Traceable Policy-Based Signatures with Delegation

Ismail Afia and Riham AlTawy

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

Abstract. In PKC 2014, a policy-based signature (PBS) scheme was proposed by Bellare and Fuchsbauer in which a signer can only sign messages conforming to some policy specified by an issuing authority. PBS construction supports the delegation of signing policy keys with possible restrictions to the original policy. Although the PBS scheme is meant to restrict the signing privileges of the scheme's users, singers could easily share their signing keys with others without being held accountable since PBS does not have a tracing capability, and a signing policy key defines a policy that should be satisfied by the message only. In this work, we build on PBS and propose a traceable policy-based signature scheme (TPBS) where we employ a rerandomizable signature scheme, a digital signature scheme, and a zero-knowledge proof system as its building blocks. TPBS introduces the notion of anonymized identity keys that are used with the policy keys for signing. Thus it achieves traceability without compromising the delegatability feature of the PBS scheme. Additionally, TPBS ensures non-frameability under the assumption of a corrupted tracing authority. We define and formally prove the security notions of the generic TPBS scheme. Finally, we propose an instantiation of TPBS utilizing the Pointcheval-Sanders rerandomizable signature scheme, Abe *et al.*'s structure-preserving signature scheme, and Groth-Sahai NIZK system, and analyze its efficiency.

Keywords: policy-based signatures · attribute-based signatures · rerandomizable signatures · group signatures.

1 Introduction

In policy-based signature (PBS) schemes, a signer can produce a valid signature of a message only if the message satisfies a specific policy [3]. PBS schemes allow an issuer to delegate signing rights to specific signers under a particular policy (by sharing a signing policy key). Yet, the produced signature is verifiable under the issuer's public key. Besides unforgeability, the standard security notion for signature schemes, the privacy of the PBS scheme ensures that signatures do not reveal the policy under which they were created. Generally speaking, PBS schemes aim to extend the functionality of digital signature schemes by offering some form of delegation of signing rights under the issuer's policy signing key. Although there exist some primitives that offer signing rights delegation, such as group signatures (GS) [4] and attribute-based signatures (ABS) [15], PBS introduces some distinct features that other primitives do not fulfill. For instance, in GS schemes, a member signs any message on behalf of the whole group. However, PBS schemes give the issuer fine-grained control over who is allowed to sign which messages. On the other hand, in ABS schemes, the produced signature attests to a specific claim predicate (policy) regarding certified attributes that the signer possesses. In other words, users can only generate signatures for policies satisfied by their attributes. Also, the policy itself is always public in ABS schemes which may be used to deduce some information about the internal structure of the signer's organization. ABS schemes do not impose any restrictions on the messages to be signed as long as the signer possesses the attributes that satisfy the claim predicate. PBS fulfills these gaps by hiding the policy under which the signature is created and requiring that the signed message conforms to the hidden policy.

Bellare and Fuchsbauer show that the PBS framework allows delegation, where a signer holding a key for some policy can delegate such a key to another signer with possible restrictions on the associated policy. Delegation enables the signing of messages that satisfy both the original and restricted policies. The holder of the newly generated key can further delegate such a key to another signer with possible further restrictions and so on. Such a delegatable PBS scheme suites applications in hierarchical settings; for instance, if an issuer of a certain company granted one of the managers the signing rights of contracts with clients X, Y, and Z, such a manager can delegate these signing rights to a team leader in his unit. Furthermore, the manager may restrict such rights and limit the team leader to signing contracts with client Z only.

The standard security requirements of PBS schemes are unforgeability and privacy [3]. Unforgeability ensures that an adversary cannot create a valid signature without having a policy key where the signed message conforms to such a policy. Privacy guarantees that a signature does not reveal the policy associated with the key. Privacy also implies unlinkability, where an adversary cannot decide whether two signatures were created using the same policy key. Although the PBS privacy definition ensures full signer anonymity, it permits key misuse without accountability. For instance, a signer of a given message may deny their responsibility for such a signature. Furthermore, policy key holders may share their keys with anyone which authorizes them to sign messages in the issuer's name without any liability.

In an attempt to tackle the aforementioned problem, Zu *et al.* have proposed a traceable policy-based signature scheme [18]. In their proposal, the user's identity is attached to the policy. More precisely, the issuer generates signing keys for the user ensuring that the user's identity is part of the key, i.e. generating the signing keys for $id||p$ where id denotes the user identity and p denotes the policy under which the signer is allowed to sign a specific message. To sign a message, the signer first encrypts his identity under the public key of an opener and provides a Non-Interactive Zero Knowledge (NIZK) proof of the issuer signature on $id||p$ such that p permits the message and id has been correctly encrypted to the given ciphertext. The generated signature contains the ciphertext in addition to the resulting NIZK proof. To trace a message to its original signers, the opener decrypts the ciphertext using its decryption key to reveal the signer's identity. Although Zu *et al.*'s proposal provides traceability, it does not protect against frameability. Since the issuer generates the signing keys of the scheme users, the issuer can frame an honest signer by using such a user's signing keys for a specific policy.

Moreover, attaching user identities to the policy seems counter-intuitive to the original goal of PBS schemes, where the signing rights are granted to users who have access to a policy key which allows them to sign messages that conform to the policy. However, in the Zu *et al.* scheme, the issuer has to issue multiple signing keys to each scheme user to include their identities for the same policy. A direct consequence of the latter is that the proposed scheme in [18] does not support policy key delegation because the policy is tied to a specific identity. Finally, the security model Zu *et al.* present does not consider traceability or non-framebility.

Our Contributions. We propose a Traceable Policy-Based Signature (TPBS) scheme that supports delegation. TPBS extends the functionality of the original PBS scheme by adding a tracing mechanism to enforce accountability and deter the abuse of signing keys. We design TPBS where the generated signature of a given message does not reveal the policy nor the identity used in the signing process. The user's signing key in TPBS consists of an identity key and a policy key which are generated independently; thus, TPBS supports policy key delegation similar to the PBS scheme. In TPBS, each user generates a secret key which is used in an interactive protocol with the TA to generate the user's identity key. However, the user's secret key is never exchanged with the TA preventing a misbehaving tracing authority or any party intercepting the user's communication with the TA from framing such a user. We formally define the extractability, simulatability, non-framability, and traceability security notions for TPBS. Moreover, we propose a generic construction for TPBS employing a rerandomizable digital signature (RDS) scheme and a simulation-sound extractable non-interactive zero-knowledge (SE-NIZK) proof system. Then we prove that the generic construction achieves the defined security notions. Finally, we give a concrete instantiation for TPBS with Pointcheval-Sanders rerandomizable signature scheme and Groth-Sahai zero-knowledge proof system and analyze its efficiency.

2 Preliminaries and Building Blocks

Sampling x uniformly at random from $\mathbb{Z}_q$ is denoted by $x \xleftarrow{\$} \mathbb{Z}_p$. We denote by i an identity from the identity universe $\mathbb{I}$, $i \in \mathbb{I}$. Let $\lambda \in \mathbb{N}$ denotes our security parameter, then a function $\epsilon(\lambda) : \mathbb{N} \to [0, 1]$ 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 $\lambda > \lambda_c$. We use $f(.)$ to denote a one-way function, and we use $\mathsf{PoK}(x : C = f(x))$ to denote an interactive perfect zero-knowledge proof of knowledge of x such that $C = f(x)$ [10].

2.1 Rerandomizable Digital Signature Scheme (RDS)

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 [19, 16, 17, 9]. 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 [16].

An RDS scheme is a tuple of five polynomial-time algorithms, $\mathsf{RDS} = \{\mathsf{ppGenRDS}, \mathsf{KeyGenRDS}, \mathsf{SignRDS}, \mathsf{RandomizeRDS}, \mathsf{VerifyRDS}\}$ which are defined as follows.

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

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

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

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

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

Some RDS schemes include a $\sigma_{RDS} \leftarrow \mathsf{SignComRDS}(sk_{RDS}, C)$ procedure that enables the signing of a commitment C of a hidden message m such that the resulting σ_{RDS} is verifiable for m [8, 16]. RDS schemes ensure 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 [16, 19] and in Appendix A.1.

2.2 Simulation-Extractable NIZK (SE-NIZK)

A SE-NIZK system enables a prover with a witness w to prove non-interactively the truthfulness of a statement x to a verifier without conveying why [12]. For x in an $\mathbb{NP}$-language $\mathcal{L}$ such that (x,w) in a relation $\mathbb{R}$ associated with $\mathcal{L}$, a SE-NIZK is a tuple of six polynomial-time algorithms, $\mathsf{NIZK} = \{\mathsf{SetupNIZK}, \mathsf{SimSetupNIZK}, \mathsf{ProveNIZK}, \mathsf{SimProveNIZK}, \mathsf{VerifyNIZK}, \mathsf{ExtrNIZK}\}$, which are defined as follows.

- $\mathsf{SetupNIZK}$. This algorithm outputs the common reference string of the system, $crs \leftarrow \mathsf{SetupNIZK}(1^\lambda)$.

- $\mathsf{SimSetupNIZK}$. This algorithm outputs a simulated crs and a trapdoor, $(crs, tr_{NIZK}) \leftarrow \mathsf{SimSetupNIZK}(1^\lambda)$.

-$\mathsf{ProveNIZK}$. On input of crs, $x \in \{0,1\}^*$ and $w \in \{0,1\}^*$ for the relation $\mathbb{R}$, this algorithm outputs a proof π_{NIZK}, $\pi_{NIZK} \leftarrow \mathsf{ProveNIZK}(crs, x, w)$.

- $\mathsf{SimProveNIZK}$. On input of crs, $x \in \{0,1\}^*$ and the trapdoor tr_{NIZK}, this algorithm outputs a proof π_{NIZK}, $\pi_{NIZK} \leftarrow \mathsf{SimProveNIZK}(crs, x, tr_{NIZK})$.

- $\mathsf{VerifyNIZK}$. This algorithm verifies the proof π_{NIZK} on x. $\{\top, \bot\} \leftarrow \mathsf{VerifyNIZK}(crs, x, \pi_{NIZK})$.

- $\mathsf{ExtrNIZK}$. This procedure extracts w from a π_{NIZK} using tr_{NIZK}, $w \leftarrow \mathsf{ExtrNIZK}(crs, x, \pi_{NIZK})$.

For the security of TPBS, we require that the SE-NIZK scheme ensures zero-knowledge and simulation-extractability. The formal definition of such security notions and their associated experiments are given in Appendix A.2.

2.3 Digital Signature Schemes

A digital signature scheme is a tuple of six polynomial-time algorithms, $\mathsf{Sig} = \{\mathsf{ppGenSig}, \mathsf{KeyGenSig}, \mathsf{SignSig}, \mathsf{VerifySig}\}$, which are defined as follows.

- $\mathsf{ppGenSig}$. This algorithm outputs the system's public parameters, $pp_{Sig} \leftarrow \mathsf{ppGenSig}(1^\lambda)$ which becomes an implicit input for all other algorithms.

- $\mathsf{KeyGenSig}$. This algorithm outputs the signer public secret key pair, $(pk_{Sig}, sk_{Sig}) \leftarrow \mathsf{KeyGenSig}(pp_{Sig})$.

- $\mathsf{SignSig}$. On input of a message m, this algorithm outputs a signature over m using sk_{Sig}, $\sigma_{Sig} \leftarrow \mathsf{SignSig}(sk_{Sig}, m)$.

- $\mathsf{VerifySig}$. This algorithm verifies σ_{Sig} over m using pk_{Sig}, $\{\top, \bot\} \leftarrow \mathsf{VerifySig}(pk_{Sig}, m, \sigma_{Sig})$.

The digital signature scheme ensures EUF-CMA as its required security notion. The formal definition of such a security notion and its associated experiment are given in Appendix A.3.

2.4 Policy-Based Signatures (PBS)

Bellare and Fuchsbauer have proposed a policy-based signature (PBS) in which a signer can only sign messages that comply with some issuer-specified policy p [3]. To achieve this, Bellare and Fuchsbauer have defined a policy checker PC as follows.

Definition 1. ***policy checker (*PC*) [3].*** PC *is an NP-relation* $\mathsf{PC} : \{0,1\}^*\{0,1\}^* \leftarrow \{0,1\}$*, where the first input is a pair* (p, m) *representing a policy* $p \in \{0,1\}^*$ *and a message* $m \in \{0,1\}^*$*, while the second input is a witness* $w_p \in \{0,1\}^*$*. the signing of* m *is permitted under policy* p *if* (p, m, w_p) *is* PC*-valid such that* $\mathsf{PC}((p, m), w_p) = 1$.

Bellare and Fuchsbauer have introduced an efficient construction utilizing Simulation-Extractable (SE) NIZK proof and an unforgeable digital signature scheme under chosen-message attacks (UF-CMA) such that the user's key is simply a signature from the issuer on the policy p using the UF-CMA signature scheme. To sign a message, the user generates a zero-knowledge proof π that the user holds an issuer-verifiable signature on the hidden policy p such that $\mathsf{PC}((p, m), w) = 1$. We give the black box construction of PBS and its SE-NIZK construction in Appendix C.

PBS scheme ensures both extractability and simulatability as stronger security notions for unforgeability and privacy, respectively. The formal definition of both security notions and their associated experiments are given in [3].

Delegatable PBS. PBS allows signing rights delegation, in which a signer who holds a key for some policy can delegate such a key to another signer with possible restriction of the associated policy. To achieve this, keys are associated with a vector of policies $(p_1, ..., p_n)$ such that a signed message m satisfies all such policies, i.e., $\mathsf{PC}((p_i, m), w_{pi}) = 1 \ \forall i \in [n]$. In such a construction, the signature scheme used by the issuer to sign a policy vector $\mathbf{p}$ is replaced by an append-only signature scheme [14]. Accordingly, the KeyGenPBS algorithm is replaced by a new algorithm DelegatePBS that issues a signing key for a new policy vector (see Appendix C).

3 Traceable Policy-Based Signatures (TPBS)

We present a Traceable Policy-Based Signatures (TPBS) scheme and keep the new scheme's definitions in line with the original work in [3]. The main idea of our scheme is that in addition to the PBS issuer's policy key, we require the use of an identity key for signing a message that satisfies the policy defined by the issuer in the policy key. Hence, we introduce a Tracing Authority (TA) where every scheme user registers with to generate an identity key. The user then uses the identity key in addition to the policy key to sign a message that conforms to the policy set by the issuer. The produced signature allows the TA to trace it to the registration information acquired from the user during identity key generation. Note that contrary to the issuer's policy key, which could be shared among users allowed by the issuer to sign a specific message, the identity key is generated by individual users and is not shared with any other entity in the system. In what follows, we give the black box definitions of the proposed construction. TPBS is a tuple of ten polynomial-time algorithms, TPBS = {ppGen, TASetup, IssuerSetup, UserKeyGen, IDKeyGen, PolicyKeyGen, Sign, Verify, Trace, Judge} which are defined as follows.

- ppGen. This algorithm outputs the public parameters of the scheme, which become an implicit input to all the other algorithms.

$$pp_{\mathsf{TPBS}} \leftarrow \mathsf{ppGen}(1^\lambda)$$

- TASetup. This algorithm generates the TA's public secret key pair $(pk^{TA}_{\mathsf{TPBS}}, sk^{TA}_{\mathsf{TPBS}})$, and initializes an empty registry Reg which the TA solely controls.

$$(pk^{TA}_{\mathsf{TPBS}}, sk^{TA}_{\mathsf{TPBS}}, Reg) \leftarrow \mathsf{TASetup}(pp_{\mathsf{TPBS}})$$

- IssuerSetup. This algorithm generates the issuer's public key secret key pair.

$$(pk^{Issuer}_{\mathsf{TPBS}}, sk^{Issuer}_{\mathsf{TPBS}}) \leftarrow \mathsf{IssuerSetup}(pp_{\mathsf{TPBS}})$$

- UserKeyGen. For user identity $i \in \mathbb{I}$, this algorithm generates the user's secret public key pair (sk_i, pk_i). We assume that pk_i is authentically associated with i in a public registry $\mathcal{D}$ such that $\mathcal{D}[i] = pk_i$, a PKI system may be used for such a purpose. Moreover, this algorithm outputs the registration information ID_i generated from sk_i using a one-way function.

$$(pk_i, sk_i, ID_i) \leftarrow \mathsf{UserKeyGen}(pp_{\mathsf{TPBS}}, i)$$

- $\mathsf{IDKeyGen}$. This two-party interactive procedure runs between a scheme user and the TA to generate the user's identity key. The inputs of the user's routine are $(i, (sk_i))$, and the inputs to the TA's routine are $((sk^{TA}_{\mathsf{TPBS}}), i, ID_i)$, where i and ID_i are sent to the TA by the user. At the end of the interaction, the user obtains the TA's signature σ^i_{ID} over their hidden secret sk_i. Finally, the user sets $sk^i_{\mathsf{TPBS}} = (sk_i, \sigma^i_{ID})$ whereas the TA obtains some registration information $Reg[i] = ID_i$.

$$((Reg[i]), (sk^i_{\mathsf{TPBS}})) \leftarrow \mathsf{IDKeyGen}((sk^{TA}_{\mathsf{TPBS}}) \xleftrightarrow[\sigma^i_{ID}]{(i, ID_i)} (sk_i)),$$

where the first (resp. second) (.) in the input and output of $\mathsf{IDKeyGen}$ contains values that are only known to the TA (resp. user).

- $\mathsf{PolicyKeyGen}$. The issuer runs this procedure to generate a secret key for a specific policy $p \in \{0,1\}^*$.

$$sk^p_{\mathsf{TPBS}} \leftarrow \mathsf{PolicyKeyGen}(sk^{Issuer}_{\mathsf{TPBS}}, p)$$

- Sign. On input of a message m, a witness $w_p \in \{0,1\}^*$ that m conforms to a specific policy p, the secret signing key sk^p_{TPBS}, the user identity key sk^i_{TPBS} , this procedure generates a signature σ_m.

$$\sigma_m \leftarrow \mathsf{Sign}(pk^{TA}_{\mathsf{TPBS}}, pk^{Issuer}_{\mathsf{TPBS}}, sk^p_{\mathsf{TPBS}}, sk^i_{\mathsf{TPBS}}, m, p, w_p)$$

- Verify. This algorithm verifies the signature σ_m over m using the issuer's and TA's public keys.

$$\{\top, \bot\} \leftarrow \mathsf{Verify}(pk^{TA}_{\mathsf{TPBS}}, pk^{Issuer}_{\mathsf{TPBS}}, m, \sigma_m)$$

- Trace. This algorithm is run by the TA to trace a signature σ_m over m to its original signer and returns the signer identity along with proof confirming such a claim.

$$(i, \pi_{Trace}) \leftarrow \mathsf{Trace}(pk^{TA}_{\mathsf{TPBS}}, pk^{Issuer}_{\mathsf{TPBS}}, Reg, m, \sigma_m)$$

- Judge. This algorithm verifies the output of the tracing algorithm.

$$\{\top, \bot\} \leftarrow \mathsf{Judge}(pk^{TA}_{\mathsf{TPBS}}, pk^{Issuer}_{\mathsf{TPBS}}, m, \sigma_m, i, \pi_{Trace})$$

TPBS **Correctness** for the correctness of TPBS, we require that for all $\lambda \in \mathbb{N}$, all $pp_{\mathsf{TPBS}} \leftarrow \mathsf{ppGen}\ (1^\lambda)$, for all $(pk^{TA}_{\mathsf{TPBS}}, (sk^{TA}_{\mathsf{TPBS}}, Reg))$ $\leftarrow \mathsf{TASetup}(pp_{\mathsf{TPBS}})$, for all $(pk^{Issuer}_{\mathsf{TPBS}}, sk^{Issuer}_{\mathsf{TPBS}}) \leftarrow \mathsf{IssuerSetup}\ (pp_{\mathsf{TPBS}})$, for all $i \in \mathbb{I}$, for all $(pk_i, sk_i, ID_i) \leftarrow \mathsf{UserKeyGen}\ (pp_{\mathsf{TPBS}})$, for all $((Reg[i]),(sk^i_{\mathsf{TPBS}})) \leftarrow \mathsf{IDKeyGen}\ ((sk^{TA}_{\mathsf{TPBS}}) \xleftrightarrow[\sigma^i_{ID}]{(i, ID_i)} (sk_i))$, for all sk^p_{TPBS} $\leftarrow \mathsf{PolicyKeyGen}\ (sk^{Issuer}_{\mathsf{TPBS}}, p)$, and for all $(m, p, w_p) \in \{0,1\}^*$ s.t $\mathsf{PC}((p, m), w_p) = 1$, we have $\sigma_m \leftarrow \mathsf{Sign}\ (pk^{TA}_{\mathsf{TPBS}}, pk^{Issuer}_{\mathsf{TPBS}}, sk^p_{\mathsf{TPBS}}, sk^i_{\mathsf{TPBS}}, m, p, w_p)$ such that $\top \leftarrow \mathsf{Verify}\ (pk^{TA}_{\mathsf{TPBS}}, pk^{Issuer}_{\mathsf{TPBS}}, m, \sigma_m)$. Moreover, we have

$(i, \pi_{Trace}) \leftarrow \mathsf{Trace}\ (pk^{TA}_{\mathsf{TPBS}}, pk^{Issuer}_{\mathsf{TPBS}}, Reg, m, \sigma_m)$ such that $\top \leftarrow \mathsf{Judge}\ (pk^{TA}_{\mathsf{TPBS}}, pk^{Issuer}_{\mathsf{TPBS}}, m, \sigma_m, i, \pi_{Trace})$.

To prevent a misbehaving TA or any party who has access to the policy key $sk^{p}_{\mathsf{TPBS}}$from framing a user, we ensure that sk^{i}_{TPBS} contains sk_i which is generated by individual users and not shared with any entity in the scheme. Moreover, Since our scheme segregates the identity keys from the policy keys, the delegatability of policy keys becomes a natural extension for our scheme and could be achieved seamlessly by applying the same technique of Bellare and Fuchsbauer [3]. Moreover, segregating the issuer and TA rules make our scheme a perfect fit for decentralized environments where multiple issuers may coexist.

3.1 TPBS Security Definitions

The security notions of PBS are privacy (policy-indistinguishability) and unforgeability [3]. Privacy of the policy ensures that a signature reveals neither the policy associated with the policy key nor the witness that was used in creating such a signature. Unforgeability is defined as the infeasibility of creating a valid signature for a message m without holding a policy key for some policy p and a witness w_p such that $\mathsf{PC}((p, m), w_p) = 1$. In the same context, Bellare and Fuchsbauer have defined simulatability and extractability as stronger versions of the aforementioned security notions [3]. The main reason behind introducing such stronger notions is that the traditional notions of policy privacy and unforgeability are insufficient for all applications. For instance, a PBS scheme with a policy checker PC such that for every message m, there is only one policy p where $\mathsf{PC}((p, m_i), w_i) = 1$ for $i \in \{0, \ldots, n\}$, such a scheme does not hide the policy, yet still satisfies indistinguishability.

Since TPBS signing requires the user's identity key and the produced signatures are traceable by the TA, we extend the definition of privacy to include user anonymity in addition to policy-privacy. Moreover, we define non-frameability and traceability to capture the newly introduced traceability feature. We also define simulatability and extractability as the stronger notions of privacy and unforgeability. Note that our definition of simulatability and extractability differs from those in PBS in that they include the newly introduced signer identity and tracing feature. In what follows, we give the formal definitions of the TPBS security notions. The oracles used in the security experiments are defined in Fig. 1.

Note that $\mathcal{O}\mathsf{KeyGen}$ is set up to generate the signer identity key from scratch and return it to the adversary along with the policy key. Such a setup allows the adversary to corrupt as many users as it wants without engaging with the oracle interactively.

3.2 Privacy

TPBS ensures privacy if it guarantees signer anonymity and policy-privacy, which are defined as follows.

Signer anonymity. Anonymity is modeled by the indistinguishability experiment in Fig. 2, where the adversary has access to $\mathcal{O}\mathsf{KeyGen}(.)$, $\mathcal{O}\mathsf{USign}(.)$, $\mathcal{O}IdLoRSign$, and $\mathcal{O}\mathsf{Trace}(.)$ oracles. The challenge oracle $\mathcal{O}IdLoRSign$ is ini-

```
$\mathcal{O}\mathsf{KeyGen}(i, p)$
if $i \in \mathcal{U}$ return $\perp$
$(pk_i, sk_i, ID_i) \leftarrow \mathsf{UserKeyGen}(pp_{\mathsf{TPBS}})$
$((Reg[i]), (sk^{i}_{\mathsf{TPBS}})) \leftarrow \mathsf{IDKeyGen}((sk^{TA}_{\mathsf{TPBS}}) \xleftrightarrow[\sigma^{i}_{ID}]{(i, ID_i)} (sk_i))$
$sk^{p}_{\mathsf{TPBS}} \leftarrow \mathsf{PolicyKeyGen}(sk^{Issuer}_{\mathsf{TPBS}}, p)$
$\mathcal{T} = \mathcal{T} \cup \{i, sk_i\}; \quad \mathcal{L} = \mathcal{L} \cup \{p\}$
return $(sk^{i}_{\mathsf{TPBS}}, sk^{p}_{\mathsf{TPBS}})$
```

```
$\mathcal{O}\mathsf{USign}(i_j, m, p, w_p)$
if $\mathsf{PC}((p, m), w_p) = 0 \vee i_j \notin \mathcal{U}$
  return $\perp$
$(sk^{i_j}_{\mathsf{TPBS}}) \leftarrow \mathcal{Q}_i[i_j]$
$sk^{p}_{\mathsf{TPBS}} \leftarrow \mathsf{PolicyKeyGen}(sk^{Issuer}_{\mathsf{TPBS}}, p)$
$\sigma_m \leftarrow \mathsf{Sign}(pk^{TA}_{\mathsf{TPBS}}, pk^{Issuer}_{\mathsf{TPBS}}, sk^{p}_{\mathsf{TPBS}}, sk^{i_j}_{\mathsf{TPBS}}, m, p, w_p)$
$\mathcal{M} = \mathcal{M} \cup \sigma_m$
return $(\sigma_m, \sigma^{i}_{ID}, sk^{p}_{\mathsf{TPBS}})$
```

```
$\mathcal{O}\mathsf{IdLoRSign}(i_{j_0}, i_{j_1}, m, p, w_p)$
if $\mathsf{PC}((p, m), w_p) = 0 \vee i_{j_0}, i_{j_1} \notin \mathcal{U}$
  return $\perp$
$(sk^{i_0}_{\mathsf{TPBS}}) \leftarrow \mathcal{Q}_i[j_0][1]; \quad (sk^{i_1}_{\mathsf{TPBS}}) \leftarrow \mathcal{Q}_i[j_1][1]$
$sk^{p}_{\mathsf{TPBS}} \leftarrow \mathsf{PolicyKeyGen}(sk^{Issuer}_{\mathsf{TPBS}}, p)$
$\sigma_{m_b} \leftarrow \mathsf{Sign}(pk^{TA}_{\mathsf{TPBS}}, pk^{Issuer}_{\mathsf{TPBS}}, sk^{p}_{\mathsf{TPBS}}, sk^{i_b}_{\mathsf{TPBS}}, m, p, w_p)$
$\mathcal{M}' = \mathcal{M}' \cup (m, \sigma_{m_b})$
return $(\sigma_{m_b}, sk^{p}_{\mathsf{TPBS}})$
```

```
$\mathcal{O}\mathsf{Sign}(m, i, p, w_p)$
if $i \in \mathcal{T} \wedge p \in \mathcal{L}$
  return $\perp$
if $i \in \mathcal{Q}_i$; $sk^{i}_{\mathsf{TPBS}} = \mathcal{Q}_i[i]$
elseif $p \in \mathcal{Q}_p$; $sk^{p}_{\mathsf{TPBS}} = \mathcal{Q}_p[p]$
else
  $(pk_i, sk_i, ID_i) \leftarrow \mathsf{UserKeyGen}(pp_{\mathsf{TPBS}})$
  $((Reg[i]), (sk^{i}_{\mathsf{TPBS}})) \leftarrow \mathsf{IDKeyGen}((sk^{TA}_{\mathsf{TPBS}}) \xleftrightarrow[\sigma^{i}_{ID}]{(i, ID_i)} (sk_i))$
  $sk^{p}_{\mathsf{TPBS}} \leftarrow \mathsf{PolicyKeyGen}(sk^{Issuer}_{\mathsf{TPBS}}, p)$
  $\mathcal{Q}_i[i] = sk^{i}_{\mathsf{TPBS}}$; $\mathcal{Q}_p[p] = sk^{p}_{\mathsf{TPBS}}$
$\sigma_m \leftarrow \mathsf{Sign}(pk^{TA}_{\mathsf{TPBS}}, pk^{Issuer}_{\mathsf{TPBS}}, sk^{p}_{\mathsf{TPBS}}, sk^{i}_{\mathsf{TPBS}}, m, p, w_p)$
$\mathcal{M} = \mathcal{M} \cup (m, \sigma_m)$
return $\sigma_m$
```

```
$\mathcal{O}\mathsf{PLoRSign}(i, m, p_0, w_{p_0}, p_1, w_{p_1})$
if $\mathsf{PC}((p_0, m), w_{p_0}) = 0 \vee \mathsf{PC}((p_1, m), w_{p_1}) = 0$
  return $\perp$
$(pk_i, sk_i, ID_i) \leftarrow \mathsf{UserKeyGen}(pp_{\mathsf{TPBS}})$
$((Reg[i]), (sk^{i}_{\mathsf{TPBS}})) \leftarrow \mathsf{IDKeyGen}((sk^{TA}_{\mathsf{TPBS}}) \xleftrightarrow[\sigma^{i}_{ID}]{(i, ID_i)} (sk_i))$
$sk^{p_0}_{\mathsf{TPBS}} \leftarrow \mathsf{PolicyKeyGen}(sk^{Issuer}_{\mathsf{TPBS}}, p_0)$
$sk^{p_1}_{\mathsf{TPBS}} \leftarrow \mathsf{PolicyKeyGen}(sk^{Issuer}_{\mathsf{TPBS}}, p_1)$
$\sigma_{m_b} \leftarrow \mathsf{Sign}(pk^{TA}_{\mathsf{TPBS}}, pk^{Issuer}_{\mathsf{TPBS}}, sk^{p_b}_{\mathsf{TPBS}}, sk^{i}_{\mathsf{TPBS}}, m, p_b, w_{p_b})$
$\mathcal{M} = \mathcal{M} \cup \{m, \sigma_m\}$
return $(\sigma_{m_b}, sk^{p_b}_{\mathsf{TPBS}}, sk^{i}_{\mathsf{TPBS}})$
```

```
$\mathcal{O}\mathsf{Sim\text{-}or\text{-}Sign}(i_j, p, m, w_p)$
if $\mathsf{PC}((p, m), w_p) = 1$
  $(sk^{i_j}_{\mathsf{TPBS}}) \leftarrow \mathcal{Q}_i[i_j]$
  $sk^{p}_{\mathsf{TPBS}} \leftarrow \mathsf{PolicyKeyGen}(sk^{Issuer}_{\mathsf{TPBS}}, p)$
  $\sigma_{m_0} \leftarrow \mathsf{Sign}(pk^{TA}_{\mathsf{TPBS}_0}, pk^{Issuer}_{\mathsf{TPBS}_0}, sk^{p}_{\mathsf{TPBS}}, sk^{i_j}_{\mathsf{TPBS}}, m, w_p)$
  $\sigma_{m_1} \leftarrow \mathsf{SimSign}(tr_{NIZK}, pk^{TA}_{\mathsf{TPBS}_1}, pk^{Issuer}_{\mathsf{TPBS}_1}, m)$
  $\mathcal{M}' = \mathcal{M}' \cup \{m, \sigma_{m_b}\}$
  return $(\sigma_{m_b}, sk^{p}_{\mathsf{TPBS}})$
return $\perp$
```

```
$\mathcal{O}\mathsf{Trace}(m, \sigma_m)$
if $\sigma_m \in \mathcal{M}'$ return $\perp$
$(i, \pi_{Trace}) \leftarrow \mathsf{Trace}(pk^{TA}_{\mathsf{TPBS}}, pk^{Issuer}_{\mathsf{TPBS}}, Reg, m, \sigma_m)$
return $(i, \pi_{Trace})$
```

Fig. 1: TPBS Security Oracles

tialized with a random bit $b \in \{0, 1\}$. The adversary inputs to $\mathcal{O}IdLoRSign$ are (i_0, i_1, m, p, w_p) where the adversary chooses i_0, i_1 from a predefined list of users $\mathcal{U}$ that it has no access to their signing keys. After verifying that $\mathsf{PC}((p, m), w_p) = 1$ and $i_0, i_1 \in \mathcal{U}$, the oracle generates σ_{m_b} for the message m using (sk^{p}_{TPBS},$sk^{i_b}_{\mathsf{TPBS}}$). Finally, the oracle returns the tuple $(\sigma_{m_b}, sk^{p}_{\mathsf{TPBS}})$. The adversary wins if it can determine the bit b with more than the negligible probability. The adversary has access to $\mathcal{O}\mathsf{USign}(.)$ oracle, which on input $(i \in \mathcal{U}, m, p, w_p)$, it obtains a signature on massage m under the identity key of $i \in \mathcal{U}$ and any policy of its choice. Furthermore, $\mathcal{O}\mathsf{USign}(.)$ returns the TA signature σ^{i}_{ID} of the user i to simulate the case where σ^{i}_{ID} is leaked without the knowledge of sk_i. Furthermore, we give the adversary access to $sk^{Issuer}_{\mathsf{TPBS}}$ to simulate the case of a corrupt issuer. Note, to prevent trivial attacks, the queries to $\mathcal{O}\mathsf{KeyGen}(.)$ are limited to users' identities not in $\mathcal{U}$ which models the set of honest users. Also, the adversary cannot query the $\mathcal{O}Trace$ with the output of $\mathcal{O}IdLoRSign$.

Anonymity is defined in a selfless setting where we do not provide the adversary with access to the identity keys of the two signers, $sk^{i_0}_{\mathsf{TPBS}}$ and $sk^{i_1}_{\mathsf{TPBS}}$, involved in the query to $\mathcal{O}IdLoRSign$ [7]. This models the case where an internal adversary should not be able to distinguish between two signatures generated under two identities different than its own, even if both signatures are generated using the same policy key. Such a restriction is essential to construct a significantly more efficient scheme [5].

Definition 2. *(*TPBS *Anonymity) The* TPBS *scheme is anonymous if for any PPT adversary* $\mathcal{A}$*,* $|\Pr[\boldsymbol{Exp}^{Anonymity}_{\mathcal{A},\mathsf{TPBS}}(\lambda) = 1] - \frac{1}{2}| \leq \epsilon(\lambda)$*, where* $\boldsymbol{Exp}^{Anonymity}_{\mathcal{A},\mathsf{TPBS}}$ *is defined in Fig. 2.*

```
$\mathbf{Exp}^{Anonymity}_{\mathcal{A},\mathsf{TPBS}}(\lambda)$
$b \xleftarrow{\$} \{0,1\}, \mathcal{U} = \{0,\ldots,n\}, \mathcal{M}' = \{\}, \mathcal{Q}_i = [\,], pp_{\mathsf{TPBS}} \leftarrow \mathsf{ppGen}(1^\lambda)$
$(pk^{TA}_{\mathsf{TPBS}}, sk^{TA}_{\mathsf{TPBS}}) \leftarrow \mathsf{TASetup}(pp_{\mathsf{TPBS}})$
$(pk^{Issuer}_{\mathsf{TPBS}}, sk^{Issuer}_{\mathsf{TPBS}}) \leftarrow \mathsf{IssuerSetup}(pp_{\mathsf{TPBS}})$
foreach $i_j \in \mathcal{U}$
  $(pk_{i_j}, sk_{i_j}, ID_{i_j}) \leftarrow \mathsf{UserKeyGen}(pp_{\mathsf{TPBS}})$
  $((Reg[i_j]), (sk^{i_j}_{\mathsf{TPBS}})) \leftarrow \mathsf{IDKeyGen}((sk^{TA}_{\mathsf{TPBS}}) \xleftrightarrow[\sigma^{i_j}_{ID}]{(i, ID_{i_j})} (sk_{i_j}))$
  $\mathcal{Q}_i[i_j] = sk^{i_j}_{\mathsf{TPBS}}$
$b' \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{KeyGen}(.),\mathcal{O}\mathsf{USign}(.),\mathcal{O}\mathsf{Trace}(.),\mathcal{O}\mathsf{IdLoRSign}(.,b)}(\mathcal{U}, pp_{\mathsf{TPBS}}, pk^{TA}_{\mathsf{TPBS}}, pk^{Issuer}_{\mathsf{TPBS}}, sk^{Issuer}_{\mathsf{TPBS}})$
if $b = b'$
  return $\top$
return $\bot$
```

Fig. 2: TPBS Anonymity Experiment

Policy-privacy. Policy-privacy is modeled by the indistinguishability experiment in Fig. 3, where the adversary has access to $\mathcal{O}\mathsf{KeyGen}(.)$ and $\mathcal{O}PLoRSign$ oracles. The challenge oracle $\mathcal{O}PLoRSign$ is initialized with a random bit $b \in \{0,1\}$. The adversary inputs to $\mathcal{O}PLoRSign$ oracle are $(i, m, p_0, w_{p_0}, p_1, w_{p_1})$. After verifying that $\mathsf{PC}((p_0, m), w_{p_0}) = 1$, and $\mathsf{PC}((p_1, m), w_{p_1}) = 1$, the oracle generates sk^{i}_{TPBS} and $sk^{p_b}_{\mathsf{TPBS}}$ for $b \in \{0,1\}$. It then signs m using $(sk^{p_b}_{\mathsf{TPBS}}, sk^{i}_{\mathsf{TPBS}})$ and returns σ_{m_b} along with $sk^{p_b}_{\mathsf{TPBS}}$ and sk^{i}_{TPBS}. The adversary wins if it can determine the bit b with a probability better than the random guess. Note that we give the adversary access to sk^{TA}_{TPBS} and $sk^{Issuer}_{\mathsf{TPBS}}$ to simulate the case of a corrupt TA and\or issuer.

Definition 3. *(*TPBS *Policy-privacy) The* TPBS *scheme is policy-private if for any PPT adversary* $\mathcal{A}$*,* $|\Pr[\boldsymbol{Exp}^{Policy-privacy}_{\mathcal{A},\mathsf{TPBS}}(\lambda) = 1] - \frac{1}{2}| \leq \epsilon(\lambda)$*, where* $\boldsymbol{Exp}^{Policy-privacy}_{\mathcal{A},\mathsf{TPBS}}$ *is defined in Fig. 3.*

```
$\mathbf{Exp}^{Policy-privacy}_{\mathcal{A},\mathsf{TPBS}}(\lambda)$
$b \xleftarrow{\$} \{0,1\}, pp_{\mathsf{TPBS}} \leftarrow \mathsf{ppGen}(1^{\lambda})$
$(pk^{TA}_{\mathsf{TPBS}}, sk^{TA}_{\mathsf{TPBS}}) \leftarrow \mathsf{TASetup}(pp_{\mathsf{TPBS}})$
$(pk^{Issuer}_{\mathsf{TPBS}}, sk^{Issuer}_{\mathsf{TPBS}}) \leftarrow \mathsf{IssuerSetup}(pp_{\mathsf{TPBS}})$
$(b') \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{KeyGen}(.),\mathcal{O}\mathsf{PLoRSign}(.,b)}(pp_{\mathsf{TPBS}}, pk^{TA}_{\mathsf{TPBS}}, pk^{Issuer}_{\mathsf{TPBS}}, sk^{TA}_{\mathsf{TPBS}}, sk^{Issuer}_{\mathsf{TPBS}})$
if $b = b'$
  return $\top$
return $\bot$
```

Fig. 3: TPBS Policy-privacy Experiment

Consider a PBS scheme where for every message m there is only one policy p such that $\mathsf{PC}((p,m),w_p) = 1$; then the aforementioned policy-privacy definition can not hide the associated policy. It has been proven that simulatability is a stronger notion of policy-privacy that remedies the aforementioned limitation [3]. Since the same limitation is inherited in TPBS, thus, we also define simulatability, and we prove that our definition implies the privacy of TPBS, which is defined as both anonymity and policy-privacy.

Simulatability. This security notion requires the existence of a simulator that can create simulated signatures without having access to any of the users' signing keys or witnesses. Yet, such signatures are indistinguishable from real signatures. Thus, we assume that for every TPBS procedure, there exists a simulated procedure whose output is indistinguishable from the non-simulated one. We denote such a procedure with the Sim prefix. More precisely, we require the following algorithms, SimppGen, SimTASetup, SimIssuerSetup, SimUserKeyGenTPBS, SimIDKeyGen, SimPolicyKeyGen, SimSign, and SimTraceTPBS. Note that SimppGen, SimTASetup, and SimIssuerSetup also output the trapdoor information tr_{NIZK}, tr_{TA}, and tr_{Issuer}, respectively. Such trapdoor outputs are used as inputs to the other relevant simulated procedures instead of the secret inputs. We give the definitions of the simulated procedures in Fig 9 after we present the generic construction.

We formally define simulatability in a selfless setting by the experiment in Fig. 4, in which the adversary has access to $\mathcal{O}$KeyGen(.), $\mathcal{O}$USign(.), $\mathcal{O}$Trace(.), and $\mathcal{O}$Sim-or-Sign(.) oracles. $\mathcal{O}$Sim-or-Sign(.) is its challenge oracle which on the input of some i_j from a predefined list of honest users identities $\mathcal{U}$, a message m, a policy p, and a witness w_p that m conforms to p, the oracle outputs a signature σ_m. The adversary wins if it can determine whether σ_m is generated using i_j identity key and p policy key or it is a simulated signature. To prevent trivial attacks, the adversary cannot query the $\mathcal{O}$Trace(.) with the signatures generated by the challenging oracle.

Definition 4. *(*TPBS *Simulatability) The* TPBS *scheme is simulatable if for any PPT adversary* $\mathcal{A}$, $|\Pr[\boldsymbol{Exp}^{SIM}_{\mathcal{A},\mathsf{TPBS}}(\lambda) = 1] - \frac{1}{2}| \leq \epsilon(\lambda)$*, where the* $\boldsymbol{Exp}^{SIM}_{\mathcal{A},\mathsf{TPBS}}$ *is defined in Fig. 4.*

```
$\mathbf{Exp}^{SIM}_{\mathcal{A},\mathsf{TPBS}}(\lambda)$
$b \xleftarrow{\$} \{0,1\}, \mathcal{U} = \{0,\dots,n\}, \mathcal{M}' = \{\}, \mathcal{Q}_i = [\,]$
$pp_{\mathsf{TPBS}_0} \leftarrow \mathsf{ppGen}(1^\lambda), (pp_{\mathsf{TPBS}_1}, tr_{NIZK}) \leftarrow \mathsf{SimppGen}(1^\lambda)$
$(pk^{TA}_{\mathsf{TPBS}_0}, sk^{TA}_{\mathsf{TPBS}_0}) \leftarrow \mathsf{TASetup}(pp_{\mathsf{TPBS}_0})$
$(pk^{Issuer}_{\mathsf{TPBS}_0}, sk^{Issuer}_{\mathsf{TPBS}_0}) \leftarrow \mathsf{IssuerSetup}(pp_{\mathsf{TPBS}_0})$
$(pk^{TA}_{\mathsf{TPBS}_1}, sk^{TA}_{\mathsf{TPBS}_1}, tr_{TA}) \leftarrow \mathsf{SimTASetup}(pp_{\mathsf{TPBS}_1})$
$(pk^{Issuer}_{\mathsf{TPBS}_1}, sk^{Issuer}_{\mathsf{TPBS}_1}, tr_{Issuer}) \leftarrow \mathsf{SimIssuerSetup}(pp_{\mathsf{TPBS}_1})$
foreach $i_j \in \mathcal{U}$
    $(pk_{i_j}, sk_{i_j}, ID_{i_j}) \leftarrow \mathsf{UserKeyGen}(pp_{\mathsf{TPBS}})$
    $((Reg[i_j]), (sk^{i_j}_{\mathsf{TPBS}})) \leftarrow \mathsf{IDKeyGen}((sk^{TA}_{\mathsf{TPBS}}) \xleftrightarrow[\sigma^{i_j}_{ID}]{(i, ID_{i_j})} (sk_{i_j}))$
    $\mathcal{Q}_i[i_j] = sk^{i_j}_{\mathsf{TPBS}}$
$b' \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{KeyGen}(.), \mathcal{O}\mathsf{USign}(.), \mathcal{O}\mathsf{Trace}(.), \mathcal{O}\mathsf{Sim\text{-}or\text{-}Sign}(.)}(\mathcal{U}, pp_{\mathsf{TPBS}_b}, pk^{TA}_{\mathsf{TPBS}_b}, sk^{TA}_{\mathsf{TPBS}_b}, pk^{Issuer}_{\mathsf{TPBS}_b}, sk^{Issuer}_{\mathsf{TPBS}_b})$
if $b = b'$ return $\top$
return $\bot$
```

Fig. 4: TPBS Simulatability Experiment

3.3 Unforgeability

Intuitively unforgeability is the infeasibility of creating a valid signature on a message m without holding the policy key for policy p to which m conforms. To model users' corruption and collusion attacks where users could combine their policy keys to sign messages non of them is authorized to, Bellare and Fuchsbauer have defined the unforgeability of the PBS scheme by an experiment where the adversary is allowed to query a key generation oracle to generate user keys and gain access to some of them. However, in their definition, it becomes hard to efficiently determine if an adversary has won the unforgeability experiment by producing a valid signature such that $\mathsf{PC}((p,m),w_p) = 1$ using a queried policy key or not since policy-privacy requires hiding the policy and witness used in generating a specific signature. To overcome the aforementioned limitation, they defined extractability as a strengthened version of unforgeability and proved that extractability implies unforgeability [3]. Since TPBS privacy requires hiding the policy, witness, and signer's identity used in generating signatures over m, we define extractability and adapt it to imply the unforgeability for TPBS.

Extractability. We formally define TPBS extractability by the experiment in Fig. 5, where we assume the existence of an extractor algorithm Extr which upon inputting a valid message signature pair (m,σ_m) in addition to trapdoor information tr_{NIZK}, it outputs the tuple $(p, sk_i, sk^p_{\mathsf{TPBS}}, w_p)$. An adversary $\mathcal{A}$ who has access to $\mathcal{O}\mathsf{KeyGen}$ and $\mathcal{O}\mathsf{Sign}$ oracles (Fig. 1) wins $\mathbf{Exp}^{Ext}_{\mathcal{A},\mathsf{TPBS}}$ if it outputs a valid message signature pair (m^*, σ_{m^*}) such that either i) it does not hold some sk^{i*}_{TPBS} that is obtained from $\mathcal{O}\mathsf{KeyGen}$ oracle or for all p, it obtained sk^p_{TPBS} by querying $\mathcal{O}\mathsf{KeyGen}$ oracle, ii) it does not hold an sk^{p*}_{TPBS} corresponds to p^* such that $\mathsf{PC}((p^*,m^*),w^*_p) = 1$ or iii) $\mathsf{PC}((p^*,m^*),w^*_p) = 0$. Note that since tr_{NIZK} is required by Extr algorithm, the extractability experiment is initialized using $\mathsf{SimppGen}(1^\lambda)$ algorithm rather than $\mathsf{ppGen}(1^\lambda)$, and all other algorithms are kept the same.

Definition 5. *(*TPBS *Extractability) a* TPBS *scheme is extractable if for any PPT adversary* $\mathcal{A}$*,* $\Pr[\boldsymbol{Exp}^{Ext}_{\mathcal{A},\mathsf{TPBS}}(\lambda) = 1] \leq \epsilon(\lambda)$*, where* $\boldsymbol{Exp}^{Ext}_{\mathcal{A},\mathsf{TPBS}}$ *is defined in Fig. 5.*

```
$\mathbf{Exp}^{Ext}_{\mathcal{A},\mathsf{TPBS}}(\lambda)$
$(pp_{\mathsf{TPBS}}, tr_{NIZK}) \leftarrow \mathsf{SimppGen}(1^\lambda)$
$(pk^{TA}_{\mathsf{TPBS}}, sk^{TA}_{\mathsf{TPBS}}, Reg) \leftarrow \mathsf{TASetup}(pp_{\mathsf{TPBS}})$
$(pk^{Issuer}_{\mathsf{TPBS}}, sk^{Issuer}_{\mathsf{TPBS}}) \leftarrow \mathsf{IssuerSetup}(pp_{\mathsf{TPBS}})$
$\mathcal{Q}_i = \mathcal{Q}_p = [\ ]$
$\mathcal{T} = \mathcal{L} = \mathcal{M} = \{\}$
$(m^*, \sigma_{m^*}) \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{KeyGen}(.),\mathcal{O}\mathsf{Sign}(.)}(pp_{\mathsf{TPBS}}, pk^{TA}_{\mathsf{TPBS}}, pk^{Issuer}_{\mathsf{TPBS}})$
if $(m^*, \sigma_{m^*}) \in \mathcal{M} \ \vee\ \mathsf{Verify}(pk^{TA}_{\mathsf{TPBS}}, pk^{Issuer}_{\mathsf{TPBS}}, m^*, \sigma_{m^*}) = \perp$
    return $\perp$
$(p^*, sk^*_i, sk^p_{\mathsf{TPBS}}, w_{p^*}) \leftarrow \mathsf{Extr}(tr_{NIZK}, m^*, \sigma_{m^*})$
if $sk^*_i \notin \mathcal{T} \vee p^* \notin \mathcal{L} \vee \mathsf{PC}((p^*, m^*), w^*_p) = 0$
    return $\top$
return $\perp$
```

Fig. 5: TPBS Extractability Experiment

3.4 Non-frameability

This property ensures that even if the tracing authority, issuer, and all corrupt users in the scheme collude together, they cannot produce a valid signature that is traced back to an honest user. TPBS non-frameability is modeled by the experiment defined in Fig. 6, in which the adversary has access to both TA and issuer secret keys $(sk^{TA}_{\mathsf{TPBS}}, sk^{Issuer}_{\mathsf{TPBS}})$, in addition to $\mathcal{O}\mathsf{KeyGen}$, $\mathcal{O}\mathsf{USign}$, and $\mathcal{O}\mathsf{Trace}$ oracles. The adversary wins if it outputs a verifiable (m^*, σ_{m^*}) that has not been queried to $\mathcal{O}\mathsf{USign}$ and when (m^*, σ_{m^*}) is traced back to its signer, the tracing algorithm outputs an identity of one of the honest users in $\mathcal{U}$. Additionally, the output of $\mathcal{O}\mathsf{Trace}$ oracle should be verifiable using the Judge algorithm.

Definition 6. *(*TPBS *Non-frameability) a* TPBS *scheme is non-frameable if for any PPT adversary* $\mathcal{A}$*,* $\Pr[\boldsymbol{Exp}^{Non-frameability}_{\mathcal{A},\mathsf{TPBS}}(\lambda) = 1] \leq \epsilon(\lambda)$*, where the non-frameability experiment is defined in Fig. 6.*

3.5 Traceability

Traceability requires that even if all scheme users collude together, they cannot produce a signature that cannot be traced. We require the tracing authority to be honest, as knowing the secret key of the tracing authority would allow the adversary to sign a dummy sk_i under the tracing authority's secret key resulting in an untraceable signature. TPBS traceability is modeled by the experiment defined in Fig. 7, in which the adversary has access to $\mathcal{O}\mathsf{KeyGen}$ and $\mathcal{O}\mathsf{Trace}$ procedures. We omit the adversarial access to $\mathcal{O}\mathsf{Sign}$ oracle since the adversary could corrupt as many users as it wants and get access to their keys. Hence

```
$\mathbf{Exp}^{Non-frameability}_{\mathcal{A},\mathsf{TPBS}}(\lambda)$
$\mathcal{U} = \{0, \ldots, n\}, \mathcal{M} = \{\}, \mathcal{Q}_i = [\,], pp_{\mathsf{TPBS}} \leftarrow \mathsf{ppGen}(1^\lambda)$
$(pk^{TA}_{\mathsf{TPBS}}, sk^{TA}_{\mathsf{TPBS}}) \leftarrow \mathsf{TASetup}(pp_{\mathsf{TPBS}}), (pk^{Issuer}_{\mathsf{TPBS}}, sk^{Issuer}_{\mathsf{TPBS}}) \leftarrow \mathsf{IssuerSetup}(pp_{\mathsf{TPBS}})$
foreach $i_j \in \mathcal{U}$
  $(pk_{i_j}, sk_{i_j}, ID_{i_j}) \leftarrow \mathsf{UserKeyGen}(pp_{\mathsf{TPBS}})$
  $((Reg[i_j]), (sk^{i_j}_{\mathsf{TPBS}})) \leftarrow \mathsf{IDKeyGen}((sk^{TA}_{\mathsf{TPBS}} \xleftrightarrow[\sigma^{i_j}_{ID}]{(i, ID_{i_j})} (sk_{i_j}))$
  $\mathcal{Q}_i[i_j] = sk^{i_j}_{\mathsf{TPBS}}$
$(m^*, \sigma_{m^*}) \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{KeyGen}(.), \mathcal{O}\mathsf{USign}(.), \mathcal{O}\mathsf{Trace}(.)}(\mathcal{U}, pp_{\mathsf{TPBS}}, pk^{TA}_{\mathsf{TPBS}}, pk^{Issuer}_{\mathsf{TPBS}}, sk^{Issuer}_{\mathsf{TPBS}}, sk^{TA}_{\mathsf{TPBS}})$
if $(m^*, \sigma_{m^*}) \in \mathcal{M} \vee \mathsf{Verify}(pk^{TA}_{\mathsf{TPBS}}, pk^{Issuer}_{\mathsf{TPBS}}, m^*, \sigma_{m^*}) = \bot$
  return $\bot$
$(i^*, \pi^*_{Trace}) \leftarrow \mathsf{Trace}(pk^{TA}_{\mathsf{TPBS}}, pk^{Issuer}_{\mathsf{TPBS}}, Reg, m, \sigma_m)$
if $i^* \notin \mathcal{U}$
  return $\bot$
return $\mathsf{Judge}(pk^{TA}_{\mathsf{TPBS}}, pk^{Issuer}_{\mathsf{TPBS}}, m^*, \sigma_{m^*}, i^*, \pi^*_{Trace})$
```

Fig. 6: TPBS Non-Frameability Experiment

it could use the signing algorithm directly Sign(.) to produce signatures. The Adversary wins if it outputs a verifiable (m^*, σ_{m^*}), which when traced, the tracing algorithm Trace outputs $\bot$.

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

```
$\mathbf{Exp}^{Traceability}_{\mathcal{A},\mathsf{TPBS}}(\lambda)$
$(pp_{\mathsf{TPBS}}) \leftarrow \mathsf{ppGen}(1^\lambda), (pk^{TA}_{\mathsf{TPBS}}, sk^{TA}_{\mathsf{TPBS}}) \leftarrow \mathsf{TASetup}(pp_{\mathsf{TPBS}})$
$(pk^{Issuer}_{\mathsf{TPBS}}, sk^{Issuer}_{\mathsf{TPBS}}) \leftarrow \mathsf{IssuerSetup}(pp_{\mathsf{TPBS}})$
$(m^*, \sigma_{m^*}) \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{KeyGen}(.), \mathcal{O}\mathsf{Trace}(.)}(pp_{\mathsf{TPBS}}, pk^{TA}_{\mathsf{TPBS}}, pk^{Issuer}_{\mathsf{TPBS}})$
if $\mathsf{Verify}(pk^{TA}_{\mathsf{TPBS}}, pk^{Issuer}_{\mathsf{TPBS}}, m^*, \sigma_{m^*})$
  $(i^*, \pi^*_{Trace}) \leftarrow \mathsf{Trace}(pk^{TA}_{\mathsf{TPBS}}, pk^{Issuer}_{\mathsf{TPBS}}, Reg, m^*, \sigma_{m^*})$
  if $i = \bot$
    return $\top$
return $\bot$
```

Fig. 7: TPBS Traceability Experiment

4 TPBS Generic Construction

The main building blocks of the new construction are a EUF-CMA RDS scheme capable of signing a commitment on a secret message, a SE-NIZK proof system, and a digital signature scheme. The general idea of the new scheme is that in addition to the policy key sk^p_{TPBS} that is generated by the issuer using PolicyKeyGen and shared with any user who is allowed to sign a message m conforming to p, each user has to run an interactive algorithm IDKeyGen with the TA to obtain an identity key sk^i_{TPBS}. In IDKeyGen, the user gets the TA's RDS signature σ^i_{ID} on a user-chosen secret value sk_i, where sk_i is generated using UserKeyGen. σ^i_{ID}

along with sk_i are the user's identity key sk^i_{TPBS}. The TA keeps track of users' registration information ID_i in a secret registry Reg. Users generate their own registration information ID_i using a one-way function over sk_i, where it contains a tracing trapdoor that allows the TA to trace the generated TPBS signature to its original signer. More precisely, ID_i contains $C_i = (c_i, \tilde{c}_i) = f(sk_i)$ and the user's digital signature τ_i over c_i where (c_i, τ_i) could be publicly mapped to such a user's identity and $\tilde{c}_i$ is held secretly by the TA as the tracing trapdoor information. To ensure non-framability, an RDS signing algorithm is used to sign the output of a one-way function c_i rather than sk_i itself, yet the produced signature is verifiable over sk_i. To sign a message m, the user generates a rerandomized version of the TA signature ${\sigma'}^i_{ID}$ along with a SE-NIZK proof π_m for the relation $\mathbb{R}'_{\mathbb{NP}}$ that is given by

$$((pk^{TA}_{\mathsf{TPBS}}, {\sigma'}^i_{ID}, pk^{Issuer}_{\mathsf{TPBS}}, m), (sk_i, p, sk^p_{\mathsf{TPBS}}, w_p)) \in \mathbb{R}'_{\mathbb{NP}} \Leftrightarrow$$

$$\mathsf{VerifyRDS}(pk^{TA}_{\mathsf{TPBS}}, sk_i, {\sigma'}^i_{ID}) = 1 \tag{1a}$$

$$\wedge\ \mathsf{VerifySig}(pk^{Issuer}_{\mathsf{TPBS}}, p, sk^p_{\mathsf{TPBS}}) = 1 \tag{1b}$$

$$\wedge\ \mathsf{PC}((p, m), w) = 1, \tag{1c}$$

whose statements $X = (pk^{TA}_{\mathsf{TPBS}}, {\sigma'}^i_{ID}, pk^{Issuer}_{\mathsf{TPBS}}, m)$ with witnesses $W = (sk_i, p, sk^p_{\mathsf{TPBS}}, w_p)$. Intuitively, π_m proves that a) ${\sigma'}^i_{ID}$ is the TA signature over some signer-generated secret value sk_i, b) the user holds the issuer's signature over some policy p, and c) the message m conforms the policy p under some witness w_p, i.e. $\mathsf{PC}((p, m), w) = 1$. Signature verification is done by verifying π_m over the statements X. To trace a signature to its signer, the TA exhaustively searches Reg for a matching $\tilde{c}_i$ that verifies ${\sigma'}^i_{ID}$ in the signature; once a match is found, the TA outputs the resisted user's identity i along with (c_i, τ_i) in addition to a NIZK for the knowledge of $\tilde{c}_i$ such that c_i and $\tilde{c}_i$ are generated $f(sk_i)$ and ${\sigma'}^i_{ID}$ is verifiable over sk_i.

One advantage of using a sign-rerandomize-proof paradigm rather than a sign-encrypt-proof paradigm is that the former paradigm produces a significantly more efficient signature than the latter [16, 5]. On the other hand, the tracing algorithm becomes a linear operation in the number of scheme users, which is considered an affordable price since tracing is an infrequent operation and is run by a computationally powerful TA [5].

Figure 8 depicts the complete generic construction of TPBS. Note that we use two different instances of the digital signature scheme. The issuer uses one to sign a policy p in $\mathsf{PolicyKeyGen}$, and the scheme users use the other to sign the output of the one-way function to generate ID_i in $\mathsf{UserKeyGen}$. We label the latter with the subscript Σ.

In Fig. 9, we show how $\mathsf{SimppGen}(.)$, $\mathsf{SimSign}(.)$, $\mathsf{Extr}(.)$ are constructed in accordance with the concrete construction in Fig. 8. Since tr_{TA}, and tr_{Issuer} is equal to sk^{TA}_{TPBS} and $sk^{Issuer}_{\mathsf{TPBS}}$, respectively, we omit the details of $\mathsf{SimTASetup}(.)$, $\mathsf{SimIssuerSetup}(.)$, $\mathsf{SimUserKeyGenTPBS}(.)$, $\mathsf{SimIDKeyGen}(.)$, $\mathsf{SimPolicyKeyGen}(.)$, and $\mathsf{SimTrace}(.)$ which are defined in the same way as $\mathsf{TASetup}(.)$, $\mathsf{IssuerSetup}(.)$, $\mathsf{IDKeyGen}(.)$, $\mathsf{PolicyKeyGen}(.)$, and $\mathsf{Trace}(.)$, respectively, .

```
$\mathsf{ppGen}(1^\lambda)$
  $crs \leftarrow \mathsf{SetupNIZK}(1^\lambda), pp_{RDS} \leftarrow \mathsf{ppGenRDS}(1^\lambda)$
  $pp_{Sig} \leftarrow \mathsf{ppGenSig}(1^\lambda), pp_{Sig_\Sigma} \leftarrow \mathsf{ppGenSig}(1^\lambda)$
  **return** $pp_{\mathsf{TPBS}} = \{crs, pp_{RDS}, pp_{Sig}, pp_{Sig_\Sigma}\}$

$\mathsf{TASetup}(pp_{\mathsf{TPBS}})$
  $(pk^{TA}_{RDS}, sk^{TA}_{RDS}) \leftarrow \mathsf{KeyGenRDS}(pp_{RDS})$
  $(pk^{TA}_{\mathsf{TPBS}}, sk^{TA}_{\mathsf{TPBS}}) = (pk^{TA}_{RDS}, sk^{TA}_{RDS}), Reg = [\,]$
  **return** $(pk^{TA}_{\mathsf{TPBS}}, sk^{TA}_{\mathsf{TPBS}})$

$\mathsf{IssuerSetup}(pp_{Sig})$
  $(pk^{Issuer}_{\mathsf{TPBS}}, sk^{Issuer}_{\mathsf{TPBS}}) \leftarrow \mathsf{KeyGenSig}(pp_{\mathsf{TPBS}})$
  **return** $(pk^{Issuer}_{\mathsf{TPBS}}, sk^{Issuer}_{\mathsf{TPBS}})$

$\mathsf{UserKeyGen}(pp_{\mathsf{TPBS}}, i)$
  $(pk^i_{Sig_\Sigma}, sk^i_{Sig_\Sigma}) \leftarrow \mathsf{KeyGenSig}(pp_{Sig_\Sigma})$
  $\mathcal{D}[i] = pk_i = (pk^i_{Sig_\Sigma}), sk_i \xleftarrow{\$} \mathbb{Z}_p^*, C_i = (c_i, \tilde{c}_i) = f(sk_i)$
  $\tau_i \leftarrow \mathsf{SignSig}(c_i, sk^i_{Sig_\Sigma}), ID_i = \{C_i, \tau_i\}$
  **return** $(pk_i, sk_i, ID_i)$

$\mathsf{IDKeyGen}((sk^{TA}_{\mathsf{TPBS}}) \underset{\sigma^i_{ID}}{\overset{(i,ID_i)}{\longleftrightarrow}} (sk_i))$
  **User**                                  **TA**
  $(i, ID_i, sk_i)$                          $(sk^{TA}_{\mathsf{TPBS}})$
        $\xrightarrow{i, ID_i}$
                                  $\{C_i, \tau_i\} \leftarrow ID_i, pk^i_{Sig_\Sigma} = \mathcal{D}[i]$
                                  **if** $Reg[i] = \emptyset$
                                    $\wedge\ \mathsf{VerifySig}(pk^i_{Sig_\Sigma}, c_i, \tau_i)$
        $\xleftrightarrow{\mathsf{PoK}(sk_i : C_i = f(sk_i))}$
        $\xleftarrow{\sigma^i_{ID}}$ $\mathsf{SignComRDS}(sk^{TA}_{RDS}, c_i)$
  $sk^i_{\mathsf{TPBS}} = (sk_i, \sigma^i_{ID})$        $Reg[i] = ID_i$

$\mathsf{PolicyKeyGen}(sk^{Issuer}_{\mathsf{TPBS}}, p)$
  $sk^p_{\mathsf{TPBS}} \leftarrow \mathsf{SignSig}(sk^{Issuer}_{\mathsf{TPBS}}, p)$
  **return** $sk^p_{\mathsf{TPBS}}$

$\mathsf{Sign}(pk^{TA}_{\mathsf{TPBS}}, pk^{Issuer}_{\mathsf{TPBS}}, sk^p_{\mathsf{TPBS}}, sk^i_{\mathsf{TPBS}}, m, p, w_p)$
  $(sk_i, \sigma^i_{ID}) = sk^i_{\mathsf{TPBS}}, {\sigma'}^i_{ID} \leftarrow \mathsf{RandomizeRDS}(\sigma^i_{ID})$
  $\pi_m \leftarrow \mathsf{ProveNIZK}(crs, (pk^{TA}_{\mathsf{TPBS}}, {\sigma'}^i_{ID}, pk^{Issuer}_{\mathsf{TPBS}}, m),$
        $(sk_i, p, sk^p_{\mathsf{TPBS}}, w_p))$
  **return** $\sigma_m = ({\sigma'}^i_{ID}, \pi_m)$

$\mathsf{Verify}(pk^{TA}_{\mathsf{TPBS}}, pk^{Issuer}_{\mathsf{TPBS}}, m, \sigma_m)$
  $({\sigma'}^i_{ID}, \pi_m) \leftarrow \sigma_m$
  **return** $\mathsf{VerifyNIZK}(crs, (pk^{TA}_{\mathsf{TPBS}}, {\sigma'}^i_{ID}, pk^{Issuer}_{\mathsf{TPBS}}, m), \pi_m)$

$\mathsf{Trace}(pk^{TA}_{\mathsf{TPBS}}, pk^{Issuer}_{\mathsf{TPBS}}, Reg, m, \sigma_m)$
  **if** $\mathsf{Verify}(pk^{TA}_{\mathsf{TPBS}}, pk^{Issuer}_{\mathsf{TPBS}}, m, \sigma_m)$
    $({\sigma'}^i_{ID}, \pi_m) = \sigma_m$
    **foreach** $\tilde{c}_i \in Reg$
      **if** $\tilde{c}_i$ verifies ${\sigma'}^i_{ID}$
        $(i, C_i, \tau_i) \leftarrow Reg[i]$
        $\pi \leftarrow \mathsf{ProveNIZK}(crs, (c_i, {\sigma'}^i_{ID}), \tilde{c}_i)$
        **return** $(i, (\pi_{Trace})) = (i, (c_i, \tau_i, \pi))$
  **return** $\perp$

$\mathsf{Judge}(pk^{TA}_{\mathsf{TPBS}}, pk^{Issuer}_{\mathsf{TPBS}}, sk^{TA}_{\mathsf{TPBS}}, m, \sigma_m, i, \pi_{Trace})$
  **if** $\mathsf{Verify}(pk^{TA}_{\mathsf{TPBS}}, pk^{Issuer}_{\mathsf{TPBS}}, m, \sigma_m)$
    $pk^i_{Sig_\Sigma} = \mathcal{D}[i], (c_i, \tau_i, \pi) = \pi_{Trace}$
    **if** $(\mathsf{VerifySig}(pk^i_{Sig_\Sigma}, c_i, \tau_i) \wedge \mathsf{VerifyNIZK}$
      **return** $\top$
  **return** $\perp$
```

Fig. 8: Generic Construction of TPBS

```
$\mathsf{SimppGen}(1^\lambda)$
  $(crs, tr_{NIZK}) \leftarrow \mathsf{SimSetupNIZK}(1^\lambda), pp_{RDS} \leftarrow \mathsf{ppGenRDS}(1^\lambda)$
  $pp_{Sig} \leftarrow \mathsf{ppGenSig}(1^\lambda), pp_{Sig_\Sigma} \leftarrow \mathsf{ppGenSig}(1^\lambda)$
  **return** $pp_{\mathsf{TPBS}} \leftarrow (crs, pp_{RDS}, pp_{Sig}, pp_{Sig_\Sigma}),\ tr_{NIZK}$

$\mathsf{SimSign}(tr_{NIZK}, (pk^{TA}_{\mathsf{TPBS}}, pk^{Issuer}_{\mathsf{TPBS}}, m))$
  $sk'_i \xleftarrow{\$} \mathbb{Z}_p^*, \sigma^i_{ID} \leftarrow \mathsf{SignRDS}(sk^{TA}_{\mathsf{TPBS}}, sk'_i)$
  ${\sigma'}^i_{ID} \leftarrow \mathsf{RandomizeRDS}(\sigma^i_{ID})$
  $\pi_m \leftarrow \mathsf{SimProve}(crs, tr_{NIZK}, (pk^{Issuer}_{\mathsf{TPBS}}, pk^{TA}_{\mathsf{TPBS}}, {\sigma'}^i_{ID}, m))$
  **return** $\sigma_m = ({\sigma'}^i_{ID}, \pi_m)$

$\mathsf{Extr}(tr_{NIZK}, m, \sigma_m)$
  $({\sigma'}^i_{ID}, \pi_m) = \sigma_m$
  $(p, sk_i, sk^p_{\mathsf{TPBS}}, w_p) \leftarrow \mathsf{Extr}_{\mathsf{NIZK}}(crs, tr_{NIZK}, m, \pi_m)$
  **return** $(p, sk_i, sk^p_{\mathsf{TPBS}}, w_p)$
```

Fig. 9: TPBS Simulated algorithms

5 TPBS Security

The definition of extractability of TPBS (see Def. 5) implies its unforgeability. The privacy of TPBS includes policy privacy and anonymity. Accordingly, we first prove that simulatability implies both anonymity and policy-privacy. Then we present the security proofs for simulatability (implies privacy), extractability (implies unforgeability), non-frameability, and traceability.

Theorem 1. *Simulatability implies both anonymity and policy-privacy*

Proof. Assuming an adversary $\mathcal{A}$ against TPBS anonymity in $\mathbf{Exp}^{Anonymity}_{\mathcal{A},\mathsf{TPBS}}$ in Fig. 2 (resp. policy-privacy in $\mathbf{Exp}^{Policy-privacy}_{\mathcal{A},\mathsf{TPBS}}$ in Fig. 3), we can construct an adversary $\mathcal{B}$ (resp. $\mathcal{B}'$) against the simulatability of TPBS. $\mathcal{B}$ receives $(\mathcal{U}, pp_{\mathsf{TPBS}_b}, pk^{TA}_{\mathsf{TPBS}_b}, sk^{TA}_{\mathsf{TPBS}_b}, pk^{Issuer}_{\mathsf{TPBS}_b}, sk^{Issuer}_{\mathsf{TPBS}_b})$ from its challenger in the $\mathbf{Exp}^{SIM}_{\mathcal{A},\mathsf{TPBS}}$ in Fig. 4, chooses $d \overset{\$}{\leftarrow} \{0,1\}$, and runs $\mathcal{A}$ on $(\mathcal{U}, pp_{\mathsf{TPBS}_b}, pk^{TA}_{\mathsf{TPBS}_b}, pk^{Issuer}_{\mathsf{TPBS}_b}, sk^{Issuer}_{\mathsf{TPBS}_b})$. Whenever $\mathcal{A}$ queries its challenging oracle $\mathcal{O}\mathsf{IdLoRSign}$ with $(i_{j_0}, i_{j_1}, m, p, w_p)$, if $\mathsf{PC}((p,m), w_p) = 0$ or $i_{j_0}, i_{j_1} \notin \mathcal{U}$, $\mathcal{B}$ returns $\perp$, otherwise it queries its challenger in the simulatability game with (i_{j_d}, m, p, w_p) and returns $(\sigma_{m_b}, sk^{p}_{\mathsf{TPBS}})$ to $\mathcal{A}$. When $\mathcal{A}$ outputs b', $\mathcal{B}$ outputs 0 if $(b' = d)$, indicating that $\mathcal{A}$ returned the identity $\mathcal{B}$ queried $\mathcal{O}\mathsf{Sim\text{-}or\text{-}Sign}$ with; thus σ_{m_b} is not a simulated signature. $\mathcal{B}$ outputs 1 otherwise. $\mathcal{B}'$ could be constructed similarly as follows. It receives $(\mathcal{U}, pp_{\mathsf{TPBS}_b}, pk^{TA}_{\mathsf{TPBS}_b}, sk^{TA}_{\mathsf{TPBS}_b}, pk^{Issuer}_{\mathsf{TPBS}_b}, sk^{Issuer}_{\mathsf{TPBS}_b})$ its challenger in the simulatability game in Fig. 4, chooses $d \overset{\$}{\leftarrow} \{0,1\}$, and runs $\mathcal{A}$ on $(pp_{\mathsf{TPBS}_b}, pk^{TA}_{\mathsf{TPBS}_b}, sk^{TA}_{\mathsf{TPBS}_b}, pk^{Issuer}_{\mathsf{TPBS}_b}, sk^{Issuer}_{\mathsf{TPBS}_b})$. Whenever $\mathcal{A}$ queries its challenge oracle $\mathcal{O}\mathsf{PLoRSign}$ with $(i, m, p_0, w_{p_0}, p_1, w_{p_1})$, if $\mathsf{PC}((p_0,m), w_{p_0}) = 0$ or $\mathsf{PC}((p_1,m), w_{p_1}) = 0$ or $i \notin \mathcal{U}$, $\mathcal{B}'$ returns $\perp$, otherwise it quires its challenger in the simulatability game with (i, m, p_d, w_{p_d}) and returns $(\sigma_{m_b}, sk^{p}_{\mathsf{TPBS}})$ to $\mathcal{A}$. When $\mathcal{A}$ outputs b', $\mathcal{B}'$ outputs 0 if $(b' = d)$ and 1 otherwise. In either case, if in $\mathbf{Exp}^{SIM}_{\mathcal{B},\mathsf{TPBS}}(\lambda)$ (resp. $\mathbf{Exp}^{SIM}_{\mathcal{B}',\mathsf{TPBS}}(\lambda)$) the challenger's bit is 0 indicating a signed signature, then $\mathcal{B}$ (resp. $\mathcal{B}'$) perfectly simulates $\mathbf{Exp}^{Anonymity}_{\mathcal{A},\mathsf{TPBS}}(\lambda)$ (resp. $\mathbf{Exp}^{Policy-privacy}_{\mathcal{A},\mathsf{TPBS}}(\lambda)$) for $\mathcal{A}$. However, if the bit is 1 indicating a simulated signature, then the bit d chosen by $\mathcal{B}$ (resp. $\mathcal{B}'$) has no relation to $\mathcal{A}$'s response. Hence, $\mathcal{B}$ outputs 1 with probability $\frac{1}{2}$. Therefore, the success probability of $\mathcal{B}$ (resp. $\mathcal{B}'$) is half that of A in the anonymity (resp. policy-privacy) experiment.

Theorem 2. *Given a zero-knowledge simulation-sound extractable NIZK system and an unlinkable RDS scheme, the traceable policy-based signature scheme in Fig. 8 is simulatable.*

Proof. Recall that for an adversary $\mathcal{A}$ to win the simulatability game $\mathbf{Exp}^{Sim}_{\mathcal{A},\mathsf{TPBS}}$ in Fig. 4, it has to guess the bit 'b' that $\mathcal{O}\mathsf{Sim\text{-}or\text{-}Sign}(.)$ is initialized with. In other words, $\mathcal{A}$ wins if it can determine whether the output signature σ_m of $\mathcal{O}\mathsf{Sim\text{-}or\text{-}Sign}(.)$ is generated using the secret keys or it is a simulated signature. By contradiction, We show that if there exists an adversary $\mathcal{A}$ that wins $\mathbf{Exp}^{Sim}_{\mathcal{A},\mathsf{TPBS}}$, we can construct an adversary $\mathcal{B}$ that wins the NIZK Zero-knowledge game $\mathbf{Exp}^{ZK}_{\mathcal{B},NIZK}$ of the underlying NIZK system in Fig. A.14 or $\mathcal{B}'$ that wins the unlinkability game $\mathbf{Exp}^{Unlinkability}_{\mathcal{B}',RDS}$ of the underlying RDS scheme in Fig. A.12, by linking the rerandomized version of σ^{i}_{ID} produced by the challenging oracle to a one that $\mathcal{B}'$ obtained from querying $\mathcal{O}\mathsf{USign}$ for the same identity i.

$\mathcal{B}$ is constructed as follows. $\mathcal{B}$ receives crs from its challenger in $\mathbf{Exp}^{ZK}_{\mathcal{B},NIZK}$ in Fig. A.14, generates the following: $pp_{RDS} \leftarrow \mathsf{ppGenRDS}(1^\lambda)$, $pp_{Sig} \leftarrow \mathsf{ppGenSig}(1^\lambda)$, and $pp_{Sig_\Sigma} \leftarrow \mathsf{ppGenSig}(1^\lambda)$. $\mathcal{B}$ sets $pp_{\mathsf{TPBS}} \leftarrow (crs, pp_{RDS}, pp_{Sig}, pp_{Sig_\Sigma})$. Then $\mathcal{B}$ runs $\mathsf{TASetup}(pp_{\mathsf{TPBS}})$ to obtain $(pk^{TA}_{\mathsf{TPBS}}, sk^{TA}_{\mathsf{TPBS}}, Reg)$, $\mathsf{IssuerSetup}(pp_{\mathsf{TPBS}})$ to obtain $(pk^{Issuer}_{\mathsf{TPBS}}, sk^{Issuer}_{\mathsf{TPBS}})$. For $i_j \in \mathcal{U}$, $\mathcal{B}$ runs the algorithms $\mathsf{UserKeyGen}(.)$ and $\mathsf{IDKeyGen}(.)$ to obtain $(sk^{i_j}_{\mathsf{TPBS}})$ and sets $\mathcal{Q}_i[i_j] = sk^{i_j}_{\mathsf{TPBS}} = (sk_{i_j})$. To simulate $\mathcal{A}$ calls to $\mathcal{O}\mathsf{KeyGen}$ oracle for $i \notin \mathcal{U}$, $\mathcal{B}$ runs the algorithms $\mathsf{UserKeyGen}(.)$ and $\mathsf{IDKeyGen}(.)$ to obtain (sk^{i}_{TPBS}) and the algorithm $\mathsf{PolicyKeyGen}(.)$ to obtain sk^{p}_{TPBS} and returns $(sk^{p}_{\mathsf{TPBS}}, sk^{i}_{\mathsf{TPBS}})$ to $\mathcal{A}$. To simulate $\mathcal{A}$ calls to $\mathcal{O}\mathsf{USign}$ oracle, $\mathcal{B}$ runs $\mathsf{PolicyKeyGen}(sk^{Issuer}_{\mathsf{TPBS}}, p)$ to obtain sk^{p}_{TPBS},and $\mathsf{Sign}(sk^{p}_{\mathsf{TPBS}}, \mathcal{Q}_i[i_j], m, p, w_p)$ to obtain σ_m. To simulate $\mathcal{A}$ calls to $\mathcal{O}\mathsf{Trace}$ oracle, $\mathcal{B}$ simply runs the Trace algorithm on Reg. To simulate $\mathcal{A}$ calls to $\mathcal{O}\mathsf{Sim\text{-}or\text{-}Sign}$ oracle, $\mathcal{B}$ parses $(sk_i, \sigma^i_{ID}) \leftarrow \mathcal{Q}_i[i][1]$ rerandomizes σ^i_{ID} to obtain ${\sigma'}^i_{ID}$ then forwards $(pk^{TA}_{\mathsf{TPBS}}, pk^{Issuer}_{\mathsf{TPBS}}, {\sigma'}^i_{ID}, m, p, sk_i, w_p)$ to $\mathcal{O}\mathsf{Sim\text{-}or\text{-}ProveNIZK}$ challenge oracle in $\mathbf{Exp}^{ZK}_{\mathcal{B},NIZK}(\lambda)$ to obtain π_m, then $\mathcal{B}'$ forwards $\sigma_m = ({\sigma'}^i_{ID}, \pi_m)$ to $\mathcal{A}$. Once $\mathcal{A}$ outputs b', $\mathcal{B}$ copies it to its challenger in $\mathbf{Exp}^{ZK}_{\mathcal{B},NIZK}(\lambda)$, and stops.

$\mathcal{B}'$ is constructed as follows. $\mathcal{B}'$ receives pp_{RDS} and pk_{RDS} from its challenger in the unlinkability game $\mathbf{Exp}^{Unlinkability}_{\mathcal{B},RDS}$ in Fig. A.12, generates the following; $(crs, tr_{NIZK}) \leftarrow \mathsf{SimSetupNIZK}(1^\lambda)$, $pp_{Sig} \leftarrow \mathsf{ppGenSig}(1^\lambda)$, and $pp_{Sig_\Sigma} \leftarrow \mathsf{ppGenSig}(1^\lambda)$. $\mathcal{B}'$ sets $pp_{\mathsf{TPBS}} \leftarrow (crs, pp_{RDS}, pp_{Sig}, pp_{Sig_\Sigma})$, $pk^{TA}_{\mathsf{TPBS}} \leftarrow pk_{RDS}$ and initializes an empty Reg. Then $\mathcal{B}'$ runs $\mathsf{IssuerSetup}(pp_{\mathsf{TPBS}})$ to obtain $(pk^{Issuer}_{\mathsf{TPBS}}, sk^{Issuer}_{\mathsf{TPBS}})$. For $j \in \mathcal{U}$, $\mathcal{B}'$ uses $\mathsf{UserKeyGen}(pp_{\mathsf{TPBS}})$ to generate $(pk_{i_j}, sk_{i_j}, ID_{i_j})$, sets $Reg[i_j] = ID_{i_j}$, parses $ID_{i_j} = ((c_{i_j}, \tilde{c}_{i_j}), \tau_{i_j})$, queries $\mathcal{O}\mathsf{CSignRDS}(c_{i_j})$ in $\mathbf{Exp}^{Unlinkability}_{\mathcal{B},RDS}$ to obtain $\sigma^{i_j}_{ID}$, and finally sets $\mathcal{Q}_i[i_j] = sk^{i_j}_{\mathsf{TPBS}} = (sk_{i_j}, \sigma^{i_j}_{ID})$. To simulate $\mathcal{A}$ calls to $\mathcal{O}\mathsf{KeyGen}$ oracle $i \notin \mathcal{U}$, $\mathcal{B}'$ uses $\mathsf{UserKeyGen}(pp_{\mathsf{TPBS}})$ to generate (pk_i, sk_i, ID_i), sets $Reg[ii] = ID_i$, parses $ID_i = ((c_i, \tilde{c}_i), \tau_i)$, queries $\mathcal{O}\mathsf{CSignRDS}(c_i)$ in $\mathbf{Exp}^{Unlinkability}_{\mathcal{B},RDS}$ to obtain σ^i_{ID}, sets $sk^i_{\mathsf{TPBS}} = (sk_i, \sigma^i_{ID})$, and finally runs $\mathsf{PolicyKeyGen}(sk^{Issuer}_{\mathsf{TPBS}}, p)$ to obtain sk^p_{TPBS}. To simulate $\mathcal{A}$ calls to $\mathcal{O}\mathsf{Trace}$ oracle, $\mathcal{B}$ simply runs the Trace algorithm on Reg. To simulate $\mathcal{A}$ calls to $\mathcal{O}\mathsf{Sim\text{-}or\text{-}Sign}$ oracle, $\mathcal{B}'$ invokes its RDS challenging oracle in $\mathbf{Exp}^{Unlinkability}_{\mathcal{B},RDS}$ on (i_j) to obtain σ'_{ID_b}, and $\pi_m \leftarrow \mathsf{SimProveNIZK}(crs, (pk^{TA}_{\mathsf{TPBS}}, \sigma'_{ID_b}, pk^{Issuer}_{\mathsf{TPBS}}, m), tr_{NIZK})$, sets $\sigma_m = (\sigma'_{ID_b}, \pi_m)$ and finally returns σ_m to $\mathcal{A}$. When $\mathcal{A}$ outputs b', $\mathcal{B}'$ copies it to its RDS unlinkability challenger and exits.

Theorem 3. *Given an unforgeable RDS scheme, an unforgeable signature scheme, and a simulation-sound extractable NIZK system, the traceable policy-based signature scheme in Fig. 8 is extractable.*

Proof. Recall that for adversary $\mathcal{A}$ to win the extractability game $\mathbf{Exp}^{Ext}_{\mathcal{A},\mathsf{TPBS}}$ in Fig. 5, it has to output a verifiable (m^*, σ_{m^*}) where m^* has never been queried to the $\mathcal{O}\mathsf{Sign}$ oracle and when the Extr algorithm is run over (m^*, σ_{m^*}) using the trapdoor information tr_{NIZK}, the returned $(p^*, sk^*_i, sk^{p^*}_{\mathsf{TPBS}}, w_{p^*})$ satisfies any of the following conditions: i) $sk^*_i \notin \mathcal{T}$, which implies that the adversary has never obtained some sk^{i}_{TPBS} through $\mathcal{O}\mathsf{KeyGen}$ where $sk_i = sk^*_i$, or ii) $p^* \notin \mathcal{L}$, which implies that the adversary has not queried $\mathcal{O}\mathsf{KeyGen}$ with p^* or iii) $\mathsf{PC}((p^*, m^*), w^*_p) = 0$. Thus we distinguish between three different types of adversaries that may win the extractability game in Fig. 5, i) $\mathcal{A}$ is of type-1 if $\mathsf{VerifyRDS}(pk^{TA}_{\mathsf{TPBS}}, sk^*_i, \sigma'^{i^*}_{ID}) = 1$ and for all $i \in \mathcal{T}$, $C^*_i \neq f(sk^*_i)$ ii) $\mathcal{A}$ is of type-2 if $\mathsf{VerifySig}(pk^{Issuer}_{\mathsf{TPBS}}, p, sk^{p}_{\mathsf{TPBS}}) = 1$ and $p^* \notin \mathcal{L}$ iii) $\mathcal{A}$ is of type-3 if $\mathsf{VerifyRDS}(pk^{TA}_{\mathsf{TPBS}}, sk^*_i, \sigma'^{i^*}_{ID}) = 0$ or $\mathsf{VerifySig}(pk^{Issuer}_{\mathsf{TPBS}}, p^*, sk^{p^*}_{\mathsf{TPBS}}) = 0$ or $\mathsf{PC}((p^*, m^*), w^*_p) = 0$. By contradiction, we show that if there exists an adversary $\mathcal{A}$ of type-1, type-2, and type-3 that wins $\mathbf{Exp}^{Ext}_{\mathcal{A},\mathsf{TPBS}}$, we can construct an adversary $\mathcal{B}$ that wins the EUF-CMA game $\mathbf{Exp}^{EUF-CMA}_{\mathcal{B},RDS}$ of the underlying RDS scheme in Fig. A.11, an adversary $\mathcal{B}'$ that wins the EUF-CMA game $\mathbf{Exp}^{EUF-CMA}_{\mathcal{B}',Sig}$ of the underlying signature scheme in Fig. A.17, and an adversary $\mathcal{B}''$ that wins the $\mathbf{Exp}^{Sim-Extr}_{\mathcal{B}'',NIZK}$ game of the underlying NIZK system in Fig. A.15, respectively.

For a type-1 $\mathcal{A}$ adversary, $\mathcal{B}$ is constructed as follows. $\mathcal{B}$ receives pp_{RDS}, and pk_{RDS} from its challenger in EUF-CMA game $\mathbf{Exp}^{EUF-CMA}_{\mathcal{B},RDS}$ in Fig. A.11, generates the following: $(crs, tr_{NIZK}) \leftarrow \mathsf{SimSetupNIZK}(1^\lambda)$, $pp_{Sig} \leftarrow \mathsf{ppGenSig}(1^\lambda)$, and $pp_{Sig_\Sigma} \leftarrow \mathsf{ppGenSig}(1^\lambda)$. Then $\mathcal{B}$ sets $pp_{\mathsf{TPBS}} \leftarrow (crs, pp_{RDS}, pp_{Sig}, pp_{Sig_\Sigma})$, $\mathcal{T} = \{\}$ and $pk^{TA}_{\mathsf{TPBS}} = pk_{RDS}$. Next, $\mathcal{B}$ runs $\mathsf{IssuerSetup}(pp_{\mathsf{TPBS}})$ to obtain $(pk^{Issuer}_{\mathsf{TPBS}}, sk^{Issuer}_{\mathsf{TPBS}})$. To simulate $\mathcal{A}$ calls to $\mathcal{O}\mathsf{KeyGen}$ oracle, $\mathcal{B}$ uses $\mathsf{UserKeyGen}(pp_{\mathsf{TPBS}})$ to generate (pk_i, sk_i, ID_i), queries $\mathcal{O}\mathsf{SignRDS}(sk_i)$ in $\mathbf{Exp}^{EUF-CMA}_{\mathcal{B},RDS}$ to obtain σ^i_{ID}, sets $sk^i_{\mathsf{TPBS}} = (sk_i, \sigma^i_{ID})$, Sets $\mathcal{T} = \mathcal{T} \cup \{i, sk_i\}$, runs $\mathsf{PolicyKeyGen}(sk^{Issuer}_{\mathsf{TPBS}}, p)$ to obtain sk^p_{TPBS} and finally returns $(sk^p_{\mathsf{TPBS}}, (sk^i_{\mathsf{TPBS}}))$ to $\mathcal{A}$. To simulate $\mathcal{A}$ calls to $\mathcal{O}\mathsf{Sign}$ oracle, if $i \notin \mathcal{T}$, $\mathcal{B}$ uses $\mathsf{UserKeyGen}(pp_{\mathsf{TPBS}})$ to generate (pk_i, sk_i, ID_i), queries $\mathcal{O}\mathsf{SignRDS}(sk_i)$ in $\mathbf{Exp}^{EUF-CMA}_{\mathcal{B},RDS}$ to obtain σ^i_{ID}, sets $sk^i_{\mathsf{TPBS}} = (sk_i, \sigma^i_{ID})$, runs $\mathsf{PolicyKeyGen}(sk^{Issuer}_{\mathsf{TPBS}}, p)$ to obtain sk^p_{TPBS}, and runs Sign directly over the message m using the generated keys. Once $\mathcal{A}$ outputs (m^*, σ_{m^*}), $\mathcal{B}$ parses $(\sigma'^{i^*}_{ID}, \pi^*_m) \leftarrow \sigma_m$, and runs $\mathsf{Extr}_{\mathsf{NIZK}}(tr_{NIZK}, pk^{Issuer}_{\mathsf{TPBS}}, pk^{TA}_{\mathsf{TPBS}}, m^*, \pi^*_m)$ to obtain $(p^*, sk^*_i, sk^p_{\mathsf{TPBS}}, w_{p^*})$, if $sk^*_i \notin \mathcal{T}$, it forwards $(sk^*_i, \sigma'^{i^*}_{ID})$ to its challenger in $\mathbf{Exp}^{EUF-CMA}_{\mathcal{B},RDS}$, which constitutes a forgery over sk^*_i.

For a type-2 $\mathcal{A}$ adversary, $\mathcal{B}'$ is constructed as follows. $\mathcal{B}'$ receives pp_{Sig}, and pk_{Sig} from its challenger in EUF-CMA game $\mathbf{Exp}^{EUF-CMA}_{\mathcal{B},Sig}$ Fig. A.17, generates the following: $(crs, tr_{NIZK}) \leftarrow \mathsf{SimSetupNIZK}(1^\lambda)$, $pp_{Sig_\Sigma} \leftarrow \mathsf{ppGenSig}(1^\lambda)$, and $pp_{RDS} \leftarrow \mathsf{ppGenRDS}(1^\lambda)$. $\mathcal{B}'$ sets $pp_{\mathsf{TPBS}} \leftarrow (crs, pp_{RDS}, pp_{Sig}, pp_{Sig_\Sigma})$, $\mathcal{L} = \{\}$, and $pk^{Issuer}_{\mathsf{TPBS}} \leftarrow pk_{Sig}$. Then $\mathcal{B}'$ runs $\mathsf{TASetup}(pp_{\mathsf{TPBS}})$ to obtain $(pk^{TA}_{\mathsf{TPBS}}, sk^{TA}_{\mathsf{TPBS}})$. To simulate $\mathcal{A}$ calls to $\mathcal{O}\mathsf{KeyGen}$ oracle, $\mathcal{B}'$ runs the algorithms $\mathsf{UserKeyGen}(.)$ and $\mathsf{IDKeyGen}(.)$ to obtain (sk^i_{TPBS}) and forwards p to $\mathcal{O}\mathsf{SignSig}(.)$ oracle in $\mathbf{Exp}^{EUF-CMA}_{\mathcal{B}',Sig}$ to obtain sk^p_{TPBS} and returns $(sk^p_{\mathsf{TPBS}}, (sk^i_{\mathsf{TPBS}}))$ to $\mathcal{A}$. To simulate $\mathcal{A}$ calls to $\mathcal{O}\mathsf{Sign}$ oracle, if $p \notin \mathcal{L}$, $\mathcal{B}'$ runs the algorithms $\mathsf{UserKeyGen}(.)$ and $\mathsf{IDKeyGen}(.)$ to obtain (sk^i_{TPBS}) and forwards p to $\mathcal{O}\mathsf{SignSig}(.)$ oracle in $\mathbf{Exp}^{EUF-CMA}_{\mathcal{B}',Sig}$ to obtain sk^p_{TPBS} and runs Sign directly over the message m using the generated keys. Once $\mathcal{A}$ outputs (m^*, σ_{m^*}), $\mathcal{B}'$ parses $(\sigma'^{i^*}_{ID}, \pi^*_m) \leftarrow \sigma_{m^*}$, and runs $\mathsf{Extr_{NIZK}}(tr_{NIZK}, pk^{Issuer}_{\mathsf{TPBS}}, pk^{TA}_{\mathsf{TPBS}}, m^*, \pi^*_m)$ to obtain $(p^*, sk^*_i, sk^{p^*}_{\mathsf{TPBS}}, w_{p^*})$, if $p \notin \mathcal{L}$, it forwards $(p^*, sk^{p^*}_{\mathsf{TPBS}})$ to its challenger in $\mathbf{Exp}^{EUF-CMA}_{\mathcal{B}',Sig}(\lambda)$, which constitutes a forgery over p^*.

For a type-3 $\mathcal{A}$ adversary, $\mathcal{B}''$ is constructed as follows. $\mathcal{B}''$ receives crs from its challenger in $\mathbf{Exp}^{Sim-Extr}_{\mathcal{B}'',NIZK}$ in Fig. A.15, generates the following: $pp_{Sig} \leftarrow \mathsf{ppGenSig}(1^\lambda)$, $pp_{Sig_\Sigma} \leftarrow \mathsf{ppGenSig}(1^\lambda)$, and $pp_{RDS} \leftarrow \mathsf{ppGenRDS}(1^\lambda)$. $\mathcal{B}''$ sets $pp_{\mathsf{TPBS}} \leftarrow (crs, pp_{RDS}, pp_{Sig}, pp_{Sig_\Sigma})$. Then $\mathcal{B}''$ runs $\mathsf{TASetup}(pp_{\mathsf{TPBS}})$ to obtain $(pk^{TA}_{\mathsf{TPBS}}, sk^{TA}_{\mathsf{TPBS}})$, $\mathsf{IssuerSetup}(pp_{\mathsf{TPBS}})$ to obtain $(pk^{Issuer}_{\mathsf{TPBS}}, sk^{Issuer}_{\mathsf{TPBS}})$. To simulate $\mathcal{A}$ calls to $\mathcal{O}\mathsf{KeyGen}$ oracle, $\mathcal{B}''$ runs the algorithms $\mathsf{UserKeyGen}(.)$ and $\mathsf{IDKeyGen}(.)$ to obtain (sk^i_{TPBS}) and the algorithm $\mathsf{PolicyKeyGen}(.)$ to obtain sk^p_{TPBS} and returns $(sk^p_{\mathsf{TPBS}}, sk^i_{\mathsf{TPBS}})$ to $\mathcal{A}$. To simulate $\mathcal{A}$ calls to $\mathcal{O}\mathsf{Sign}$ oracle, $\mathcal{B}''$ runs the algorithms $\mathsf{UserKeyGen}(.)$ and $\mathsf{IDKeyGen}(.)$ to obtain (sk^i_{TPBS}) and the algorithm $\mathsf{PolicyKeyGen}(.)$ to obtain sk^p_{TPBS}, parses $(sk_i, \sigma^i_{ID})(sk^i_{\mathsf{TPBS}})$, generates $\sigma'^i_{ID} \leftarrow \mathsf{RandomizeRDS}(\sigma^i_{ID})$, sends $((pk^{TA}_{\mathsf{TPBS}}, \sigma'^i_{ID}, pk^{Issuer}_{\mathsf{TPBS}}, m), (sk_i, p, sk^p_{\mathsf{TPBS}}, w_p))$ to $\mathcal{O}\mathsf{SimNIZK}$ oracle in $\mathbf{Exp}^{Sim-Extr}_{\mathcal{B}'',NIZK}$ to obtain π_m, then $\mathcal{B}''$ forwards $\sigma_m = (\sigma'^i_{ID}, \pi_m)$ to $\mathcal{A}$. Once $\mathcal{A}$ outputs (m^*, σ_{m^*}), $\mathcal{B}''$ parses $(\sigma'^{i^*}_{ID}, \pi^*_m) \leftarrow \sigma_m$, and sends $((pk^{TA}_{\mathsf{TPBS}}, \sigma'^{i^*}_{ID}, pk^{Issuer}_{\mathsf{TPBS}}, m), \pi^*_m)$ to its challenger in $\mathbf{Exp}^{Sim-Extr}_{\mathcal{B}'',NIZK}$, which constitutes a winning condition where the relation $\mathbb{R}'_{\mathbb{NP}}((pk^{TA}_{\mathsf{TPBS}}, \sigma'^i_{ID}, pk^{Issuer}_{\mathsf{TPBS}}, m^*), (sk^*_i, p, sk^{p*}_{\mathsf{TPBS}}, w_{p^*})) = 0$.

Theorem 4. *Given a one-way function, an unforgeable digital signature scheme, and a simulation-sound extractable NIZK system, the traceable policy-based signature scheme in Fig. 8 is non-frameable.*

Proof. Recall that for an adversary $\mathcal{A}$ to win $\mathbf{Exp}^{Non-Frameability}_{\mathcal{A},\mathsf{TPBS}}$ in Fig. 6; it has to output (m^*, σ_{m^*}) when it is traced back, the tracing result (i^*, π^*_{Trace}) points to some honest signer whose signing keys were not shared with $\mathcal{A}$ and yet such tracing result is accepted by Judge algorithm. Without loss of generality, let an adversary $\mathcal{A}$ win $\mathbf{Exp}^{Non-Frameability}_{\mathcal{A},\mathsf{TPBS}}$ for user identity $i_z \in \mathcal{U}$. Here we distinguish between two tracing results; i) $i^* = i_z$ and $ID_{i^*} = Reg[i_z]$, which implies that $\mathcal{A}$ has successfully revealed sk_{i_z} from an earlier signature by the user i_z, and ii) $i^* = i_z$ however $ID_{i^*} \neq Reg[i_z]$ that the honest user i_z has used in $\mathsf{IDKeyGen}(.)$ algorithm which implies $\mathcal{A}$ has manipulated Reg. Given a one-way function f(.), and an interactive perfect zero-knowledge proof of knowledge PoK, by contradiction, We show that if there exists an adversary $\mathcal{A}$ who wins $\mathbf{Exp}^{Non-Frameability}_{\mathcal{A},\mathsf{TPBS}}$, we can construct an adversary $\mathcal{B}$ who wins $\mathbf{Exp}^{ZK}_{\mathcal{B},NIZK}$ of the underlying NIZK scheme in Fig. A.14 or we can construct a successful adversary $\mathcal{B}'$ against $\mathbf{Exp}^{EUF-CMA}_{\mathcal{B}',Sig}$ of the underlying user digital signature scheme in Fig. A.17.

$\mathcal{B}$ is constructed as follows. $\mathcal{B}$ receives crs from its challenger in Fig. A.14. $\mathcal{B}$ generates $pp_{RDS} \leftarrow \mathsf{ppGenRDS}(1^\lambda)$, $(pp_{Sig}) \leftarrow \mathsf{ppGenSig}(1^\lambda)$, and $pp_{Sig_\Sigma} \leftarrow \mathsf{ppGenSig}(1^\lambda)$. Then $\mathcal{B}$ sets $pp_{\mathsf{TPBS}} \leftarrow (crs, pp_{RDS}, pp_{Sig}, pp_{Sig_\Sigma})$. $\mathcal{B}$ runs $\mathsf{TASetup}(pp_{\mathsf{TPBS}})$ to obtain $(pk^{TA}_{\mathsf{TPBS}}, sk^{TA}_{\mathsf{TPBS}}, Reg)$, and $\mathsf{IssuerSetup}(pp_{\mathsf{TPBS}})$ to obtain $(pk^{Issuer}_{\mathsf{TPBS}}, sk^{Issuer}_{\mathsf{TPBS}})$. For $j \in \mathcal{U}$, $\mathcal{B}$ uses $\mathsf{UserKeyGen}(pp_{\mathsf{TPBS}})$ to generate $(pk_{i_j}, sk_{i_j}, ID_{i_j})$, generates $((Reg[i_j]), (sk^{i_j}_{\mathsf{TPBS}})) \Leftrightarrow \mathsf{IDKeyGen}(.)$ and sets $\mathcal{Q}_i[i_j] \leftarrow sk^{i_j}_{\mathsf{TPBS}}$. To simulate $\mathcal{A}$ calls to $\mathcal{O}\mathsf{KeyGen}$ oracle for $i \notin \mathcal{U}$, $\mathcal{B}$ runs the algorithms $\mathsf{UserKeyGen}(.)$ and $\mathsf{IDKeyGen}(.)$ to obtain (sk^{i}_{TPBS}) and the algorithm $\mathsf{PolicyKeyGen}(.)$ to obtain sk^{p}_{TPBS} and returns $(sk^{p}_{\mathsf{TPBS}}, sk^{i}_{\mathsf{TPBS}})$ to $\mathcal{A}$. To simulate $\mathcal{A}$ calls to $\mathcal{O}\mathsf{USign}$ oracle, $\mathcal{B}$ runs $\mathsf{PolicyKeyGen}(.)$ to obtain sk^{p}_{TPBS}, For $j \in \mathcal{U} \wedge j \neq z$, $\mathcal{B}$ runs $\mathsf{Sign}(sk^{p}_{\mathsf{TPBS}}, \mathcal{Q}_i[i_j], m, p, w_p)$ to obtain σ_m. For user i_z, $\mathcal{B}$ parses $\sigma^{i}_{ID} \leftarrow \mathcal{Q}_i[z][1]$, calculates ${\sigma'}^{i}_{ID} \leftarrow \mathsf{RandomizeRDS}(\sigma^{i}_{ID})$, and sends $(pk^{TA}_{\mathsf{TPBS}}, {\sigma'}^{i}_{ID}, pk^{Issuer}_{\mathsf{TPBS}}, m), (sk_i, p, sk^{p}_{\mathsf{TPBS}}, w_p))$ to its challenging oracle $\mathsf{OSim\text{-}or\text{-}ProveNIZK(.)}$ in Fig. A.14 to obtain π_m and finally outputs $\sigma_m = ({\sigma'}^{i}_{ID}, \pi_m)$ to $\mathcal{A}$. To simulate $\mathcal{A}$ calls to $\mathcal{O}\mathsf{Trace}$ oracle, $\mathcal{B}$ simply runs the Trace algorithm on Reg since all identity keys are generated using the $\mathsf{IDKeyGen}(.)$ algorithm. When $\mathcal{A}$ outputs (m^*, σ_{m^*}), which passes the winning conditions defined in the experiment Fig. 6, $\mathcal{B}$ runs $(i^*_z, \pi^*_{Trace}) \leftarrow \mathsf{Trace}(Reg, m^*, \sigma_{m^*})$, if $ID_{i^*} = ID_{i_z}$, it outputs 0 to its challenger oracle in Fig. A.14 and 1 otherwise.

On the other hand, $\mathcal{B}'$ could be constructed as follows. $pp_{Sig_\Sigma}, pk_{Sig_\Sigma}$ from its challenger in Fig. A.17. $\mathcal{B}'$ sets $\mathcal{D}[i_z] \leftarrow pk_{Sig_\Sigma}$ and generates $(crs) \leftarrow \mathsf{SetupNIZK}(1^\lambda)$, $pp_{RDS} \leftarrow \mathsf{ppGenRDS}(1^\lambda)$, and $pp_{Sig} \leftarrow \mathsf{ppGenSig}(1^\lambda)$. $\mathcal{B}'$ sets $pp_{\mathsf{TPBS}} \leftarrow (crs, pp_{RDS}, pp_{Sig}, pp_{Sig_\Sigma})$. Then $\mathcal{B}'$ runs $\mathsf{TASetup}(pp_{\mathsf{TPBS}})$ to obtain $(pk^{TA}_{\mathsf{TPBS}}, sk^{TA}_{\mathsf{TPBS}}, Reg)$, and $\mathsf{IssuerSetup}(pp_{\mathsf{TPBS}})$ to obtain $(pk^{Issuer}_{\mathsf{TPBS}}, sk^{Issuer}_{\mathsf{TPBS}})$. For $j \in \mathcal{U} \wedge j \neq z$, $\mathcal{B}'$ uses $\mathsf{UserKeyGen}(pp_{\mathsf{TPBS}})$ to generate $(pk_{i_j}, sk_{i_j}, ID_{i_j})$, generates $((Reg[i_j]), (sk^{i_j}_{\mathsf{TPBS}})) \Leftrightarrow \mathsf{IDKeyGen}(.)$ and sets $\mathcal{Q}_i[i_j] \leftarrow sk^{i_j}_{\mathsf{TPBS}}$

For i_z, $\mathcal{B}'$ picks $sk_{i_z} \xleftarrow{\$} \mathbb{Z}^*_p$, calculates $C_{i_z} = (c_{i_z}, \tilde{c}_{i_z}) \xleftarrow{\$} f(sk_{i_j})$, sends c_{i_z} to its challenger signing oracle $\mathcal{O}\mathsf{SignSig}$ in Fig. A.17 to obtain τ_{i_z}, constructs ID_{i_z} and generates $\sigma^{i_z}_{ID} \leftarrow \mathsf{SignComRDS}(sk^{TA}_{RDS}, c_{i_z})$ and sets $\mathcal{Q}_i[i_z] \leftarrow sk^{i_z}_{\mathsf{TPBS}} = (sk_{i_z}, \sigma^{i_z}_{ID})$ and $Reg[i_z] = ID_{i_z}$. To simulate $\mathcal{A}$ calls to $\mathcal{O}\mathsf{USign}$ oracle, $\mathcal{B}'$ runs $\mathsf{PolicyKeyGen}(.)$ to obtain sk^{p}_{TPBS}, and runs $\mathsf{Sign}(sk^{p}_{\mathsf{TPBS}}, \mathcal{Q}_i[i_j], m, p, w_p)$ to obtain σ_m. To simulate $\mathcal{A}$ calls to $\mathcal{O}\mathsf{Trace}$ oracle, $\mathcal{B}'$ simply run the Trace algorithm on Reg. When $\mathcal{A}$ outputs (m^*, σ_{m^*}), which passes the winning conditions defined in the experiment Fig. 6, $\mathcal{B}'$ runs $(i^*_z, \pi^*_{Trace}) \leftarrow \mathsf{Trace}(Reg, m^*, \sigma_{m^*})$, parses $\{C^*_{i_z}, \tau^*_{i_z}\} \leftarrow ID^*_{i_z}$ and sends $\{c^*_{i_z}, \tau^*_{i_z}\}$ to its challenger oracle in Fig. A.17, which constitutes a winning condition since $c^*_{i_z} \neq c_{i_z}$ such that $ID_{i_z} = (c_{i_z}, \tilde{c}_{i_z}, \tau_{i_z})$.

Theorem 5. *Given an unforgeable RDS scheme and a simulation-sound extractable NIZK system, the traceable policy-based signature scheme in Fig. 8 is traceable.*

Proof. Recall that for an adversary to win $\mathbf{Exp}^{Traceability}_{\mathcal{A},\mathsf{TPBS}}$ in Fig. 7; it has to output (m^*, σ_{m^*}) where the produced signature cannot be traced to some signer whose signing keys are obtained through $\mathcal{O}\mathsf{KeyGen}(.)$ oracle. In other words, for adversary $\mathcal{A}$ to win, it should have access to a verifiable σ^{i}_{ID} under pk^{TA}_{TPBS} that has been obtained without calling the $\mathcal{O}\mathsf{KeyGen}(.)$ oracle or adversary $\mathcal{A}$ has succeeded in generating NIZK proof for a false statement such that $\mathsf{VerifyRDS}(pk^{TA}_{\mathsf{TPBS}}, sk^*_i, {\sigma'}^{i^*}_{ID}) = 0$. Thus we distinguish between two different types of adversaries that may win the traceability experiment in Fig. 7, i) $\mathcal{A}$ is of type-1 if $\mathsf{VerifyRDS}(pk^{TA}_{\mathsf{TPBS}}, sk^*_i, {\sigma'}^{i^*}_{ID}) = 1$ and $Reg[i] = \emptyset$ ii) $\mathcal{A}$ is of type-2 if $\mathsf{VerifyRDS}(pk^{TA}_{\mathsf{TPBS}}, sk^*_i, {\sigma'}^{i^*}_{ID}) = 0$. We show by contradiction that if an adversary $\mathcal{A}$ of type-1 or type-2 wins in $\mathbf{Exp}^{Traceability}_{\mathcal{A},\mathsf{TPBS}}$, we can construct an adversary $\mathcal{B}$ that wins the $\mathbf{Exp}^{EUF-CMA}_{\mathcal{B},RDS}$ in Fig. A.11 or an adversary $\mathcal{B}'$ that wins the $\mathbf{Exp}^{Sim-Extr}_{\mathcal{B}',NIZK}$ in Fig. A.15, respectively.

For $\mathcal{A}$ of type-1, $\mathcal{B}$ is constructed as follows, $\mathcal{B}$ receives pp_{RDS}, and pk_{RDS} from its challenger in $\mathbf{Exp}^{EUF-CMA}_{\mathcal{B},RDS}$ Fig. A.11, generates the following: $(crs, tr_{NIZK}) \leftarrow \mathsf{SimSetupNIZK}(1^\lambda)$, $pp_{Sig} \leftarrow \mathsf{ppGenSig}(1^\lambda)$, and $pp_{Sig_\Sigma} \leftarrow \mathsf{ppGenSig}(1^\lambda)$. $\mathcal{B}$ sets $pp_{\mathsf{TPBS}} \leftarrow (crs, pp_{RDS}, pp_{Sig}, pp_{Sig_\Sigma})$, and $pk^{TA}_{\mathsf{TPBS}} \leftarrow pk_{RDS}$ and initializes an empty Reg. Then $\mathcal{B}$ runs $\mathsf{IssuerSetup}(pp_{\mathsf{TPBS}})$ to obtain $(pk^{Issuer}_{\mathsf{TPBS}}, sk^{Issuer}_{\mathsf{TPBS}})$. To simulate $\mathcal{A}$ calls to $\mathcal{O}\mathsf{KeyGen}$ oracle, $\mathcal{B}$ uses $\mathsf{UserKeyGen}(pp_{\mathsf{TPBS}})$ to generate (pk_i, sk_i, ID_i), queries $\mathcal{O}\mathsf{SignRDS}(sk_i)$ in $\mathbf{Exp}^{EUF-CMA}_{\mathcal{B},RDS}$ to obtain σ^i_{ID}, sets $sk^i_{\mathsf{TPBS}} = (sk_i, \sigma^i_{ID})$, $Reg[i] = ID_i$, runs $\mathsf{PolicyKeyGen}(sk^{Issuer}_{\mathsf{TPBS}}, p)$ to obtain sk^p_{TPBS} and finally returns $(sk^p_{\mathsf{TPBS}}, (sk^i_{\mathsf{TPBS}}))$ to $\mathcal{A}$. To simulate $\mathcal{A}$ calls to the $\mathcal{O}\mathsf{Trace}$ oracle, $\mathcal{B}$ simply runs the Trace algorithm on Reg. Once $\mathcal{A}$ outputs (m^*, σ_{m^*}), $\mathcal{B}$ parses $({\sigma'}^{i^*}_{ID}, \pi^*_m) \leftarrow \sigma_{m^*}$, and runs $\mathsf{Extr}_{\mathsf{NIZK}}(tr_{NIZK}, pk^{Issuer}_{\mathsf{TPBS}}, pk^{TA}_{\mathsf{TPBS}}, {\sigma'}^{i^*}_{ID}, m^*, \pi^*_m)$ to obtain $(p^*, sk^*_i, sk^p_{\mathsf{TPBS}}, w_{p^*})$, and forwards $(sk^*_i, {\sigma'}^{i^*}_{ID})$ to its challenger in $\mathbf{Exp}^{EUF-CMA}_{\mathcal{B},RDS}$, which constitutes a forgery over sk^*_i.

For a type-2 $\mathcal{A}$ adversary, $\mathcal{B}'$ is constructed as follows. $\mathcal{B}'$ receives crs from its challenger in $\mathbf{Exp}^{Sim-Extr}_{\mathcal{B}',NIZK}$ in Fig. A.15, generates the following: $pp_{Sig} \leftarrow \mathsf{ppGenSig}(1^\lambda)$, $pp_{Sig_\Sigma} \leftarrow \mathsf{ppGenSig}(1^\lambda)$, and $pp_{RDS} \leftarrow \mathsf{ppGenRDS}(1^\lambda)$. $\mathcal{B}'$ sets $pp_{\mathsf{TPBS}} \leftarrow (crs, pp_{RDS}, pp_{Sig}, pp_{Sig_{Sigma}})$. Then $\mathcal{B}'$ runs $\mathsf{TASetup}(pp_{\mathsf{TPBS}})$ to obtain $(pk^{TA}_{\mathsf{TPBS}}, (sk^{TA}_{\mathsf{TPBS}}, Reg))$, $\mathsf{IssuerSetup}(pp_{\mathsf{TPBS}})$ to obtain $(pk^{Issuer}_{\mathsf{TPBS}}, sk^{Issuer}_{\mathsf{TPBS}})$. To simulate $\mathcal{A}$ calls to $\mathcal{O}\mathsf{KeyGen}$ oracle, $\mathcal{B}'$ runs the algorithms $\mathsf{UserKeyGen}(.)$ and $\mathsf{IDKeyGen}(.)$ to obtain (sk^i_{TPBS}) and the algorithm $\mathsf{PolicyKeyGen}(.)$ to obtain sk^p_{TPBS} and returns $(sk^p_{\mathsf{TPBS}}, sk^i_{\mathsf{TPBS}})$ to $\mathcal{A}$. To simulate $\mathcal{A}$ calls to $\mathcal{O}\mathsf{Trace}$ oracle, $\mathcal{B}'$ simply run the Trace algorithm on Reg. Once $\mathcal{A}$ outputs (m^*, σ_{m^*}), $\mathcal{B}'$ parses $({\sigma'}^{i^*}_{ID}, \pi^*_m) \leftarrow \sigma_{m^*}$, and sends $(m, {\sigma'}^{i^*}_{ID}, \pi^*_m)$ to its challenger in $\mathbf{Exp}^{Sim-Extr}_{\mathcal{B}',NIZK}$ which constitutes a winning condition for statement (1a) in relation $\mathbb{R}'_{\mathbb{NP}}$ that is defined in (1).

6 TPBS Instantiation and performance

We instantiate TPBS with Pointcheval-Sanders (PS) RDS Scheme [16, 17][1] in Appendix B.1 because of its short signature size and low signing cost in addition to its ability to sign a hiding commitment over a message using a special form of its signing algorithm. we consider the One-way function $f(.)$ over a type-3 bilinear group map defined by $(p, \mathbb{G}, \tilde{\mathbb{G}}, \mathbb{G}_T, e)$ where the SDH assumption holds to be simply the function $f(sk_i) = (g^{sk_i}, \tilde{g}^{sk_i})/$ for $(g, \tilde{g}) \in (\mathbb{G}, \tilde{\mathbb{G}})$ and $sk_i \in \mathbb{Z}^*_p$. We instantiate the issuer digital signature algorithm with the structure-preserving signature scheme in Appendix B.2 of Abe *et al.* [2]. we instantiate the SE-NIZK scheme with the Groth-Sahai proof system [13]. Moreover, any

[1] PS scheme has two variants one is based interactive assumption to prove its security [16] and a slightly modified one [17] where its security is proved based on the SDH assumption both could be used to instantiate our scheme

digital signature scheme can be utilized for the user's digital signature algorithm. However, we keep it as a black box since it is not utilized in TPBS signature generation or verification. Finally, we instantiate the PoK with the four-move perfect zero-knowledge protocol of Cramer *et al.* [10]. Furthermore, we keep the original definition of Bellare and Fuchsbauer for a policy p that defines a set of Pairing Product Equations (PPEs) $(E_1, \ldots, E_n)$, such that the policy checker $\mathsf{PC}((p,m), w_p) = 1$ iff $E_j((p,m), w_p) = 1$ for all $j \in [n]$.

$\underline{\mathsf{ppGen}}$. for a security parameter λ, let $(p, \mathbb{G}, \tilde{\mathbb{G}}, \mathbb{G}_T, e, g, \tilde{g})$ defines a type-3 bilinear group map that is generated by $(g, \tilde{g})$ that is used by all the scheme algorithms, Run $pp_{Sig} \leftarrow \mathsf{ppGenAbe}(1^\lambda)$, $pp_{Sig_\Sigma} \leftarrow \mathsf{ppGenSig}(1^\lambda)$, $pp_{RDS} \leftarrow \mathsf{ppGenPS}(1^\lambda)$, and $crs \leftarrow \mathsf{SetupGS}$. Set $pp_{\mathsf{TPBS}} = \{crs, pp_{RDS}, pp_{Sig}, pp_{Sig_\Sigma}\}$, where pp_{TPBS} becomes an implicit input for all TPBS algorithms.

$\underline{\mathsf{TASetup}}$. $(pk^{TA}_{\mathsf{TPBS}}, sk^{TA}_{\mathsf{TPBS}}) \leftarrow \mathsf{KeyGenPS}(pp_{RDS})$ such that $pk^{TA}_{\mathsf{TPBS}} = (g_1, \tilde{A}, \tilde{B})$, $sk^{TA}_{\mathsf{TPBS}} = (a, b)$. Setup an empty $Reg = [\,]$.

$\underline{\mathsf{IssuerSetup}}$. $(pk^{Issuer}_{\mathsf{TPBS}}, sk^{Issuer}_{\mathsf{TPBS}}) \leftarrow \mathsf{KeyGenAbe}(pp_{Abe})$ such that $pk^{Issuer}_{\mathsf{TPBS}} = (U, V, H, Z)$, and $sk^{Issuer}_{\mathsf{TPBS}} = (u, v, h, z)$ for $U \in \mathbb{G}$, $(V, H, Z) \in \tilde{\mathbb{G}}$ and $(u, v, h, z) \in \mathbb{Z}^*_p$

$\underline{\mathsf{UserKeyGen}}$. Generates $(pk^i_{Sig_\Sigma}, sk^i_{Sig_\Sigma}) \leftarrow \mathsf{KeyGenSig}(pp_{Sig_\Sigma})$, sets $\mathcal{D}[i] = (pk^i_{Sig_\Sigma})$, picks $sk_i \overset{\$}{\leftarrow} \mathbb{Z}^*_p$, calculates $C_i = (c_i, \tilde{c}_i) = (g_1^{sk_i}, \tilde{B}^{sk_i})$, generates $\tau_i \leftarrow \mathsf{SignSig}(c_i, sk^i_{Sig_\Sigma})$, sets $ID_i = \{C_i, \tau_i\}$, finally return (pk_i, sk_i, ID_i).

$\underline{\mathsf{IDKeyGen}}$. The user sends (i, ID_i) to the TA, the TA parses ID_i as $\{(c_i, \tilde{c}_i), \tau_i\}$ and obtains an authentic copy of $pk^i_{Sig_\Sigma}$, if $Reg[i] = \emptyset \wedge \mathsf{VerifySig}(pk^i_{Sig_\Sigma}, c_i, \tau_i)$ $\wedge$ $e(c_i, \tilde{B}) = e(g_1, \tilde{c}_i)$, the TA engages with the user to start the interactive zero-knowledge protocol $\mathsf{PoK}(sk_i : c_i = g_1^{sk_i})$, if TA verifies that the user knows sk_i such that the relation of PoK holds, the TA generates $\sigma^i_{ID} \leftarrow \mathsf{SignComPS}(sk^{TA}_{\mathsf{TPBS}}, c_i)$ as follows, the TA picks $r \overset{\$}{\leftarrow} \mathbb{Z}^*_p$ and generates $\sigma^i_{ID} = (\sigma^i_{ID_1}, \sigma^i_{ID_2}) \leftarrow (g_1^r, (g_1^a(c_i)^b)^r$, finally the TA sets $Reg[i] = ID_i$ and the user set his scheme identity key as $sk^i_{\mathsf{TPBS}} = (sk_i, \sigma^i_{ID})$.

$\underline{\mathsf{PolicyKeyGen}}$. For policy $p \in \{0,1\}^*$, which is presented by a set of PPE equations $(E_1, \ldots, E_n)$ for a number of secret group elements $(M, \tilde{N}) \in \mathbb{G}^{k_M} \times \tilde{\mathbb{G}}^{k_N}$, the issuer generates $sk^p_{\mathsf{TPBS}} \leftarrow \mathsf{SignAbe}(sk^{Issuer}_{\mathsf{TPBS}}, (M, \tilde{N}))$ such that $sk^p_{\mathsf{TPBS}} = (R, S, T)$.

$\underline{\mathsf{Sign}}$. To sign a message m, the signer first generates a rerandomized version of σ^i_{ID}, $\sigma'^i_{ID} \leftarrow \mathsf{RandomizePS}(\sigma^i_{ID})$, along with a SE-NIZK proof π_m for relation $\mathbb{R}'_{\mathbb{NP}}$ that is defined in 1 as follows

$$((pk^{TA}_{\mathsf{TPBS}}, \sigma'^i_{ID}, pk^{Issuer}_{\mathsf{TPBS}}, m), (sk_i, p, sk^p_{\mathsf{TPBS}}, w_p)) \in \mathbb{R}'_{\mathbb{NP}} \Leftrightarrow$$
$$e(\sigma'_{ID1}, \tilde{A})e(\sigma'_{ID1}, \tilde{K}) = e(\sigma'_{ID2}, \tilde{g}) \wedge e(g, \tilde{K}) = e(g^{sk_i}, \tilde{B}) \quad (1a)$$
$$\wedge\, e(R, V)e(S, \tilde{g})e(M, H) = e(g, Z) \wedge e(R, T)e(U, N) = e(g, \tilde{g}) \quad (1b)$$
$$\wedge\, E_j(((M, \tilde{N}), m), (W_p, \tilde{W}_p)) = 1 \;\forall j \in [n] \quad (1c)$$

<u>Verify</u>. To verify a message signature pair (m, σ_m), the verifier parses $({\sigma'}^{i}_{ID}, \pi_m)$ from σ_m and runs $\mathsf{VerifyNIZK}(crs, (pk^{TA}_{\mathsf{TPBS}}, {\sigma'}^{i}_{ID}, pk^{Issuer}_{\mathsf{TPBS}}, m), \pi_m)$. Finally, the verifier outputs $\top$ in case of verification success and $\perp$ otherwise.

<u>Trace</u>. To trace a message signature pair (m, σ_m) to its original signer, the TA verifies such pair. If the verification succeeds, it parses $({\sigma'}^{i}_{ID}, \pi_m)$ from σ_m and exhaustively searches Reg for a matching i as follows.

```
foreach $\tilde{c}_i \in Reg$
    if $e(\sigma'_{ID_2}, \tilde{g}) e(\sigma'_{ID_1}, \tilde{A})^{-1} = e(\sigma'_{ID_1}, \tilde{c}_i)$
    return $(i, ID_i)$
$\pi \leftarrow \mathsf{ProveNIZK}(crs, (c_i, {\sigma'}^{i}_{ID}), \tilde{c}_i) \ni e(\sigma'_{ID_2}, \tilde{g}) e(\sigma'_{ID_1}, \tilde{A})^{-1} = e(\sigma'_{ID_1}, \tilde{c}_i)$
                                   $\wedge\, e(c_i, \tilde{B}) = e(g_1, \tilde{c}_i)$
$\pi_{Trace} \leftarrow (c_i, \tau_i, \pi)$
return $(i, \pi_{Trace})$
```

<u>Judge</u>. After verifying (m, σ_m), parses (c_i, τ_i, π) from π_{Trace} and outputs $\top$ if $\mathsf{VerifySig}(pk^{i}_{Sig_\Sigma}, c_i, \tau_i) \wedge \mathsf{VerifyNIZK}(crs, (c_i, {\sigma'}^{i}_{ID}), \pi))$ or $\perp$ otherwise.

Performance Analysis. To sign a message, the signer needs to perform a rerandomization for an RDS signature and calculate a SE-NIZK proof for relation 1. To verify a message signature pair, the verifier verifies a SE-NIZK proof for relation 1. Tracing a signature to its origin requires verifying the message signature pair, exhaustively searching the registry for matching registration information, and calculating a SE-NIZK proof for the search result. Finally, to verify the output of the tracing algorithm, the Judge procedure verifies the message signature pair, the user's digital signature on the registration information, and the proof generated by the tracing algorithm. Concretely, let the TPBS be initialized with n users and the policy p be expressed in 1 PPE uniquely defined by $(M, \tilde{N}) \in \mathbb{G} \times \tilde{\mathbb{G}}$ group elements. To sign a message m that conforms to p, The proposed instantiation produces a total signature size of 14 elements in $\mathbb{G}$ + 16 elements in $\tilde{\mathbb{G}}$, where ${\sigma'}^{i}_{ID}$ is a PS signature of size 2 elements in $\mathbb{G}$, and π_m is a Groth-Sahai proof of knowledge of size 12 elements in $\mathbb{G}$ + 16 elements $\tilde{\mathbb{G}}$. Signing costs two exponentiations in $\mathbb{G}$ to generate ${\sigma'}^{i}_{ID}$ and approximately 40 exponentiations in $\mathbb{G}$ + 70 exponentiations in $\tilde{\mathbb{G}}$ to produce π_m. Verifying a given TPBS message signature pair costs approximately a total of 100 pairing operations to verify π_m[2]. For tracing a signature, the TA performs at most $3n$ pairing operations and produces a proof π of size 16 group elements in $\tilde{\mathbb{G}}$, which costs around 10 exponentiations in $\mathbb{G}$ and 20 exponentiations in $\tilde{\mathbb{G}}$. To verify the output of the tracing algorithm, the Judge performs around 40 pairing operations to verify π in addition to the verification cost of the TPBS signature and the verification cost of the signature τ_i of the user on the registration information.

[2] The verification cost of Groth-Sahai proofs could be enhanced using batch verification [6]

7 Conclusion

We have proposed TPBS, a traceable policy-based signature scheme that supports delegatability. Our scheme fills the gap in the original policy-based schemes by linking a signature to the identity of its original signer, thus holding the signer of a specific message accountable for the produced signature. We have analyzed the security of TPBS and proved that it is an anonymous, policy-private, unforgeable, traceable, and non-frameable signature scheme. Moreover, we provided a concrete instantiation of TPBS using the Pointcheval-Sanders rerandomizable signature scheme, the structure-preserving signature scheme of Abe *et al.*, and the Groth-Sahai NIZK system and analyzed its efficiency.

References

1. Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structure-preserving signatures and commitments to group elements. In: Annual Cryptology Conference. pp. 209–236. Springer (2010)
2. Abe, M., Groth, J., Haralambiev, K., Ohkubo, M.: Optimal structure-preserving signatures in asymmetric bilinear groups. In: Annual Cryptology Conference. pp. 649–666. Springer (2011)
3. Bellare, M., Fuchsbauer, G.: Policy-based signatures. In: International Workshop on Public Key Cryptography. pp. 520–537. Springer (2014)
4. 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)
5. Bichsel, P., Camenisch, J., Neven, G., Smart, N.P., Warinschi, B.: Get shorty via group signatures without encryption. In: International Conference on Security and Cryptography for Networks. pp. 381–398. Springer (2010)
6. 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)
7. Boneh, D., Shacham, H.: Group signatures with verifier-local revocation. In: Proceedings of the 11th ACM conference on Computer and communications security. pp. 168–177 (2004)
8. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: International conference on the theory and applications of cryptographic techniques. pp. 93–118. Springer (2001)
9. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Annual international cryptology conference. pp. 56–72. Springer (2004)
10. Cramer, R., Damgård, I., MacKenzie, P.: Efficient zero-knowledge proofs of knowledge without intractability assumptions. In: International Workshop on Public Key Cryptography. pp. 354–372. Springer (2000)
11. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discrete Applied Mathematics **156**(16), 3113–3121 (2008)
12. Groth, J.: Simulation-sound nizk proofs for a practical language and constant size group signatures. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 444–459. Springer (2006)
13. Groth, J., Sahai, A.: Efficient noninteractive proof systems for bilinear groups. SIAM Journal on Computing **41**(5), 1193–1232 (2012)

14. Kiltz, E., Mityagin, A., Panjwani, S., Raghavan, B.: Append-only signatures. In: International Colloquium on Automata, Languages, and Programming. pp. 434–445. Springer (2005)
15. 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
16. Pointcheval, D., Sanders, O.: Short randomizable signatures. In: Cryptographers' Track at the RSA Conference. pp. 111–126. Springer (2016)
17. Pointcheval, D., Sanders, O.: Reassessing security of randomizable signatures. In: Cryptographers' Track at the RSA Conference. pp. 319–338. Springer (2018)
18. Xu, Y., Safavi-Naini, R., Nguyen, K., Wang, H.: Traceable policy-based signatures and instantiation from lattices. Information Sciences **607**, 1286–1310 (2022). https://doi.org/https://doi.org/10.1016/j.ins.2022.06.031, https://www.sciencedirect.com/science/article/pii/S0020025522006211
19. 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

A Building Blocks Security Definitions

A.1 RDS Schemes Security

In what follows, we give the formal definitions of the security properties of RDS schemes that are required to prove the security of TPBS.

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.10), 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 [16].

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

```
$\mathcal{O}$CSignRDS($C$)
$j = j + 1$
$(\sigma_{RDS}) \leftarrow$ SignComRDS($sk_{RDS}, C$)
$\mathcal{C}[j][0] = C$;    $\mathcal{C}[j][1] = \sigma_{RDS}$
return $\sigma_{RDS}$
```

```
$\mathcal{O}$LoRRDS($j$)
$(\sigma'_{RDS_0}) \leftarrow$ RandomizeRDS($\mathcal{C}[j][1]$)
$m \xleftarrow{\$} \mathbb{Z}_p^*$
$(\sigma_{RDS_1}) \leftarrow$ SignRDS($sk_{RDS}, m$)
  return $(\sigma'_{RDS_b})$
return 0
```

Fig. A.10: RDS security experiments oracles

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.11.*

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

Fig. A.11: RDS EUF-CMA experiment.

Unlinkability. Unlinkability of RDS schemes is the infeasibility of linking a rerandomized version of a signature over a message m to the original signature it has been created from if one does not explicitly know m [17]. Unlinkability is formally defined by the experiment in Fig. A.12, where an adversary $\mathcal{A}$ is given access $\mathcal{O}\mathsf{CSign}(.)$ and $\mathcal{O}\mathsf{LoRRDS}(.)$ oracles, which are defined in Fig. A.10. the adversary $\mathcal{A}$ constructs a list $\mathcal{C}$ of RDS signatures over some commitments C it does not know their corresponding openings using $\mathcal{O}\mathsf{CSign}(.)$ then asks $\mathcal{O}\mathsf{LoRRDS}(.)$ to rerandomize one of two signatures in $\mathcal{C}$ of its choice. The oracle $\mathcal{O}\mathsf{LoRRDS}(.)$ 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 signature and outputs σ'_{RDS_b}. The adversary wins if it can determine which signature is used in the rerandomization process with probability better than the random guess [19].

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.12.*

```
$\mathbf{Exp}^{Unlinkability}_{\mathcal{A},RDS}(\lambda)$
$pp_{RDS} \leftarrow \mathsf{ppGenRDS}(1^{\lambda})$
$(pk_{RDS}, sk_{RDS}) \leftarrow \mathsf{KeyGenRDS}(pp_{RDS})$
$j \leftarrow 0; \quad \mathcal{C} = [\,]$
$b \xleftarrow{\$} \{0,1\}$
$a \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{CSignRDS}(.),\mathcal{O}\mathsf{LoRRDS}(.,b)}(pk_{RDS})$
if $a = b$
    return 1
return 0
```

Fig. A.12: RDS unlinkability experiment.

A.2 SE-NIZK Schemes Security

In what follows, we give the formal definitions of the security properties of SE-NIZK schemes that are required for proving the security of TPBS.

Zero knowledge. This security notion implies that given access to a prove oracle $\mathcal{O}\mathsf{Sim\text{-}or\text{-}ProveNIZK}$ (see Fig. A.13), it is hard for adversary $\mathcal{A}$ to distinguish between a proof for a statement x using a witness w from a simulated one.

```
$\mathcal{O}\mathsf{Sim\text{-}or\text{-}ProveNIZK}(x, w)$
$\pi_{NIZK_0} \leftarrow \mathsf{ProveNIZK}(crs, x, w)$
if $\mathbb{R}(x, w) = 1$
    $\pi_{NIZK_1} \leftarrow \mathsf{SimProveNIZK}(crs, x, tr_{NIZK})$
        else return $\perp$
return $\pi_{NIZK_b}$
```

```
$\mathcal{O}\mathsf{SimNIZK}(\mathsf{x})$
$\pi_{NIZK} \leftarrow \mathsf{SimProveNIZK}(crs, x, tr_{NIZK})$
$\mathcal{M} = \mathcal{M} \cup (x, \pi_{NIZK})$
return $\pi_{NIZK}$
```

Fig. A.13: NIZK system security experiments oracles

Definition 10. *(NIZK Zero-knowledge) The NIZK system is zero-knowledge if for any PPT adversary $\mathcal{A}$, $|\Pr[\boldsymbol{Exp}^{ZK}_{\mathcal{A},NIZK}(\lambda) = 1] - \frac{1}{2}| \leq \epsilon(\lambda)$, where the zero-knowledge experiment is defined in Fig. A.14.*

```
$Exp^{ZK}_{\mathcal{A},NIZK}(\lambda)$
$crs_0 \leftarrow \mathsf{SetupNIZK}(1^\lambda)$
$(crs_1, tr_{NIZK}) \leftarrow \mathsf{SimSetupNIZK}(1^\lambda)$
$b \xleftarrow{\$} \{0,1\}$
$a \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{Sim\text{-}or\text{-}ProveNIZK}(.,\mathsf{b},\mathsf{tr_{NIZK}})}(crs_b)$
if $a = b$
  return 1
return 0
```

Fig. A.14: NIZK system zero-knowledge experiment.

Simulation-extractability. This security notion implies that given access to a simulated prove oracle $\mathcal{O}\mathsf{SimNIZK}$ (see Fig. A.13), it is hard for adversary $\mathcal{A}$ to output a verifiable proof for a statement x using a witness w such that $\mathbb{R}(x, w) = 0$.

Definition 11. *(NIZK Simulation-extractability) The NIZK system is simulation-extractable if the for any PPT adversary $\mathcal{A}$, $\mathsf{Pr}[Exp^{Sim-Extr}_{\mathcal{A},NIZK}(\lambda) = 1] \leq \epsilon(\lambda)$, where the NIZK simulation-extractability experiment is defined in Fig. A.15.*

```
$Exp^{Sim-Extr}_{\mathcal{A},NIZK}(\lambda)$
$(crs, tr_{NIZK}) \leftarrow \mathsf{SimSetupNIZK}(1^\lambda)$
$\mathcal{M} = \{\}$
$(x^*, \pi^*_{NIZK}) \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{SimNIZK}(.,\mathsf{tr_{NIZK}})}(crs)$
if $(x^*, \pi^*_{NIZK}) \notin \mathcal{M} \wedge \mathsf{VerifyNIZK}(crs, x^*, \pi^*_{NIZK})$
  $w \leftarrow Extr(tr_{NIZK}, x^*, \pi^*_{NIZK})$
  if $\mathbb{R}(x, w) = 0$
    return 1
return 0
```

Fig. A.15: NIZK system simulation-extractability experiment.

A.3 Digital Signature Schemes Security

In what follows, we give the formal definition of Existential Unforgeability under Chosen Message Attack (EUF-CMA) of digital signature schemes that are required for proving the security of TPBS.

Existential Unforgeability under Chosen Message Attack (EUF-CMA). This security notion implies that given access to a signing oracle $\mathcal{O}\mathsf{SignSig}$ (see Fig. A.16), 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^*_{Sig}) for which m^* was never queried to the signing oracle.

```
$\mathcal{O}\mathsf{SignSig}(m)$
$\sigma_{Sig} \leftarrow \mathsf{SignSig}(sk_{Sig}, m)$
$\mathcal{M} = \mathcal{M} \cup \{m, \sigma_{Sig}\}$
return $\sigma_{Sig}$
```

Fig. A.16: Digital signature security experiments oracles

Definition 12. *(Digital signature scheme EUF-CMA) The digital signature scheme is EUF-CMA secure, if the for any PPT adversary $\mathcal{A}$,* $\Pr[\boldsymbol{Exp}^{EUF-CMA}_{\mathcal{A},Sig}(\lambda) = 1] \leq \epsilon(\lambda)$*, where the EUF-CMA experiment is defined in Fig. A.17.*

```
$\mathbf{Exp}^{EUF-CMA}_{\mathcal{A},Sig}(\lambda)$
$\mathcal{M} = \{\}$
$pp_{Sig} \leftarrow \mathsf{ppGenSig}(1^\lambda)$
$(pk_{Sig}, sk_{Sig}) \leftarrow \mathsf{KeyGenSig}(pp_{Sig})$
$(m^*, \sigma^*_{Sig}) \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{SignSig}(.)}(pk_{Sig})$
if $(m^*, \sigma^*_{Sig}) \notin \mathcal{M}$
  return $\mathsf{VerifySig}(pk_{Sig}, m^*, \sigma^*_{Sig})$
```

Fig. A.17: Digital signature scheme EUF-CMA experiment.

B Protocols used in instantiating TPBS

B.1 Pointcheval-Sanders (PS) RDS Scheme

PS is a pairing-based RDS scheme that enables the produced signature to be rerandomized and still be verifiable using the verification keys of the signer [16]. It also allows signing a commitment on a hidden message such that the resulting signature is verifiable for the message itself. The PS scheme specifies the following six procedures.

- ppGenPS. The algorithm outputs the public parameters of the scheme such that $pp_{PS} = (p, \mathbb{G}, \tilde{\mathbb{G}}, \mathbb{G}_T, e)$ where $(p, \mathbb{G}, \tilde{\mathbb{G}}, \mathbb{G}_T, e)$ defines a type-3 bilinear group map [11].

- KeyGenPS. This procedure returns the signer's secret and public key pair, the signer picks $g_1 \overset{\$}{\leftarrow} \mathbb{G}$, $\tilde{g}_1 \overset{\$}{\leftarrow} \tilde{\mathbb{G}}$ and $(a, b) \overset{\$}{\leftarrow} \mathbb{Z}^*_p$, sets $sk^{signer}_{PS} = (a, b)$ and computes $(\tilde{A}, \tilde{B}) \leftarrow (\tilde{g}^a, \tilde{g}^b)$, sets $pk^{signer}_{PS} = (g_1, \tilde{g}_1, \tilde{A}, \tilde{B})$.

- SignPS. This algorithm outputs the digital signature σ_{PS} for a message $m \in \mathbb{Z}_p$ by randomly choosing $h \overset{\$}{\leftarrow} \mathbb{G}_1 \backslash \{1_{\mathbb{G}1}\}$ and sets $\sigma_{PS} = (\sigma_{PS1}, \sigma_{PS2}) \leftarrow (h, h^{(a+b.m)})$.

- RandomizePS. This algorithm rerandomizes the digital signature on a message m and outputs σ'_{PS} by randomly choosing $r' \xleftarrow{\$} \mathbb{G}_1 \backslash \{1_{\mathbb{G}1}\}$ and computing $\sigma'_{PS} \leftarrow (\sigma^{r'}{}_{PS1}, \sigma^{r'}{}_{PS2}) \leftarrow (h^{r'}, h^{r'(a+b.m)})$.

- VerifyPS. This algorithm verifies the signature σ_{PS} over m by verifying $e(\sigma_{PS1}, \tilde{A}\tilde{B}^m) = e(\sigma_{PS2}, \tilde{g}_1)$ and outputs $\{\top, \bot\}$.

- SignComPS: This a special form of SignPS algorithm that allows the signer to generate a signature over a message m by only knowing a commitment of the message g_1^m by randomly choosing $r \xleftarrow{\$} \mathbb{Z}_p^*$ and computing $\sigma_{PS} \leftarrow (\sigma_{PS1}, \sigma_{PS2}) \leftarrow (g_1^r, (g_1^a(g_1^m)^b)^r$, $\sigma_{PS} \leftarrow \mathsf{SignComPS}(sk_{PS}^{Signer}, g_1^m)$.

B.2 Abe *et al.* Optimal Structure-Preserving Signatures

Abe *et al.* demonstrated that the lower bounds for a structure-preserving signature scheme to protect against random message attack as i) it must use at least two pairing product equations to verify a signature, and ii) the signature size must be at least 3 group elements [2]. Moreover, they presented a structure-preserving signature scheme that matches such lower bounds to sign a pair of group elements $(M, N) \in \mathbb{G} \times \tilde{\mathbb{G}}$ as follows:

- ppGenAbe. This algorithm outputs the public parameters of the system such that $pp_{Abe} = (p, \mathbb{G}, \tilde{\mathbb{G}}, \mathbb{G}_T, e, g, \tilde{g})$ where $(p, \mathbb{G}, \tilde{\mathbb{G}}, \mathbb{G}_T, e)$ defines a type-3 bilinear group map [11] which is generated by $(g, \tilde{g})$.

- KeyGenAbe. picks $u, v, h, z \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $U = g^u$, $V = \tilde{g}^v$, $H = \tilde{g}^h$, $Z = \tilde{g}^z$ $(pk_{Abe}, sk_{Abe}) = ((U, V, H, Z), (u, v, w, z))$.

- SignAbe. On input of a message m in the form of $(M, N) \in \mathbb{G} \times \tilde{\mathbb{G}}$, the signer picks $r \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $R = g^r$, $S =^{z-rv} .M^{-w}$, $T = (\tilde{g}.N^{-u})^{\frac{1}{r}}$ and outputs $\sigma_{Abe} = (R, S, T)$.

- VerifyAbe. This algorithm accepts σ_{Abe} over m if $M, R, S \in \mathbb{G}$ and $N, T \in \tilde{\mathbb{G}}$ and

$$e(R, V)e(S, \tilde{g})e(M, H) = e(g, Z) \wedge e(R, T)e(U, N) = e(g, \tilde{g})$$

Finally, the authors showed how the above scheme could be extended to sign messages in $\mathbb{G}^{k_M} \times \tilde{\mathbb{G}}^{k_N}$

B.3 Groth-Sahai Zero-knowledge Proof System

In what follows, we introduce Groth-Sahai (GS) pairing-based Non-interactive zero-knowledge system as one of the TPBS building blocks. Groth-Sahai enables a prover to convince a verifier that a set of variables $X_i \in \mathbb{G}_1, \tilde{Y_j} \in \mathbb{G}_2, x_i, y_j \in \mathbb{Z}_p$ simultaneously satisfy a set of equations [13]. Groth and Sahai presented four general equations that can be used to represent the statement to be proved.

Pairing-product equation. For known $A_j \in \mathbb{G}_1, \tilde{B_i} \in \mathbb{G}_2$ and $\gamma_{ij} \in \mathbb{Z}_p$

$$\prod_{j=1}^{n} e(A_j, \tilde{Y_j}) \prod_{i=1}^{m} e(X_i, \tilde{B_i}) \prod_{i=1}^{m}\prod_{j=1}^{n} e(X_i, \tilde{Y_j})^{\gamma_{ij}} = 1$$

Multi-exponentiation equation in $\mathbb{G}_1$. For known $A_j, T \in \mathbb{G}_1$ and $b_i, \gamma_{ij} \in \mathbb{Z}_p$ (Can be written for $\mathbb{G}_2$ as well)

$$\prod_{j=1}^{n} A_j^{y_j} \prod_{i=1}^{m} X_i^{b_i} \prod_{i=1}^{m} \prod_{j=1}^{n} X_i^{y_j \gamma_{ij}} = T$$

Quadratic Equation. For known $a_j, b_i \gamma_{ij}, t \in \mathbb{Z}_p$

$$\sum_{j=1}^{n} a_j y_j \sum_{i=1}^{m} x_i b_i \sum_{i=1}^{m} \sum_{j=1}^{n} x_i \gamma_{ij} y_j = t$$

The Groth-Sahai NIZK system is defined by the following three procedures:

SetupGS. The algorithm outputs the Common Reference String (CRS) parameters of the system either in hiding or binding setting, $CRS_{GS} \leftarrow \mathsf{CRSGenGS}(1^\lambda)$.

ProveGS. The prover uses this algorithm to generate the proof elements π_{GS} and θ_{GS}. $\{\pi_{GS}, \theta_{GS}\} \leftarrow \mathsf{ProveGS}(CRS_{GS}, X_i, \tilde{Y}_j, x_i, y_j, C, D)$ where (C, D) are the commitments of the secret variables $X_i \in \mathbb{G}_1$, $\tilde{Y}_j \in \mathbb{G}_2$, and $x_i, y_j \in \mathbb{Z}_p$ either in a hiding or a binding setting.

VerifyGS. The verifier uses this algorithm to verify the proof elements π and θ satisfy the prover statement and outputs $\{\top, \bot\} \leftarrow \mathsf{VerifyGS}(CRS_{GS}, C_{GS}, D_{GS}, \pi_{GS}, \theta_{GS})$.

C PBS Scheme

PBS scheme is a tuple of four polynomial-time algorithms, $\mathsf{PBS} = \{\mathsf{SetupPBS}, \mathsf{KeyGenPBS}, \mathsf{SignPBS}, \mathsf{VerifyPBS}\}$, which are defined as follows.

SetupPBS. This algorithm outputs the public parameters of the scheme and the issuer's master secret key pair, $(pp_{PBS}, msk_{PBS}) \leftarrow \mathsf{SetupPBS}(1^\lambda)$.

KeyGenPBS. This procedure generates the signer's secret key for a specific policy $p \in \{0,1\}^*$, $sk_{PBS}^{Signer} \leftarrow \mathsf{KeyGenPBS}(pp_{PBS}, msk_{PBS}, p)$.

SignPBS. On input of a message m, a witness $w_p \in \{0,1\}^*$ that m conforms a specific policy p, and the secret signing key sk_{PBS}^{Signer}, this procedure generates a signature σ_{PBS} over m, $\sigma_{PBS} \leftarrow \mathsf{SignPBS}(pp_{PBS}, sk_{PBS}^{Signer}, m, w_p)$.

VerifyPBS. This algorithm verifies the signature σ_{PBS} over m, $\{\top, \bot\} \leftarrow \mathsf{VerifyPBS}(pp_{PBS}, m, \sigma_{PBS})$. Bellare and Fuchsbauer have presented an efficient construction for the PBS scheme by relying on Simulation-Extractable (SE) NIZK proof system [12, 13] and eliminating the need for IND-CPA-secure public-key encryption scheme since the used NIZK is extractable.

The generated signature is a SE-NIZK for a policy checker PC defined in the following $\mathbb{NP}$-relation whose statements are of form $X = (pk_{PBS}, m)$ with witnesses $W = (p, sk_{PBS}^{Singer}, w_p)$ and,

$$((pk_{PBS}, m), (p, sk_{PBS}^{Singer}, w_p)) \in \mathbb{R}_{\mathbb{NP}} \Leftrightarrow$$

$$\mathsf{VerifySig}(pk_{PBS}, p, sk_{PBS}^{Singer}) = 1 \tag{2a}$$

$$\wedge\ ((p, m), w_p) \in \mathsf{PC} \tag{2b}$$

PBS based on SE-NIZK is instantiated using the Groth-Sahai simulation-extractable NIZK proof system [13] and structure-preserving digital signature scheme [1]. In such instantiation, policies are defined as one or more pairing product equations (PPEs), which must be satisfied by the message and the witness. We refer the reader to the complete PBS paper [3] for the complete instantiation procedures.

```
$\mathsf{SetupPBS}(1^\lambda)$
-----------------------------------------
$crs \leftarrow \mathsf{SetupNIZK}(1^\lambda)$
$pp_{Sig} \leftarrow \mathsf{ppGenSig}(1^\lambda)$
$(pk_{PBS}, msk_{PBS}) \leftarrow \mathsf{KeyGenSig}(pp_{Sig})$
$\mathbf{return}\ pp_{PBS} \leftarrow (crs, pp_{Sig}, pk_{PBS}), msk_{PBS}$

$\mathsf{KeyGenPBS}(pp_{PBS}, msk_{PBS}, p)$
-----------------------------------------
$sk_{PBS}^{Singer} \leftarrow \mathsf{SignSig}(pp_{PBS}, p, msk_{PBS})$
$\mathbf{return}\ (pp_{PBS}, p, sk_{PBS}^{Singer})$

$\mathsf{SignPBS}(pp_{PBS}, p, m, sk_{PBS}^{Singer}, w_p)$
-----------------------------------------
$\sigma_{PBS} \leftarrow \mathsf{ProveNIZK}(crs, (pk_{PBS}, m), (p, sk_{PBS}^{Singer}, w_p))$
$\mathbf{return}\ \sigma_{PBS}$

$\mathsf{VerifyPBS}(pp_{PBS}, m, \sigma_{PBS})$
-----------------------------------------
$\mathbf{return}\ \mathsf{VerifyNIZK}(crs, (pk_{PBS}, m), \sigma_{PBS})$
```

Fig. C.18: TPBS PBS based on SE-NIZK

Delegatable PBS

PBS allows signing rights delegation, in which a signer who holds a key for some policy can delegate such a key to another signer with possible restriction of the associated policy. To achieve this, keys are associated with a vector of policies $(p_1, ..., p_n)$ such that a signed message m satisfies all such policies, i.e., $\mathsf{PC}((p_i, m), w_{pi}) = 1\ \forall i \in [n]$. In such a construction, the signature scheme used by the issuer to sign a policy vector $\mathbf{p}$ is replaced by an append-only signature scheme [14]. Accordingly, the KeyGenPBS algorithm is replaced by a new algorithm DelegatePBS that issues a signing key for a new policy vector. Where the issuer runs DelegatePBS with $(msk_{PBS}, \mathbf{p})$ the same way as KeyGenPBS to produce sk_{DPBS}^{Signer}, however, a signer can delegate such key and further restrict its policy by p' using DelegatePBS on the input of the signing key sk_{PBS}^{Signer} for a policy vector $\mathbf{p} = (p_1, ..., p_n)$ and a policy p' which outputs a signing key sk'_{PBS} that allows signing a message m if it satisfies the policies $p_1, ..., p_n$ and p'.

- $\mathsf{DelegatePBS}$ on the input of the signing key sk_{PBS}^{Signer} for a policy vector $\mathbf{p} = (p_1, ..., p_n)$ and a policy p' which outputs a signing key sk'_{PBS} that allows signing a message m if it satisfies the policies $p_1, ..., p_n$ and p'.