
$(m^*, \sigma^*_{RDS}) \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{SignRDS}(.)}(pk_{RDS})$
if $(m^*, \sigma^*_{RDS}) \notin \mathcal{M}$
  return $\mathsf{VerifyRDS}(pk_{RDS}, m^*, \sigma^*_{RDS})$
return 0

Fig. A.2: RDS EUF-CMA experiment.

two valid message signature pairs $(m_0, \sigma_{RDS,0})$ and $(m_1, \sigma_{RDS,1})$ to $\mathcal{O}\mathsf{LoRRDS}(.)$ oracle, the oracle is initialized with a secret random bit '$b \in \{0, 1\}$'. Depending on 'b', the oracle calls $\mathsf{RandomizeRDS}$ on either the left or right input message signature pair and outputs $\sigma'_{RDS,b}$. The adversary wins if it could determine which message signature pair is used in the rerandomization process with probability better than the random guess [36]. Note that RDS unlinkability implies that no adversary can distinguish between a freshly signed message signature pair and rerandomized version of the same message as with the case if the adversary obtains two different signatures for the same message m (since RDS schemes are probabilistic schemes) by querying $\mathcal{O}\mathsf{SignRDS}$ twice with the same message m, then inputs $(m, \sigma_{RDS,0})$ and $(m, \sigma_{RDS,1})$ to $\mathcal{O}\mathsf{LoRRDS}(.)$.
Note: According to [32] the unlinkability game of the underlying RDS scheme in Fig. A.3 can only be possible if the adversary does not explicitly know the RDS signed message, hence the adversary cannot link the Challenger output to the originating message using the RDS verification algorithm. However, for $\mathsf{UP3S}$ unlinkability proof, since the adversary inputs two identical messages to $\mathcal{O}\mathsf{LoRRDS}(.)$, thus the aforementioned restriction does not apply.

Definition 9. *(RDS Unlinkability) The RDS scheme is unlinkable if for any PPT adversary $\mathcal{A}$, $|\mathsf{Pr}[\boldsymbol{Exp}^{Unlinkability}_{\mathcal{A},RDS}(\lambda) = 1] - \frac{1}{2}| \leq \epsilon(\lambda)$, where the unlinkability experiment is defined in Fig. A.3.*

$\mathbf{Exp}^{Unlinkability}_{\mathcal{A},RDS}(\lambda)$

$pp_{RDS} \leftarrow \mathsf{ParGenRDS}(1^\lambda)$
$(pk_{RDS}, sk_{RDS}) \leftarrow \mathsf{KeyGenRDS}(pp_{RDS})$
$b \xleftarrow{\$} \{0,1\}$
$a \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{SignRDS}(.),\mathcal{O}\mathsf{LoRRDS}(.,.,\mathsf{b})}(pk_{RDS})$
if $a = b$
  return 1
return 0

Fig. A.3: RDS unlinkability experiment.

B TABS Schemes Security

In what follows we give the formal definitions of the security properties of TABS schemes that are required for proving the security of UP3S.

Unforgeability. This notion requires that an adversary cannot produce a verifiable signature σ_{TABS} for a message m under a predicate Υ such that $\Upsilon(\mathbb{S}) \neq 1$ where $\mathbb{S}$ is the set of attributes that the adversary holds. In other words, an adversary cannot generate a valid signature under a predicate where they do not possess the corresponding set of attributes that satisfy such a predicate [21]. The experiment, defined in Fig. B.5, models the unforgeability security notion in which the adversary is given access to the three oracles $\mathcal{O}$KeyGenTABS, $\mathcal{O}$SignTABS, and $\mathcal{O}$ProveTABS which are defined in Fig. B.4. The adversary wins if it could generate a verifiable signature $(m^*, \sigma^*_{TABS}, \Upsilon^*)$ such that $\Upsilon^*(\mathbb{S}_{Adv}) = 0$ for all the set of attributes $\mathbb{S}_{Adv}$ queried by the adversary to $\mathcal{O}$KeyGenTABS and the pair (m^*, Υ^*) have not been queried before to $\mathcal{O}$SignTABS.

$\mathcal{O}\mathsf{KeyGenTABS}(i, \mathbb{S}_i)$

$\mathbb{S}_{Adv} = \mathbb{S}_{Adv} \cup \{i, \mathbb{S}_i\}$
$sk^{User,i}_{TABS} \leftarrow \mathsf{SignerKeyGenTABS}(pp_{TABS}, msk^{AA}_{TABS}, i, \mathbb{S}_i)$
return $sk^{User,i}_{TABS}$

$\mathcal{O}\mathsf{SignTABS}(m, \Upsilon)$

$\sigma_{TABS} \leftarrow \mathsf{SignTABS}(pp_{TABS}, sk^{User,i}_{TABS}, m, \Upsilon)$
$\mathcal{M} = \mathcal{M} \cup (m, \sigma_{TABS}, \Upsilon)$
return $(m, \sigma_{TABS}, \Upsilon)$

$\mathcal{O}\mathsf{ProveTABS}(m, \sigma_{TABS}, \Upsilon)$

  if $(m, \sigma_{TABS}, \Upsilon) \in \mathcal{M}$
    $(i, \pi) \leftarrow \mathsf{TraceTABS}(tsk^{TA}_{TABS}, m, \sigma_{TABS}, \Upsilon)$
      return (i, π)
return 0

$\mathcal{O}\mathsf{LoRSignTABS}(m, \Upsilon)$

$\sigma_{TABS} \leftarrow \mathsf{SignTABS}(pp_{TABS}, sk^{User,b}_{TABS}, m, \Upsilon)$
return $(m, \sigma_{TABS}, \Upsilon)$

Fig. B.4: TABS security experiments oracles

Definition 10. *(TABS Unforgeability) a TABS scheme is unforgeable if for any PPT adversary $\mathcal{A}$, $\mathsf{Pr}[\boldsymbol{Exp}^{Unforgeability}_{\mathcal{A},TABS}(\lambda)=1] \leq \epsilon(\lambda)$, where the unforgeability experiment is defined in Fig. B.5.*

$\mathbf{Exp}^{Unforgeability}_{\mathcal{A},TABS}(\lambda)$

$pp_{TABS} \leftarrow \mathsf{ppGenTABS}(1^{\lambda})$
$tsk^{TA}_{TABS} \leftarrow \mathsf{TAKeyGenTABS}(pp_{TABS})$
$(pk_{TABS}, msk^{AA}_{TABS}) \leftarrow \mathsf{AAKeyGenTABS}(pp_{TABS})$
$\mathcal{M} = \mathbb{S}_{Adv} = \{\}$
$(m^*, \sigma^*_{TABS}, \Upsilon^*) \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{KeyGenTABS}(.),\mathcal{O}\mathsf{SignTABS}(.),\mathcal{O}\mathsf{ProveTABS}(.)}(pk_{TABS})$
if $\mathsf{VerifyTABS}(pp_{TABS}, pk_{TABS}, m^*, \sigma^*_{TABS}, \Upsilon^*) \wedge (m^*, \sigma^*_{TABS}, \Upsilon^*) \notin \mathcal{M} \wedge$
$\quad \forall \{\mathbb{S}'\} \in \mathbb{S}_{Adv}, \Upsilon^*(\mathbb{S}') = 0$
  return 1
return 0

Fig. B.5: TABS unforgeability experiment.

Privacy. Generally speaking, TABS privacy implies that the generated signature only attests to the fact that a set of attributes possessed by a signer satisfies a predicate while hiding the identity of the signer and the set of attributes used to satisfy a such predicate. While preserving the anonymity of the signer. Privacy also implies unlinkability, where an observer cannot distinguish if two valid signatures for the same signing policy have been computed by the same signer [29]. TABS privacy is modeled by an indistinguishability experiment that is defined in Fig. B.6, in which, the adversary has access to key generation oracle $\mathcal{O}\mathsf{KeyGenTABS}$, a signing oracle $\mathcal{O}\mathsf{SignTABS}$, and proving oracle $\mathcal{O}\mathsf{ProveTABS}$ where anonymity revocation is restricted to signatures generated by $\mathcal{O}\mathsf{SignTABS}$ only, see Fig. B.4. The adversary is challenged by $\mathcal{O}\mathsf{LoRSignTABS}$ oracle, which is initialized by two signing secret signing keys $sk^{User,0}_{TABS}$ and $sk^{User,1}_{TABS}$ of two different users identities, and a random bit $b \in \{0,1\}$. Upon the input of a message m, $\mathcal{O}\mathsf{LoRSignTABS}$ outputs $(m, \sigma_{TABS}, \Upsilon)$ signed by $sk^{User,b}_{TABS}$ such that $\Upsilon(\mathbb{S}_{User,0}) = \Upsilon(\mathbb{S}_{User,1}) = 1$. The adversary wins if it could guess the bit b.

Definition 11. *(TABS Privacy) TABS scheme is private if for any PPT adversary $\mathcal{A}$, $|\mathsf{Pr}[\boldsymbol{Exp}^{privacy}_{\mathcal{A},TABS}(\lambda)=1] - \frac{1}{2}| \leq \epsilon(\lambda)$, where the privacy experiment is defined in Fig. B.6.*

Non-frameability. This property ensures that even if all authorities (AA and TA) and users in the scheme collude together dishonestly, they cannot produce a valid signature that is traced back to an honest user [24]. TABS non-frameability is modeled by the experiment defined in Fig. B.7, in which the adversary has access to both TA and AA secret keys $(tsk^{TA}_{TABS}, msk^{AA}_{TABS})$, in addition to $\mathcal{O}\mathsf{KeyGenTABS}$, $\mathcal{O}\mathsf{SignTABS}$, and $\mathcal{O}\mathsf{ProveTABS}$. The adversary wins if it outputs a verifiable $(m^*, \sigma^*_{TABS}, \Upsilon^*)$ under pk_{TABS} that has not been queried to

$\mathbf{Exp}^{Privacy}_{\mathcal{A},TABS}(\lambda)$

$pp_{TABS} \leftarrow \mathsf{ppGenTABS}(1^\lambda)$
$tsk^{TA}_{TABS} \leftarrow \mathsf{TAKeyGenTABS}(pp_{TABS})$
$(pk_{TABS}, msk^{AA}_{TABS}) \leftarrow \mathsf{AAKeyGenTABS}(pp_{TABS})$
$sk^{User,0}_{TABS} \leftarrow \mathsf{SignerKeyGenTABS}(pp_{TABS}, msk^{AA}_{TABS}, i_0, \mathbb{S}_{User,0})$
$sk^{User,1}_{TABS} \leftarrow \mathsf{SignerKeyGenTABS}(pp_{TABS}, msk^{AA}_{TABS}, i_1, \mathbb{S}_{User,1})$
$\mathcal{M} = \{\}$
$b \xleftarrow{\$} \{0,1\}$
$a \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{KeyGenTABS}(.),\mathcal{O}\mathsf{SignTABS}(.),\mathcal{O}\mathsf{LoRSignTABS}(.,b)}(pk_{TABS})$
if $a = b$
  return 1
return 0

Fig. B.6: TABS privacy experiment.

$\mathcal{O}$SignTABS and when $(m^*, \sigma^*_{TABS}, \Upsilon^*)$ is traced back to its signer, the tracing algorithm outputs an identity that has never been queried to $\mathcal{O}$KeyGenTABS. Additionally, the output of the tracing algorithm is verifiable using the JudgeTABS algorithm.

$\mathbf{Exp}^{Non-frameability}_{\mathcal{A},TABS}(\lambda)$

$pp_{TABS} \leftarrow \mathsf{ppGenTABS}(1^\lambda)$
$tsk^{TA}_{TABS} \leftarrow \mathsf{TAKeyGenTABS}(pp_{TABS})$
$(pk_{TABS}, msk^{AA}_{TABS}) \leftarrow \mathsf{AAKeyGenTABS}(pp_{TABS})$
$\mathcal{M} = \mathbb{S}_{Adv} = \{\}$
$(m^*, \sigma^*_{TABS}, \Upsilon^*) \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{KeyGenTABS}(.),\mathcal{O}\mathsf{SignTABS}(.),\mathcal{O}\mathsf{ProveTABS}(.)}(tsk^{TA}_{TABS}, pk_{TABS}, msk^{AA}_{TABS})$
if $\mathsf{VerifyTABS}(pp_{TABS}, pk_{TABS}, m^*, \sigma^*_{TABS}, \Upsilon^*)$
  $(i^*, \pi^*) \leftarrow \mathsf{TraceTABS}(tsk^{TA}_{TABS}, m^*, \sigma^*_{TABS}, \Upsilon^*)$
  if $\mathsf{JudgeTABS}(pp_{TABS}, pk_{TABS}, m^*, \sigma^*_{TABS}, \Upsilon^*, i^*, \pi^*) \wedge i^* \notin \mathbb{S}_{Adv} : \Upsilon^*(\mathbb{S}_i) = 1$
  $\wedge (m^*, \sigma^*_{TABS}, \Upsilon^*) \notin \mathcal{M}$
    return 1
return 0

Fig. B.7: TABS non-frameability experiment.

Definition 12. *(TABS Non-frameability) a TABS scheme is non-frameable if for any PPT adversary* $\mathcal{A}$, $\mathsf{Pr}[\boldsymbol{Exp}^{Non-frameability}_{\mathcal{A},TABS}(\lambda) = 1] \leq \epsilon(\lambda)$, *where the non-frameability experiment is defined in Fig. B.7.*

Traceability. TABS traceability ensures that no efficient adversary can produce a signature that cannot be traced. TABS traceability is modeled by the experiment defined in Fig. B.8, in which the adversary has access to $\mathcal{O}$KeyGenTABS,

$\mathcal{O}$SignTABS, and $\mathcal{O}$ProveTABS where identity revocation is restricted to signatures generated by $\mathcal{O}$SignTABS only. The Adversary wins if it outputs a verifiable $(m^*, \sigma^*_{TABS}, \Upsilon^*)$ under pk_{TABS}, (m^*, Υ^*) has been never queried to the signing oracle, and when $(m^*, \sigma^*_{TABS}, \Upsilon^*)$ is traced back, either the ProveTABS or JudgeTABS outputs $\perp$.

Definition 13. *(TABS Traceability) a TABS scheme is traceable if for any PPT adversary $\mathcal{A}$, $\mathsf{Pr}[\boldsymbol{Exp}^{Traceability}_{\mathcal{A},TABS}(\lambda) = 1] \leq \epsilon(\lambda)$, where the traceability experiment is defined in Fig. B.8.*

$\mathbf{Exp}^{Traceability}_{\mathcal{A},TABS}(\lambda)$

$pp_{TABS} \leftarrow \mathsf{ppGenTABS}(1^\lambda)$
$tsk^{TA}_{TABS} \leftarrow \mathsf{TAKeyGenTABS}(pp_{TABS})$
$(pk_{TABS}, msk^{AA}_{TABS}) \leftarrow \mathsf{AAKeyGenTABS}(pp_{TABS})$
$\mathcal{M} = \mathbb{S}_{Adv} = \{\}$
$(m^*, \sigma^*_{TABS}, \Upsilon^*) \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{KeyGenTABS}(.), \mathcal{O}\mathsf{SignTABS}(.), \mathcal{O}\mathsf{ProveTABS}(.)}(pk_{TABS})$
if $\mathsf{VerifyTABS}(pp_{TABS}, pk_{TABS}, m^*, \sigma^*_{TABS}, \Upsilon^*) \wedge (m^*, \sigma^*_{TABS}, \Upsilon^*) \notin \mathcal{M}$
  $(i^*, \pi^*) \leftarrow \mathsf{TraceTABS}(tsk^{TA}_{TABS}, m^*, \sigma^*_{TABS}, \Upsilon^*)$
    if $i^* = \perp \vee \mathsf{JudgeTABS}(pp_{TABS}, pk_{TABS}, m^*, \sigma^*_{TABS}, \Upsilon^*, i^*, \pi^*) = \perp$
      return 1
return 0

Fig. B.8: TABS traceability experiment.

References

1. Ateniese, G., Chou, D.H., Medeiros, B.d., Tsudik, G.: Sanitizable signatures. In: European Symposium on Research in Computer Security. pp. 159–177. Springer (2005)
2. Badertscher, C., Matt, C., Maurer, U.: Strengthening access control encryption. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 502–532. Springer (2017)
3. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In: International conference on the theory and applications of cryptographic techniques. pp. 614–629. Springer (2003)
4. Bellare, M., Shi, H., Zhang, C.: Foundations of group signatures: The case of dynamic groups. In: Cryptographers' Track at the RSA Conference. pp. 136–153. Springer (2005)
5. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: 2007 IEEE symposium on security and privacy (SP'07). pp. 321–334. IEEE (2007)

6. Bilzhause, A., Pöhls, H.C., Samelin, K.: Position paper: the past, present, and future of sanitizable and redactable signatures. In: Proceedings of the 12th International Conference on Availability, Reliability and Security. pp. 1–9 (2017)
7. Blazy, O., Fuchsbauer, G., Izabachene, M., Jambert, A., Sibert, H., Vergnaud, D.: Batch groth–sahai. In: International Conference on Applied Cryptography and Network Security. pp. 218–235. Springer (2010)
8. Brzuska, C., Fischlin, M., Freudenreich, T., Lehmann, A., Page, M., Schelbert, J., Schröder, D., Volk, F.: Security of sanitizable signatures revisited. In: International Workshop on Public Key Cryptography. pp. 317–336. Springer (2009)
9. Brzuska, C., Fischlin, M., Lehmann, A., Schröder, D.: Sanitizable signatures: How to partially delegate control for authenticated data. BIOSIG 2009: biometrics and electronic signatures (2009)
10. Brzuska, C., Fischlin, M., Lehmann, A., Schröder, D.: Unlinkability of sanitizable signatures. In: International Workshop on Public Key Cryptography. pp. 444–461. Springer (2010)
11. Brzuska, C., Pöhls, H.C., Samelin, K.: Non-interactive public accountability for sanitizable signatures. In: European Public Key Infrastructure Workshop. pp. 178–193. Springer (2012)
12. Brzuska, C., Pöhls, H.C., Samelin, K.: Efficient and perfectly unlinkable sanitizable signatures without group signatures. In: European Public Key Infrastructure Workshop. pp. 12–30. Springer (2013)
13. Bultel, X., Lafourcade, P., Lai, R.W., Malavolta, G., Schröder, D., Thyagarajan, S.A.K.: Efficient invisible and unlinkable sanitizable signatures. In: IACR International Workshop on Public Key Cryptography. pp. 159–189. Springer (2019)
14. Camenisch, J., Derler, D., Krenn, S., Pöhls, H.C., Samelin, K., Slamanig, D.: Chameleon-hashes with ephemeral trapdoors. In: IACR International Workshop on Public Key Cryptography. pp. 152–182. Springer (2017)
15. Canard, S., Jambert, A.: On extended sanitizable signature schemes. In: Cryptographers' Track at the RSA Conference. pp. 179–194. Springer (2010)
16. Canard, S., Jambert, A., Lescuyer, R.: Sanitizable signatures with several signers and sanitizers. In: International Conference on Cryptology in Africa. pp. 35–52. Springer (2012)
17. Canard, S., Laguillaumie, F., Milhau, M.: Trapdoor sanitizable signatures and their application to content protection. In: International Conference on Applied Cryptography and Network Security. pp. 258–276. Springer (2008)
18. Damgård, I., Haagh, H., Orlandi, C.: Access control encryption: Enforcing information flow with cryptography. In: Theory of Cryptography Conference. pp. 547–576. Springer (2016)
19. Derler, D., Samelin, K., Slamanig, D., Striecks, C.: Fine-grained and controlled rewriting in blockchains: Chameleon-hashing gone attribute-based. Cryptology ePrint Archive (2019)
20. Derler, D., Slamanig, D.: Rethinking privacy for extended sanitizable signatures and a black-box construction of strongly private schemes. In: International Conference on Provable Security. pp. 455–474. Springer (2015)
21. Escala, A., Herranz, J., Morillo, P.: Revocable attribute-based signatures with adaptive security in the standard model. In: Nitaj, A., Pointcheval, D. (eds.) Progress in Cryptology – AFRICACRYPT 2011. pp. 224–241. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
22. Escala, A., Herranz, J., Morillo, P.: Revocable attribute-based signatures with adaptive security in the standard model. In: International conference on cryptology in Africa. pp. 224–241. Springer (2011)

23. Fleischhacker, N., Krupp, J., Malavolta, G., Schneider, J., Schröder, D., Simkin, M.: Efficient unlinkable sanitizable signatures from signatures with re-randomizable keys. In: Public-Key Cryptography–PKC 2016, pp. 301–330. Springer (2016)
24. Ghadafi, E.: Stronger security notions for decentralized traceable attribute-based signatures and more efficient constructions. In: Cryptographers' Track at the RSA Conference. pp. 391–409. Springer (2015)
25. Groth, J., Sahai, A.: Efficient noninteractive proof systems for bilinear groups. SIAM Journal on Computing **41**(5), 1193–1232 (2012)
26. Lai, J., Ding, X., Wu, Y.: Accountable trapdoor sanitizable signatures. In: International Conference on Information Security Practice and Experience. pp. 117–131. Springer (2013)
27. Lai, R.W., Zhang, T., Chow, S.S., Schröder, D.: Efficient sanitizable signatures without random oracles. In: European Symposium on Research in Computer Security. pp. 363–380. Springer (2016)
28. Maji, H., Prabhakaran, M., Rosulek, M.: Attribute-based signatures: Achieving attribute-privacy and collusion-resistance. Cryptology ePrint Archive, Report 2008/328 (2008), https://ia.cr/2008/328
29. Maji, H.K., Prabhakaran, M., Rosulek, M.: Attribute-based signatures. In: Cryptographers' track at the RSA conference. pp. 376–392. Springer (2011)
30. Miyazaki, K., Iwamura, M., Matsumoto, T., Sasaki, R., Yoshiura, H., Tezuka, S., Imai, H.: Digitally signed document sanitizing scheme with disclosure condition control. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **88**(1), 239–246 (2005)
31. Pointcheval, D., Sanders, O.: Short randomizable signatures. In: Cryptographers' Track at the RSA Conference. pp. 111–126. Springer (2016)
32. Pointcheval, D., Sanders, O.: Reassessing security of randomizable signatures. In: Cryptographers' Track at the RSA Conference. pp. 319–338. Springer (2018)
33. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Annual international conference on the theory and applications of cryptographic techniques. pp. 457–473. Springer (2005)
34. Sakai, Y., Schuldt, J.C., Emura, K., Hanaoka, G., Ohta, K.: On the security of dynamic group signatures: Preventing signature hijacking. In: International Workshop on Public Key Cryptography. pp. 715–732. Springer (2012)
35. Samelin, K., Slamanig, D.: Policy-based sanitizable signatures. In: Jarecki, S. (ed.) Topics in Cryptology – CT-RSA 2020. pp. 538–563. Springer International Publishing, Cham (2020)
36. Zhou, S., Lin, D.: Unlinkable randomizable signature and its application in group signature. vol. 2007, p. 213 (08 2007). https://doi.org/10.1007/978-3-540-79499-8˙26