Stronger Security Notions for Decentralized Traceable Attribute-Based Signatures and More Efficient Constructions

Essam Ghadafi

University of Bristol, United Kingdom

Abstract. Traceable attribute-based signatures extend standard attribute-based signatures by granting a designated tracing authority the power to revoke the anonymity of signatures by revealing who signed them. Such a feature is important in deterring abuse and enforcing accountability.
In this work, we revisit the notion of Decentralized Traceable Attribute-Based Signatures (DTABS) introduced by El Kaafarani et al. (CT-RSA 2014) and improve the state-of-the-art in two directions: Firstly, we provide a new stronger security model which circumvents some shortcomings in existing models. Our model minimizes the trust placed in attribute authorities and hence provides, among other things, a stronger definition for non-frameability. In addition, unlike previous models, our model captures the notion of tracing soundness which ensures that even if all parties in the system are fully corrupt, no one but the user who produced the signature could claim authorship of the signature. Secondly, we provide a generic construction that is secure w.r.t. our strong security model and show two example instantiations in the standard model which are much more efficient than existing constructions (secure under weaker security definitions).

Keywords. Attribute-based signatures, security definitions, traceability, standard model.

1 Introduction

Attribute-based cryptography provides a versatile solution for designing role-based cryptosystems. In attribute-based cryptosystems, the private operation, e.g. decryption/signing, is performed w.r.t. a security policy. Only users possessing attributes satisfying the policy can perform the operation. Goyal et al. [17], inspired by the work of Sahai and Waters [33], put forward the first attribute-based cryptosystems.

In Attribute-Based Signatures (ABS) [28, 29], messages are signed w.r.t. signing policies expressed as predicates. A signature convinces the verifier that it it was produced by a user with attributes satisfying the associated signing policy revealing neither the identity of the user nor the attributes used. Attribute-based signatures have many applications, including trust negotiation, e.g. [12], attribute-based messaging, e.g. [8], and leaking secrets. For more details and comparison with related notions refer to [29, 32].

The security of attribute-based signatures [28] requires user's privacy and unforgeability. Informally, user's privacy (i.e. anonymity), requires that signatures reveal neither the user's identity nor the attributes used in the signing. On the other hand, unforgeability requires that a user cannot forge a signature w.r.t. a signing predicate that her attributes do not satisfy, even if she colludes with other users.

Traceable Attribute-Based Signatures (TABS) [11] extend standard attribute-based signatures by adding an anonymity revocation mechanism which allows a tracing authority to recover the identity of the user who produced the signature. Such a feature is of vital importance in enforcing accountability and deterring abuse.

Related Work. Various constructions of attribute-based signatures exist in the literature [26, 35, 25, 29, 31, 32, 19, 13]. Those constructions vary in terms of the expressiveness of the policies they support and whether they offer selective or adaptive security. Adaptively secure schemes supporting more expressive policies are preferable since they cover a larger scale of potential applications.

While there exist constructions supporting threshold policies with constant-size signatures, e.g. [19, 13], constructions supporting monotonic/non-monotonic policies, e.g. [29, 31, 32], yield signatures that are linearly dependent on the number of attributes in the policy or the systems' security parameter.

Supporting multiple attribute authorities was first considered by [28, 31]. However, the multi–authority setting still had the problem of requiring a central trusted authority. Furthermore, in some cases, the security of the entire system is compromised if the central authority is corrupted. Okamoto and Takashima [32] recently proposed the first fully decentralized construction.

Escala et al. [11] added the traceability feature to standard ABS schemes and proposed a model for the single attribute authority setting. More recently, El Kaafarani et al. [10] proposed a security model and two generic constructions for decentralized traceable attribute-based signatures. They also provided instantiations without random oracles [3]. Besides correctness, the recent model of [10] defines three security requirements: anonymity, full unforgeability and traceability. Informally, anonymity requires that a signature reveals neither the identity of the signer nor the set of attributes used the signing; full unforgeability requires that users cannot forge signatures w.r.t. signing policies their individual attributes do not satisfy even if they collude, which also captures non-frameability; and traceability requires that the tracing authority is always able to establish the identity of the signer and prove such a claim.

We end by noting that there exist other weaker variants of traceable attribute-based signatures suiting specific applications. For instance, [22] proposed the notion of attribute-based group signatures which attaches public attributes (i.e. signatures do not hide the attributes used in the signing) to standard group signatures. Also, [23] proposed a traceable attribute-based signature scheme where the signing policy is determined beforehand by the verifier and hence requiring interaction in the signing protocol.

Shortcomings in Existing Models. The unforgeability/non-frameability requirements in all existing models for traceable attribute-based signatures [11, 10] (and even those for standard (i.e. non-traceable) attribute-based signatures, e.g. [28, 31, 32]) besides placing full trust in attribute authorities, assume the existence of secure means for the delivery of the secret attributes' keys from attribute authorities to users. More specifically, learning the key for any attribute a user owns allows framing the user w.r.t. to those attributes. For instance, the non-frameability definition in the single-authority model of [11] relies on the assumption that the attribute authority is fully honest, whereas the full unforgeability definition (also capturing non-frameability) in the stronger and more recent model of [10] assumes that at least one attribute authority is fully honest.

While this is not an issue in standard attribute-based signatures (since signatures are perfectly anonymous and hence it is infeasible for any party to identify the signer), we emphasize that this could be a serious limitation in the traceable setting. In particular, the innocence of users could be jeopardized by being falsely accused of producing signatures they have not produced. A misbehaving attribute authority or any party intercepting the secret attributes' keys is capable of signing on behalf of the user w.r.t. any predicate satisfied by the compromised subset of attributes.

We believe that the overly strong assumptions upon which the unforgeability/non-frameability notions in existing models rely is the result of the absence of the assignment of personal keys to the users. Moreover, the absence of users' personal keys further complicates the constructions and degrades the efficiency. For instance, the recent constructions in [10], similarly to [29], rely on the so-called pseudo-attribute technique in order to bind the signature to the message: the user proves that she either owns attributes satisfying the signing predicate or she has a special signature on the message and the encoding of the signing predicate that verifies w.r.t. some trapdoor verification key.

Another shortcoming of existing models for traceable attribute-based signatures is the absence of the tracing soundness requirement which was defined recently in the context of traditional group signatures [34]. This requirement ensures that a valid signature can only trace to a single user even if all entities in the system are fully corrupt. Tracing soundness is very vital for many applications, for example, applications where users might get rewarded for signatures they have produced or where abusing signing rights might result in legal consequences.

Our Contribution. We first rectify the aforementioned shortcomings in existing models by presenting a stronger security model for the primitive. Our model is for the interesting dynamic and fully decentralized setting in which attributes' management is distributed among different authorities who may not even be aware of one another, and where users and attribute authorities can join the system at any time. Our model offers a stronger security definition for non-frameability which circumvents the limitations inherent in existing models. In addition, our model provides a cleaner definition for traceability, and unlike previous models, it captures the useful notion of tracing soundness [34].

Our second contribution is a generic construction for the primitive which permits expressive signing policies and meets strong adaptive security requirements.

Finally, we provide two example instantiations of the generic framework in the standard model. Besides offering stronger security, our instantiations are more efficient than existing constructions.

Paper Organization. In Section 2, we give some preliminary definitions. We present our model in Section 3. We list the building blocks we use in Section 4. In Section 5, we present our generic construction and prove its security. In Section 6, we present instantiations in the standard model and compare their efficiency to existing constructions.

Notation. A function $\nu(.) : \mathbb{N} \rightarrow \mathbb{R}^+$ is negligible in c if for every polynomial $p(.)$ and all sufficiently large values of c, it holds that $\nu(c) < \frac{1}{p(c)}$. Given a probability distribution S, we denote by $y \leftarrow S$ the operation of selecting an element according to S. If A is a probabilistic machine, we denote by $A(x_1, \ldots, x_n)$ the output distribution of A on inputs $(x_1, \ldots, x_n)$. By PPT we mean running in probabilistic polynomial time in the relevant security parameter.

2 Preliminaries

In this section we provide some preliminary definitions.

Bilinear Groups. Let $\mathbb{G}_1 := \langle G \rangle$, $\mathbb{G}_2 := \langle \tilde{G} \rangle$ and $\mathbb{G}_T$ be groups of a prime order p. A bilinear group is a tuple $\mathcal{P} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, G, \tilde{G}, e)$ where $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ is a non-degenerate bilinear map. We will use multiplicative notation for all the groups and let $\mathbb{G}_1^\times := \mathbb{G}_1 \setminus \{1_{\mathbb{G}_1}\}$ and $\mathbb{G}_2^\times := \mathbb{G}_2 \setminus \{1_{\mathbb{G}_2}\}$. We will accent elements from $\mathbb{G}_2$ with ˜ for the sake of clarity. We will be working with Type-3 groups [14] where $\mathbb{G}_1 \neq \mathbb{G}_2$ and there is no efficient isomorphism between the groups in either direction. We assume the existence of an algorithm $\mathsf{BGrpSetup}$ which on input a security parameter λ outputs a description of bilinear groups.

Complexity Assumptions. We will use the following assumptions from the literature:

SXDH. The Decisional Diffie-Hellman (DDH) assumption holds in both groups $\mathbb{G}_1$ and $\mathbb{G}_2$.

SDLIN in $\mathbb{G}_1$ [21] [1]**.** This is a variant of the DLIN assumption [9] and states in Type-3 bilinear groups, given the tuple $(G, G^f, G^h, G^{rf}, G^{sh}, G^t, \tilde{G}, \tilde{G}^t) \in \mathbb{G}_1^6 \times \mathbb{G}_2^2$ for unknown $f, h, r, s, t \in \mathbb{Z}_p$, it is hard to tell whether $t = r + s$ or t is random.

q**-SDH** [7]**.** Given $(G, G^x, \ldots, G^{x^q}, \tilde{G}, \tilde{G}^x)$ for $x \leftarrow \mathbb{Z}_p$, it is hard to output a pair $(c, G^{\frac{1}{x+c}}) \in \mathbb{Z}_p \times \mathbb{G}_1$ for an arbitrary $c \in \mathbb{Z}_p \backslash \{-x\}$.

q**-AGHO** [2]**.** Given a uniformly random tuple $(G, \tilde{G}, \tilde{W}, \tilde{X}, \tilde{Y}) \in \mathbb{G}_1 \times \mathbb{G}_2^4$, and q uniformly random tuples $(A_i, B_i, R_i, \tilde{D}_i) \in \mathbb{G}_1^3 \times \mathbb{G}_2$, each satisfying

$$e(A_i, \tilde{D}_i) = e(G, \tilde{G}) \quad \text{and} \quad e(G, \tilde{X}) = e(A_i, \tilde{W})e(B_i, \tilde{G})e(R_i, \tilde{Y}),$$

it is hard to output a new tuple $(A^*, B^*, R^*, \tilde{D}^*)$ satisfying the above equations and different from the q tuples.

Span Programs. For a field $\mathbb{F}$ and a variable set $\mathcal{A} = \{\alpha_1, \ldots, \alpha_n\}$, a monotone span program [20] is defined by a $\beta \times \gamma$ matrix $\mathbf{S}$ (over $\mathbb{F}$) along with a labeling map ρ which associates each row in $\mathbf{S}$ with an element $\alpha_i \in \mathcal{A}$.

The span program accepts a set $\mathcal{A}'$ iff $\mathbf{1} \in \text{Span}(\mathbf{S}_{\mathcal{A}'})$, where $\mathbf{S}_{\mathcal{A}'}$ is the sub-matrix of $\mathbf{S}$ containing only rows with labels $\alpha_i \in \mathcal{A}'$, i.e., the program only accepts $\mathcal{A}'$ if there exists a vector $\boldsymbol{z}$ s.t. $\boldsymbol{z}\mathbf{S}_{\mathcal{A}'} = [1, 0, \ldots, 0]$.

[1] It can similarly be defined in $\mathbb{G}_2$.

3 Syntax and Security of Decentralized Traceable Attribute-Based Signatures

A DTABS scheme involves the following entities: a set of attribute authorities, each with a unique identity aid and a pair of secret/verification keys $(\mathsf{ask}_{\mathsf{aid}}, \mathsf{avk}_{\mathsf{aid}})$; a tracing authority TA with a secret tracing key tk that is used to identify the signer of a given signature; a set of users, each with a unique identity uid, a personal secret/public key pair $(\mathbf{usk}[\mathsf{uid}], \mathbf{uvk}[\mathsf{uid}])$ and a set of attributes $\mathcal{A} \subseteq \mathbb{A}$ (where $\mathbb{A}$ is the attribute universe). Attributes in the system can be distinctly identified by concatenating the identity of the managing authority with the name of the attribute. This way, the identities (and hence the public keys) of attribute authorities managing attributes appearing in the signing policy can be inferred from the predicate itself which eliminates the need for any additional meta-data to be attached. In our model, attribute authorities as well as users can join the system at any time.

A DTABS scheme is a tuple of polynomial-time algorithms

$$\mathcal{DTABS} := (\mathsf{Setup}, \mathsf{AKeyGen}, \mathsf{UKeyGen}, \mathsf{AttKeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Trace}, \mathsf{Judge})\cdot$$

The definition of the algorithms are as follows; where to aid notation all algorithms bar the first three are assumed to take as implicit input the public parameters pp output by algorithm Setup.

- $\mathsf{Setup}(1^\lambda)$ is run by some trusted third party. On input a security parameter 1^λ, it outputs public parameters pp and a tracing key tk.
- $\mathsf{AKeyGen}(\mathsf{pp}, \mathsf{aid})$ is run by attribute authority aid to generate its key pair $(\mathsf{ask}_{\mathsf{aid}}, \mathsf{avk}_{\mathsf{aid}})$. The attribute authority publishes its public key $\mathsf{avk}_{\mathsf{aid}}$.
- $\mathsf{UKeyGen}(\mathsf{pp})$ on input the public parameters pp, it outputs a personal secret/verification key pair $(\mathbf{usk}[\mathsf{uid}], \mathbf{uvk}[\mathsf{uid}])$ for the user with identity uid. We assume that the public key table $\mathbf{uvk}$ is publicly available (possibly via some PKI) so that anyone can obtain authentic copies of uers' public keys.
- $\mathsf{AttKeyGen}(\mathsf{ask}_{\mathsf{aid}(\alpha)}, \mathsf{uid}, \mathbf{uvk}[\mathsf{uid}], \alpha)$ on input the secret key of the attribute authority managing attribute α (i.e. $\mathsf{ask}_{\mathsf{aid}(\alpha)}$), a user's identity uid, a user's personal public key $\mathbf{uvk}[\mathsf{uid}]$ and an attribute $\alpha \in \mathbb{A}$, it outputs a secret key $\mathsf{sk}_{\mathsf{uid},\alpha}$ for attribute α for the user. The key $\mathsf{sk}_{\mathsf{uid},\alpha}$ is given to uid.
- $\mathsf{Sign}(\{\mathsf{avk}_{\mathsf{aid}(\alpha)}\}_{\alpha\in\mathcal{A}}, \mathsf{uid}, \mathbf{usk}[\mathsf{uid}], \mathbf{uvk}[\mathsf{uid}], \{\mathsf{sk}_{\mathsf{uid},\alpha}\}_{\alpha\in\mathcal{A}}, m, \mathbb{P})$ on input an ordered list of attribute authorities' verification keys $\{\mathsf{avk}_{\mathsf{aid}(\alpha)}\}_{\alpha\in\mathcal{A}}$, a user's identity uid, a user's secret and public keys $(\mathbf{usk}[\mathsf{uid}], \mathbf{uvk}[\mathsf{uid}])$, an ordered list of attributes' secret keys $\{\mathsf{sk}_{\mathsf{uid},\alpha}\}_{\alpha\in\mathcal{A}}$ for attributes $\mathcal{A}$ that user uid owns, a message m and a signing predicate $\mathbb{P}$, it outputs a signature Σ on m w.r.t. the signing policy $\mathbb{P}$.
- $\mathsf{Verify}(\{\mathsf{avk}_{\mathsf{aid}(\alpha)}\}_{\alpha\in\mathbb{P}}, m, \Sigma, \mathbb{P})$ is a deterministic algorithm which on input an ordered list of attribute authorities' verification keys $\{\mathsf{avk}_{\mathsf{aid}(\alpha)}\}_{\alpha\in\mathbb{P}}$, a message m, a signature Σ and a signing predicate $\mathbb{P}$, it outputs 1 if Σ is a valid signature on m w.r.t. the signing predicate $\mathbb{P}$ or 0 otherwise.
- $\mathsf{Trace}(\mathsf{tk}, m, \Sigma, \mathbb{P}, \mathbf{uvk})$ is a deterministic algorithm which on input the tracing authority's key tk, a message m, a signature Σ, a signing predicate $\mathbb{P}$, and the public keys table $\mathbf{uvk}$, it outputs an identity $\mathsf{uid} > 0$ of the user who produced Σ plus a proof π_{Trace} attesting to this claim. If it is unable to trace the signature to a user, it returns $(0, \pi_{\mathsf{Trace}})$.
- $\mathsf{Judge}(\{\mathsf{avk}_{\mathsf{aid}(\alpha)}\}_{\alpha\in\mathbb{P}}, m, \Sigma, \mathbb{P}, \mathsf{uid}, \mathbf{uvk}[\mathsf{uid}], \pi_{\mathsf{Trace}})$ is a deterministic algorithm which on input an ordered list of attribute authorities' verification keys $\{\mathsf{avk}_{\mathsf{aid}(\alpha)}\}_{\alpha\in\mathbb{P}}$, a message m, a signature Σ, a signing predicate $\mathbb{P}$, a user's identity uid, a user's public verification key $\mathbf{uvk}[\mathsf{uid}]$, and a tracing proof π_{Trace}, it outputs 1 if π_{Trace} is a valid proof that uid has produced Σ or 0 otherwise.

Security of Decentralized Traceble Attribute-Based Signatures. The security properties we require from a DTABS scheme are: correctness, anonymity, unforgeability, non-frameability, traceability, and tracing soundness. Unlike the model of El Kaafrani et al. [10], we split the games of unforgeability and non-frameability. This separation allows us to strengthen the definition of non-frameability where we allow for the corruption of all authorities. Also, unlike previous models, our model defines the notion of

```
$\mathsf{AddU}(\mathsf{uid})$
- If $\mathsf{uid} \in \mathsf{HUL} \cup \mathsf{CUL}$ Then Return $\perp$.
- $(\mathbf{usk}[\mathsf{uid}], \mathbf{uvk}[\mathsf{uid}]) \leftarrow \mathsf{UKeyGen}(\mathsf{pp})$.
- $\mathsf{HUL} := \mathsf{HUL} \cup \{\mathsf{uid}\}$.

$\mathsf{AddAtt}(\mathsf{uid}, \mathcal{A})$
- If $\exists \alpha \in \mathcal{A}$ s.t. $(\mathsf{uid}, \alpha) \in \mathsf{HAttL}$ Then Return $\perp$.
- If $\mathsf{uid} \notin \mathsf{HUL} \cup \mathsf{CUL}$ Then
  ◦ If $\mathsf{AddU}(\mathsf{uid}) = \perp$ Then Return $\perp$.
- For each $\alpha \in \mathcal{A}$ Do
  ◦ If $\mathsf{aid}(\alpha) \notin \mathsf{HAL}$ Then
      ▪ If $\mathsf{aid}(\alpha) \in \mathsf{CAL}$ Then Return $\perp$.
      ▪ If $\mathsf{AddA}(\mathsf{aid}(\alpha)) = \perp$ Then Return $\perp$.
  ◦ If $\mathsf{ask}_{\mathsf{aid}(\alpha)} = \perp$ Then Return $\perp$.
  ◦ $\mathsf{sk}_{\mathsf{uid},\alpha} \leftarrow \mathsf{AttKeyGen}(\mathsf{ask}_{\mathsf{aid}(\alpha)}, \mathsf{uid}, \mathbf{uvk}[\mathsf{uid}], \alpha)$.
- $\mathsf{HAttL} := \mathsf{HAttL} \cup \{(\mathsf{uid}, \alpha)\}_{\alpha \in \mathcal{A}}$.

$\mathsf{Sign}(\mathsf{uid}, \mathcal{A}, m, \mathbb{P})$
- If $\mathsf{uid} \notin \mathsf{HUL}$ or $\exists \alpha \in \mathcal{A}$ s.t. $(\mathsf{uid}, \alpha) \notin \mathsf{HAttL}$ Then Return $\perp$.
- Return $\perp$ if $\mathbf{usk}[\mathsf{uid}] = \perp$ or $\mathbb{P}(\mathcal{A}) \neq 1$ or $\exists \alpha \in \mathcal{A}$ s.t. $\mathsf{sk}_{\mathsf{uid},\alpha} = \perp$.
- $\Sigma \leftarrow \mathsf{Sign}(\{\mathsf{avk}_{\mathsf{aid}(\alpha)}\}_{\alpha \in \mathcal{A}}, \mathsf{uid}, \mathbf{usk}[\mathsf{uid}], \mathbf{uvk}[\mathsf{uid}], \{\mathsf{sk}_{\mathsf{uid},\alpha}\}_{\alpha \in \mathcal{A}}, m, \mathbb{P})$.
- $\mathsf{SL} := \mathsf{SL} \cup \{(\mathsf{uid}, \mathcal{A}, m, \Sigma, \mathbb{P})\}$.
- Return $\Sigma$.

$\mathsf{CH}_b((\mathsf{uid}_0, \mathcal{A}_0), (\mathsf{uid}_1, \mathcal{A}_1), m, \mathbb{P})$
- If $\exists b \in \{0,1\}$ s.t. $\mathsf{uid}_b \notin \mathsf{HUL}$ or $\mathbb{P}(\mathcal{A}_b) \neq 1$ Then Return $\perp$.
- For i=0 To 1 Do
  ◦ For each $\alpha \in \mathcal{A}_i$ s.t. $(\mathsf{uid}_i, \alpha) \notin \mathsf{HAttL}$ DO
      ▪ If $\mathsf{AddAtt}(\mathsf{uid}_i, \alpha) = \perp$ Then Return $\perp$.
  ◦ If $\mathbf{usk}[\mathsf{uid}_i] = \perp$ or $\exists \alpha \in \mathcal{A}_i$ s.t. $\mathsf{sk}_{\mathsf{uid}_i,\alpha} = \perp$ Then Return $\perp$.
- $\Sigma \leftarrow \mathsf{Sign}(\{\mathsf{avk}_{\mathsf{aid}(\alpha)}\}_{\alpha \in \mathcal{A}_b}, \mathsf{uid}_b, \mathbf{usk}[\mathsf{uid}_b], \mathbf{uvk}[\mathsf{uid}_b], \{\mathsf{sk}_{\mathsf{uid}_b,\alpha}\}_{\alpha \in \mathcal{A}_b}, m, \mathbb{P})$.
- $\mathsf{CL} := \mathsf{CL} \cup \{(m, \Sigma, \mathbb{P})\}$.
- Return $\Sigma$.
```

```
$\mathsf{AddA}(\mathsf{aid})$
- If $\mathsf{aid} \in \mathsf{HAL} \cup \mathsf{CAL}$ Then Return $\perp$.
- $(\mathsf{ask}_{\mathsf{aid}}, \mathsf{avk}_{\mathsf{aid}}) \leftarrow \mathsf{AKeyGen}(\mathsf{pp}, \mathsf{aid})$.
- $\mathsf{HAL} := \mathsf{HAL} \cup \{\mathsf{aid}\}$.

$\mathsf{RevealA}(\mathsf{aid})$
- If $\mathsf{aid} \notin \mathsf{HAL} \setminus (\mathsf{CAL} \cup \mathsf{BAL})$ Then Return $\perp$.
- $\mathsf{BAL} := \mathsf{BAL} \cup \{\mathsf{aid}\}$.
- Return $\mathsf{ask}_{\mathsf{aid}}$.

$\mathsf{RevealU}(\mathsf{uid})$
- If $\mathsf{uid} \notin \mathsf{HUL} \setminus (\mathsf{CUL} \cup \mathsf{BUL})$ Return $\perp$.
- $\mathsf{BUL} := \mathsf{BUL} \cup \{\mathsf{uid}\}$.
- Return $\mathbf{usk}[\mathsf{uid}]$.

$\mathsf{RevealAtt}(\mathsf{uid}, \mathcal{A})$
- Return $\perp$ if $\exists \alpha \in \mathcal{A}$ s.t. $(\mathsf{uid}, \alpha) \notin \mathsf{HAttL} \setminus \mathsf{BAttL}$.
- $\mathsf{BAttL} := \mathsf{BAttL} \cup \{(\mathsf{uid}, \alpha)\}_{\alpha \in \mathcal{A}}$.
- Return $\{\mathsf{sk}_{\mathsf{uid},\alpha}\}_{\alpha \in \mathcal{A}}$.

$\mathsf{CrptA}(\mathsf{aid}, \mathsf{vk})$
- If $\mathsf{aid} \in \mathsf{HAL} \cup \mathsf{CAL}$ Then Return $\perp$.
- $\mathsf{CAL} := \mathsf{CAL} \cup \{\mathsf{aid}\}$.

$\mathsf{CrptU}(\mathsf{uid}, \mathsf{vk})$
- If $\mathsf{uid} \in \mathsf{HUL} \cup \mathsf{CUL}$ Then Return $\perp$.
- $\mathsf{CUL} := \mathsf{CUL} \cup \{\mathsf{uid}\}$.

$\mathsf{Trace}(m, \Sigma, \mathbb{P})$
- Return $\perp$ if $\mathsf{Verify}(\{\mathsf{avk}_{\mathsf{aid}(\alpha)}\}_{\alpha \in \mathbb{P}}, m, \Sigma, \mathbb{P}) = 0$.
- If $(m, \Sigma, \mathbb{P}) \in \mathsf{CL}$ Then Return $\perp$.
- Return $\mathsf{Trace}(\mathsf{tk}, m, \Sigma, \mathbb{P}, \mathbf{uvk})$.
```

Fig. 1. Oracles used in the security games for DTABS

tracing soundness which was recently proposed in the context of group signatures [34]. In our model, we distinguish between bad entities, i.e. those who were initially honest until the adversary learned their secret keys and corrupt entities whose keys have been chosen by the adversary itself.

The experiments used to define the security requirements are shown in Fig. 2. In those experiments, the following global lists are used: HUL is a list of honest users; HAL is a list of honest attribute authorities; HAttL is a list of honest users' attributes and has entries of the form (uid, α); BUL is a list of bad users whose personal secret keys have been revealed to the adversary; BAttL is a list of bad users' attributes whose keys have been revealed to the adversary with entries of the form (uid, α); BAL is a list of bad attribute authorities whose secret keys have been learned by the adversary; CUL is a list of corrupt users whose keys have been chosen by the adversary; CAL is a list of corrupt attribute authorities whose keys have been chosen by the adversary; SL is a list of signatures obtained from the Sign oracle; CL is a list of challenge signatures.

The details of the following oracles are given in Fig. 1.

$\mathsf{AddA}(\mathsf{aid})$ adds an honest attribute authority with identity aid.
$\mathsf{AddU}(\mathsf{uid})$ adds an honest user with identity uid.
$\mathsf{AddAtt}(\mathsf{uid}, \mathcal{A})$ adds honest attributes $\mathcal{A} \subseteq \mathbb{A}$ for user uid. It can be called multiple times to add more attributes for an existing user.
$\mathsf{CrptA}(\mathsf{aid}, \mathsf{vk})$ adds a corrupt attribute authority whose keys are chosen by the adversary.
$\mathsf{CrptU}(\mathsf{uid}, \mathsf{vk})$ adds a corrupt user with identity uid whose personal keys are chosen by the adversary.
$\mathsf{RevealA}(\mathsf{aid})$ returns the secret key $\mathsf{ask}_{\mathsf{aid}}$ of the honest attribute authority aid.
$\mathsf{RevealU}(\mathsf{uid})$ returns the personal secret key $\mathbf{usk}[\mathsf{uid}]$ of user uid.
$\mathsf{RevealAtt}(\mathsf{uid}, \mathcal{A})$ returns the secret keys $\{\mathsf{sk}_{\mathsf{uid},\alpha}\}_{\alpha \in \mathcal{A}}$ for attributes $\mathcal{A} \subseteq \mathbb{A}$ owned by user uid. It can be called multiple times.

$\mathsf{Sign}(\mathsf{uid}, \mathcal{A}, m, \mathbb{P})$ returns a signature Σ on m using attributes $\mathcal{A}$ belonging to user uid where $\mathbb{P}(\mathcal{A}) = 1$.

$\mathsf{CH}_b((\mathsf{uid}_0, \mathcal{A}_0), (\mathsf{uid}_1, \mathcal{A}_1), m, \mathbb{P})$ is a left-right oracle for defining anonymity. On input $(\mathsf{uid}_0, \mathcal{A}_0)$, $(\mathsf{uid}_1, \mathcal{A}_1)$, a message m and a signing policy $\mathbb{P}$ with $\mathbb{P}(\mathcal{A}_0) = \mathbb{P}(\mathcal{A}_1) = 1$, it returns a signature on m using attributes $\mathcal{A}_b$ belonging to user uid_b for $b \leftarrow \{0, 1\}$.

$\mathsf{Trace}(m, \Sigma, \mathbb{P})$ allows the adversary to ask for signatures to be traced.

The details of the security requirements are as follows:

Correctness. This requires that honestly generated signatures verify correctly and trace to the user who produced them. In addition, the Judge algorithm accepts the tracing proof produced by the Trace algorithm. Formally, a DTABS scheme is *correct* if for all $\lambda \in \mathbb{N}$, all PPT adversaries $\mathcal{B}$ have a negligible advantage $\mathsf{Adv}^{\mathrm{Corr}}_{\mathcal{DTABS},\mathcal{B}}(\lambda) := \Pr[\mathsf{Exp}^{\mathrm{Corr}}_{\mathcal{DTABS},\mathcal{B}}(\lambda) = 1]$.

Anonymity. This requires that a signature reveals neither the identity of the user nor the set of attributes used in the signing. In the game, the adversary chooses a message, a signing policy and two users with two, possibly different, sets of attributes satisfying the signing policy. The adversary gets a signature by either user and wins if it correctly guesses the user.

In the game, the adversary can fully corrupt all attribute authorities and learn any user's personal secret key/attribute keys including those used for the challenge. Thus, our definition captures full-key exposure attacks. Since the adversary can sign on behalf of any user, it is redundant to provide it with a sign oracle. The only restriction we impose on the adversary is that it may not query the Trace oracle on the challenge signature.

We focus on CCA-anonymity [4] where the adversary can ask Trace queries at any stage of the game on any signature except the challenge signature. Similarly to [10] WLOG in order to simplify the security proofs, we only allow the adversary a single call to the challenge oracle. One can show by means of a hybrid argument (similar to that used in [10]) that this is sufficient. Also, our definition captures unlinkability since the adversary has access to all users' personal secret keys/attribute keys. Formally, a DTABS scheme is *(fully) anonymous* if for all $\lambda \in \mathbb{N}$, all PPT adversaries $\mathcal{B}$ have a negligible advantage $\mathsf{Adv}^{\mathrm{Anon}}_{\mathcal{DTABS},\mathcal{B}}(\lambda) := \left|\Pr[\mathsf{Exp}^{\mathrm{Anon\text{-}1}}_{\mathcal{DTABS},\mathcal{B}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathrm{Anon\text{-}0}}_{\mathcal{DTABS},\mathcal{B}}(\lambda) = 1]\right|$.

Unforgeability. This captures unforgeability scenarios where the forgery opens to a particular user. It guarantees that even if all users in the system pool their individual attributes, they cannot output a signature that traces to a user whose individual attributes do not satisfy the signing predicate. In the game, the adversary can adaptively create corrupt attribute authorities and learn some of the honest authorities' secret keys as long as there is at least a single honest attribute authority managing one of the attributes required for satisfying the policy used in the forgery. The adversary can also fully corrupt the tracing authority.

Our definition is adaptive and allows the adversary to adaptively choose both the signing predicate and the message used in the forgery. Note that we consider the stronger variant of unforgeability, i.e. (strong unforgeability) where the adversary wins even if it forges a new signature on a message/predicate pair that was queried to the sign oracle. It is easy to adapt the definition if the weaker variant of unforgeability is desired. Formally, a DTABS scheme is *unforgeable* if for all $\lambda \in \mathbb{N}$, all PPT adversaries $\mathcal{B}$ have a negligible advantage $\mathsf{Adv}^{\mathrm{Unforge}}_{\mathcal{DTABS},\mathcal{B}}(\lambda) := \Pr[\mathsf{Exp}^{\mathrm{Unforge}}_{\mathcal{DTABS},\mathcal{B}}(\lambda) = 1]$.

Non-Frameability. This ensures that even if all authorities and users collude, they cannot produce a signature that traces to an honest user whose personal secret key has not been learned by the adversary.

Our definition guarantees that even if the secret attributes' keys for attributes owned by a user are leaked (for instance, by means of interception or leakage by dishonest attribute authorities), it is still impossible to sign on behalf of the user without knowledge of her personal secret key. Thus, our model overcomes the shortcoming of previous models [11, 10] and ensures that an innocent user cannot be framed by dishonest attribute authorities or parties who intercept the communication between the user and the attribute authorities.

```
Experiment: $\mathsf{Exp}^{\text{Corr}}_{\mathcal{DTABS},\mathcal{B}}(\lambda)$
– $(\mathsf{pp}, \mathsf{tk}) \leftarrow \mathsf{Setup}(1^\lambda)$.
– $\mathsf{HUL}, \mathsf{HAttL}, \mathsf{HAL} := \emptyset$.
– $(\mathsf{uid}, \mathcal{A}, m, \mathbb{P}) \leftarrow \mathcal{B}(\mathsf{pp} : \mathsf{AddU}(\cdot), \mathsf{AddAtt}(\cdot,\cdot), \mathsf{AddA}(\cdot))$.
– If $\mathbb{P}(\mathcal{A}) \neq 1$ or $\mathsf{uid} \notin \mathsf{HUL}$ or $\mathbf{usk}[\mathsf{uid}] = \perp$ Then Return 0.
– If $\exists \alpha \in \mathcal{A}$ s.t. $(\mathsf{uid}, \alpha) \notin \mathsf{HAttL}$ or $\mathsf{sk}_{\mathsf{uid},\alpha} = \perp$ or $\mathsf{aid}(\alpha) \notin \mathsf{HAL}$ Then Return 0.
– $\Sigma \leftarrow \mathsf{Sign}(\{\mathsf{avk}_{\mathsf{aid}(\alpha)}\}_{\alpha \in \mathcal{A}}, \mathsf{uid}, \mathbf{usk}[\mathsf{uid}], \mathbf{uvk}[\mathsf{uid}], \{\mathsf{sk}_{\mathsf{uid},\alpha}\}_{\alpha \in \mathcal{A}}, m, \mathbb{P})$.
– If $\mathsf{Verify}(\{\mathsf{avk}_{\mathsf{aid}(\alpha)}\}_{\alpha \in \mathbb{P}}, m, \Sigma, \mathbb{P}) = 0$ Then Return 1.
– $(\mathsf{uid}^*, \pi_{\mathsf{Trace}}) \leftarrow \mathsf{Trace}(\mathsf{tk}, m, \Sigma, \mathbb{P}, \mathbf{uvk})$.
– If $\mathsf{uid}^* \neq \mathsf{uid}$ or $\mathsf{Judge}(\{\mathsf{avk}_{\mathsf{aid}(\alpha)}\}_{\alpha \in \mathbb{P}}, m, \Sigma, \mathbb{P}, \mathsf{uid}, \mathbf{uvk}[\mathsf{uid}], \pi_{\mathsf{Trace}}) = 0$ Then Return 1 Else Return 0.
```

```
Experiment: $\mathsf{Exp}^{\text{Anon-}b}_{\mathcal{DTABS},\mathcal{B}}(\lambda)$
– $(\mathsf{pp}, \mathsf{tk}) \leftarrow \mathsf{Setup}(1^\lambda)$.
– $\mathsf{CAL}, \mathsf{CUL}, \mathsf{HAL}, \mathsf{HUL}, \mathsf{HAttL}, \mathsf{BAL}, \mathsf{BUL}, \mathsf{BAttL}, \mathsf{CL} := \emptyset$.
– $b^* \leftarrow \mathcal{B}\big(\mathsf{pp} : \mathsf{AddU}(\cdot), \mathsf{AddAtt}(\cdot,\cdot), \mathsf{AddA}(\cdot), \mathsf{CrptA}(\cdot,\cdot), \mathsf{CrptU}(\cdot,\cdot), \mathsf{RevealA}(\cdot)$
        $, \mathsf{RevealU}(\cdot), \mathsf{RevealAtt}(\cdot,\cdot), \mathsf{CH}_b((\cdot,\cdot),(\cdot,\cdot),\cdot,\cdot), \mathsf{Trace}(\cdot,\cdot,\cdot)\big)$.
– Return $b^*$.
```

```
Experiment: $\mathsf{Exp}^{\text{Unforge}}_{\mathcal{DTABS},\mathcal{B}}(\lambda)$
– $(\mathsf{pp}, \mathsf{tk}) \leftarrow \mathsf{Setup}(1^\lambda)$.
– $\mathsf{CAL}, \mathsf{CUL}, \mathsf{HAL}, \mathsf{HUL}, \mathsf{HAttL}, \mathsf{BAL}, \mathsf{BUL}, \mathsf{BAttL}, \mathsf{SL} := \emptyset$.
– $(m^*, \Sigma^*, \mathbb{P}^*, \mathsf{uid}^*, \pi^*_{\mathsf{Trace}}) \leftarrow \mathcal{B}\big(\mathsf{pp}, \mathsf{tk} : \mathsf{AddU}(\cdot), \mathsf{AddAtt}(\cdot,\cdot), \mathsf{AddA}(\cdot), \mathsf{CrptA}(\cdot,\cdot), \mathsf{CrptU}(\cdot,\cdot), \mathsf{RevealA}(\cdot)$
        $, \mathsf{RevealU}(\cdot), \mathsf{RevealAtt}(\cdot,\cdot), \mathsf{Sign}(\cdot,\cdot,\cdot,\cdot)\big)$.
– If $\mathsf{Verify}(\{\mathsf{avk}_{\mathsf{aid}(\alpha)}\}_{\alpha \in \mathbb{P}^*}, m^*, \Sigma^*, \mathbb{P}^*) = 0$ Then Return 0.
– If $\mathsf{Judge}(\{\mathsf{avk}_{\mathsf{aid}(\alpha)}\}_{\alpha \in \mathbb{P}^*}, m^*, \Sigma^*, \mathbb{P}^*, \mathsf{uid}^*, \mathbf{uvk}[\mathsf{uid}^*], \pi^*_{\mathsf{Trace}}) = 0$ Then Return 0.
– Let $\mathcal{A}_{\mathsf{uid}^*}$ be the attributes of $\mathsf{uid}^*$ managed by dishonest (i.e. $\in \mathsf{CAL} \cup \mathsf{BAL}$) attribute authorities.
– If $\exists \mathcal{A}$ s.t. $\{(\mathsf{uid}^*, \alpha)\}_{\alpha \in \mathcal{A}} \subseteq \mathsf{BAttL}$ and $\mathbb{P}^*(\mathcal{A} \cup \mathcal{A}_{\mathsf{uid}^*}) = 1$ Then Return 0.
– If $\exists (\mathsf{uid}^*, \cdot, m^*, \Sigma^*, \mathbb{P}^*) \in \mathsf{SL}$ Then Return 0 Else Return 1.
```

```
Experiment: $\mathsf{Exp}^{\text{NF}}_{\mathcal{DTABS},\mathcal{B}}(\lambda)$
– $(\mathsf{pp}, \mathsf{tk}) \leftarrow \mathsf{Setup}(1^\lambda)$.
– $\mathsf{CAL}, \mathsf{CUL}, \mathsf{HAL}, \mathsf{HUL}, \mathsf{HAttL}, \mathsf{BAL}, \mathsf{BUL}, \mathsf{BAttL}, \mathsf{SL} := \emptyset$.
– $(m^*, \Sigma^*, \mathbb{P}^*, \mathsf{uid}^*, \pi^*_{\mathsf{Trace}}) \leftarrow \mathcal{B}\big(\mathsf{pp}, \mathsf{tk} : \mathsf{AddU}(\cdot), \mathsf{AddAtt}(\cdot,\cdot), \mathsf{AddA}(\cdot), \mathsf{CrptA}(\cdot,\cdot), \mathsf{CrptU}(\cdot,\cdot), \mathsf{RevealA}(\cdot)$
        $, \mathsf{RevealU}(\cdot), \mathsf{RevealAtt}(\cdot,\cdot), \mathsf{Sign}(\cdot,\cdot,\cdot,\cdot)\big)$.
– If $\mathsf{Verify}(\{\mathsf{avk}_{\mathsf{aid}(\alpha)}\}_{\alpha \in \mathbb{P}^*}, m^*, \Sigma^*, \mathbb{P}^*) = 0$ Then Return 0.
– If $\mathsf{Judge}(\{\mathsf{avk}_{\mathsf{aid}(\alpha)}\}_{\alpha \in \mathbb{P}^*}, m^*, \Sigma^*, \mathbb{P}^*, \mathsf{uid}^*, \mathbf{uvk}[\mathsf{uid}^*], \pi^*_{\mathsf{Trace}}) = 0$ Then Return 0.
– If $\mathsf{uid} \notin \mathsf{HUL} \setminus \mathsf{BUL}$ or $\exists (\mathsf{uid}^*, \cdot, m^*, \Sigma^*, \mathbb{P}^*) \in \mathsf{SL}$ Then Return 0 Else Return 1.
```

```
Experiment: $\mathsf{Exp}^{\text{Trace}}_{\mathcal{DTABS},\mathcal{B}}(\lambda)$
– $(\mathsf{pp}, \mathsf{tk}) \leftarrow \mathsf{Setup}(1^\lambda)$.
– $\mathsf{CUL}, \mathsf{HAL}, \mathsf{HUL}, \mathsf{HAttL}, \mathsf{BUL}, \mathsf{BAttL}, \mathsf{SL} := \emptyset$.
– $(m^*, \Sigma^*, \mathbb{P}^*) \leftarrow \mathcal{B}\big(\mathsf{pp}, \mathsf{tk} : \mathsf{AddU}(\cdot), \mathsf{AddAtt}(\cdot,\cdot), \mathsf{AddA}(\cdot), \mathsf{CrptU}(\cdot,\cdot), \mathsf{RevealU}(\cdot), \mathsf{RevealAtt}(\cdot,\cdot), \mathsf{Sign}(\cdot,\cdot,\cdot,\cdot)\big)$.
– If $\mathsf{Verify}(\{\mathsf{avk}_{\mathsf{aid}(\alpha)}\}_{\alpha \in \mathbb{P}^*}, m^*, \Sigma^*, \mathbb{P}^*) = 0$ Then Return 0.
– $(\mathsf{uid}^*, \pi^*_{\mathsf{Trace}}) \leftarrow \mathsf{Trace}(\mathsf{tk}, m^*, \Sigma^*, \mathbb{P}^*, \mathbf{uvk})$.
– If $\mathsf{uid}^* = 0$ or $\mathsf{Judge}(\{\mathsf{avk}_{\mathsf{aid}(\alpha)}\}_{\alpha \in \mathbb{P}^*}, m^*, \Sigma^*, \mathbb{P}^*, \mathsf{uid}^*, \mathbf{uvk}[\mathsf{uid}^*], \pi^*_{\mathsf{Trace}}) = 0$ Then Return 1 Else Return 0.
```

```
Experiment: $\mathsf{Exp}^{\text{TS}}_{\mathcal{DTABS},\mathcal{B}}(\lambda)$
– $(\mathsf{pp}, \mathsf{tk}) \leftarrow \mathsf{Setup}(1^\lambda)$.
– $\mathsf{CAL}, \mathsf{CUL}, \mathsf{HAL}, \mathsf{HUL}, \mathsf{HAttL}, \mathsf{BAL}, \mathsf{BUL}, \mathsf{BAttL} := \emptyset$.
– $(m^*, \Sigma^*, \mathbb{P}^*, \mathsf{uid}_1, \pi_{\mathsf{Trace},1}, \mathsf{uid}_2, \pi_{\mathsf{Trace},2}) \leftarrow \mathcal{B}\big(\mathsf{pp}, \mathsf{tk} : \mathsf{AddU}(\cdot), \mathsf{AddAtt}(\cdot,\cdot), \mathsf{AddA}(\cdot), \mathsf{CrptA}(\cdot,\cdot), \mathsf{CrptU}(\cdot,\cdot)$
        $, \mathsf{RevealA}(\cdot), \mathsf{RevealU}(\cdot), \mathsf{RevealAtt}(\cdot,\cdot)$.
– If $\mathsf{uid}_1 = \mathsf{uid}_2$ or $\mathsf{Verify}(\{\mathsf{avk}_{\mathsf{aid}(\alpha)}\}_{\alpha \in \mathbb{P}^*}, m^*, \Sigma^*, \mathbb{P}^*) = 0$ Then Return 0.
– If $\exists i \in \{1,2\}$ s.t. $\mathsf{Judge}(\{\mathsf{avk}_{\mathsf{aid}(\alpha)}\}_{\alpha \in \mathbb{P}^*}, m^*, \Sigma^*, \mathbb{P}^*, \mathsf{uid}_i, \mathbf{uvk}[\mathsf{uid}_i], \pi_{\mathsf{Trace},i}) = 0$ Then Return 0 Else Return 1.
```

Fig. 2. Security experiments for decentralized traceable attribute-based signatures

In the game, the adversary can fully corrupt all attribute authorities as well as the tracing authority. It can also corrupt as many users of the system as it wishes. We just require that the forgery output by the adversary is a valid signature and traces to a user whose personal secret key has not been revealed to the adversary. Formally, a DTABS scheme is *non-frameable* if for all $\lambda \in \mathbb{N}$, all PPT adversaries $\mathcal{B}$ have a negligible advantage $\mathsf{Adv}^{\text{NF}}_{\mathcal{DTABS},\mathcal{B}}(\lambda) := \Pr[\mathsf{Exp}^{\text{NF}}_{\mathcal{DTABS},\mathcal{B}}(\lambda) = 1]$.

Traceability. This ensures that the adversary cannot produce a signature that cannot be traced. In the game, the adversary is allowed to corrupt the tracing authority and learn both the personal secret key and attributes' keys of any user. However, unlike in the unforgeability and non-frameability games, we require that all the attribute authorities are honest. We emphasize that such an assumption is inevitable as knowing the secret key of any attribute authority would allow the adversary to grant attributes to dummy users resulting in untraceable signature. Formally, a DTABS scheme is *traceable* if for all $\lambda \in \mathbb{N}$, all PPT adversaries $\mathcal{B}$ have a negligible advantage $\mathsf{Adv}^{\text{Trace}}_{\mathcal{DTABS},\mathcal{B}}(\lambda) := \Pr[\mathsf{Exp}^{\text{Trace}}_{\mathcal{DTABS},\mathcal{B}}(\lambda) = 1]$.

Tracing Soundness. This new requirement, which was not defined in previous models [11, 10], ensures that even if all authorities (including the tracing authority) and users in the system are all corrupt and collude, they cannot produce a valid signature that traces to two different users. Among other things, this prevents users from claiming authorship of signatures they did not produce or imputing possibly problematic signatures to other users. Formally, a DTABS scheme satisfies *tracing soundness* if for all $\lambda \in \mathbb{N}$, all PPT adversaries $\mathcal{B}$ have a negligible advantage $\mathsf{Adv}^{\text{TS}}_{\mathcal{DTABS},\mathcal{B}}(\lambda) := \Pr[\mathsf{Exp}^{\text{TS}}_{\mathcal{DTABS},\mathcal{B}}(\lambda) = 1]$.

3.1 Comparison with Existing Models

As mentioned earlier, our model is stronger than previous models. Assigning personal keys to users serves to minimize the trust placed in attribute authorities and hence we get a stronger and (more realistic) definition for non-frameability. It is important in this setting to prevent corrupt authorities and/or malicious parties intercepting the communication between attribute authorities and users from being able to frame users, which might result in innocent users being held accountable for signatures they have not produced. In addition, we get a cleaner definition for traceability than that used in [10]. In particular, the public-key table is used as a reference to determine whether a particular user has indeed joined the system, which eliminates the need for the tracing authority to have a read access to the local registration tables maintained by the attribute authorities. Moreover, unlike previous models, we capture the useful notion of tracing soundness [34], which ensures that only the signer can claim authorship of the signature. The latter is vital in ensuring that a valid signature can only trace to a single user.

4 Building Blocks

In this section we present the building blocks that we use in our constructions.

4.1 Digital Signatures

A *digital signature* for a message space $\mathcal{M}_{\mathcal{DS}}$ is a tuple of polynomial-time algorithms $\mathcal{DS} := (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$, where KeyGen outputs a pair of secret/verification keys $(\mathsf{sk}, \mathsf{vk})$ for the signer; $\mathsf{Sign}(\mathsf{sk}, m)$ outputs a signature σ on the message m; $\mathsf{Verify}(\mathsf{vk}, m, \sigma)$ outputs 1 if σ is a valid signature on the message m or 0 otherwise.

Besides correctness, the security of a digital signature requires existential unforgeability under an adaptive chosen-message attack which demands that all PPT adversaries $\mathcal{B}$ have a negligible advantage in winning the following game:

- A key pair $(\mathsf{sk}, \mathsf{vk})$ is generated and vk is sent to $\mathcal{B}$.
- Adversary $\mathcal{B}$ makes a polynomial number of queries to a sign oracle $\mathsf{Sign}(\mathsf{sk}, \cdot)$.
- Eventually, $\mathcal{B}$ halts by outputting (σ^*, m^*) and wins if σ^* is valid on m^*, and m^* was not queried to Sign.

A weaker variant of existential unforgeability (i.e. existential unforgeability under a weak chosen-message attack) requires that the adversary sends all its queries before seeing the verification key.

```
$\mathcal{DS}.\mathsf{KeyGen}(\mathcal{P})$
- Choose $x, y \leftarrow \mathbb{Z}_p$ and set $(\tilde{X}, \tilde{Y}) := (\tilde{G}^x, \tilde{G}^y)$.
- Return $\mathsf{sk} := (x, y)$ and $\mathsf{vk} := (\tilde{X}, \tilde{Y})$.
$\mathcal{DS}.\mathsf{Sign}(\mathsf{sk}, m)$
- To sign $m \in \mathbb{Z}_p$, choose $r \leftarrow \mathbb{Z}_p$ s.t. $x + r \cdot y + m \neq 0$,
      set $\sigma := G^{\frac{1}{x+r \cdot y+m}}$. Return $(\sigma, r)$.
$\mathcal{DS}.\mathsf{Verify}(\mathsf{vk}, m, (\sigma, r))$
- Return 1 if $e(\sigma, \tilde{X} \cdot \tilde{Y}^r \cdot \tilde{G}^m) = e(G, \tilde{G})$ and 0 otherwise.
```

```
$\mathcal{DS}.\mathsf{KeyGen}(\mathcal{P})$
- Choose $x \leftarrow \mathbb{Z}_p$ and set $\tilde{X} := \tilde{G}^x$.
- Return $\mathsf{sk} := x$ and $\mathsf{vk} := \tilde{X}$.
$\mathcal{DS}.\mathsf{Sign}(\mathsf{sk}, m)$
- To sign $m \in \mathbb{Z}_p$ s.t. $x + m \neq 0$, return $\sigma := G^{\frac{1}{x+m}}$.
$\mathcal{DS}.\mathsf{Verify}(\mathsf{vk}, m, \sigma)$
- Return 1 if $e(\sigma, \tilde{X} \cdot \tilde{G}^m) = e(G, \tilde{G})$ and 0 otherwise.
```

Fig. 3. The full Boneh-Boyen (Left) and the weak Boneh-Boyen (Right) signatures

```
$\mathcal{TS}.\mathsf{KeyGen}(\mathcal{P})$
- $x_1, x_2, y \leftarrow \mathbb{Z}_p$, set $(X_1, X_2, \tilde{Y}) := (G^{x_1}, G^{x_2}, \tilde{G}^y)$.
- Return $\big(\mathsf{sk} := (x_1, x_2, y), \mathsf{vk} := (X_1, X_2, \tilde{Y})\big)$.
$\mathcal{TS}.\mathsf{Sign}(\mathsf{sk}, \tilde{\tau}, \tilde{M})$
- $a \leftarrow \mathbb{Z}_p$, $A := G^a$, $B := A^y$, $\tilde{D} := (\tilde{G} \cdot \tilde{\tau}^{-x_1} \cdot \tilde{M}^{-x_2})^{\frac{1}{a}}$.
- Return $\sigma := \Big(A, B, \tilde{D}\Big)$.
$\mathcal{TS}.\mathsf{Verify}(\mathsf{vk}, \tilde{\tau}, \tilde{M}, \sigma)$
- Return 1 if $e(A, \tilde{Y}) = e(B, \tilde{G})$
      and $e(A, \tilde{D}) e(X_1, \tilde{\tau}) e(X_2, \tilde{M}) = e(G, \tilde{G})$.
```

```
$\mathcal{TS}.\mathsf{KeyGen}(\mathcal{P})$
- $w, x, \{y_i\}_{i=1}^{3} \leftarrow \mathbb{Z}_p$, set $(\tilde{W}, \tilde{X}, \tilde{Y}_i) := (\tilde{G}^w, \tilde{G}^x, \tilde{G}^{y_i})$.
- Return $\big(\mathsf{sk} := (w, x, \{y_i\}_{i=1}^{3}), \mathsf{vk} := (\tilde{W}, \tilde{X}, \{\tilde{Y}_i\}_{i=1}^{3})\big)$.
$\mathcal{TS}.\mathsf{Sign}(\mathsf{sk}, \tau, M)$
- $R \leftarrow \mathbb{G}$, $a \leftarrow \mathbb{Z}_p$, $A := G^a$, $\tilde{D} := \tilde{G}^{\frac{1}{a}}$,
    $B := G^{x - aw} \cdot R^{-y_1} \cdot \tau^{-y_2} \cdot M^{-y_3}$.
- Return $\sigma := \Big(A, B, \tilde{D}, R\Big)$.
$\mathcal{TS}.\mathsf{Verify}(\mathsf{vk}, \tau, M, \sigma)$
- Return 1 if $e(A, \tilde{D}) = e(G, \tilde{G})$ and
    $e(G, \tilde{X}) = e(A, \tilde{W}) e(B, \tilde{G}) e(R, \tilde{Y}_1) e(\tau, \tilde{Y}_2) e(M, \tilde{Y}_3)$.
```

Fig. 4. Two instantiations of tagged signatures

In this paper, we will use two digital signatures by Boneh and Boyen [7], which we refer to as the full Boneh-Boyen signature (Fig. 3 (Left)) and the weak Boneh-Boyen signature (Fig. 3 (Right)), respectively. Both schemes are secure under the q-SDH assumption. The weaker scheme is only secure under a weak chosen-message attack. Let $\mathcal{P} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, G, \tilde{G}, e)$ be the description of an asymmetric bilinear group. The schemes are given in Fig. 3.

4.2 Tagged Signatures

In [10], the authors defined a variant of a signature scheme they called a *tagged signature*. Tagged signatures are similar to digital signatures with the only difference being that the signing and verification algorithms take as an additional input a tag τ. Formally, a tagged signature scheme for a message space $\mathcal{M}_{\mathcal{TS}}$ and a tag space $\mathcal{T}_{\mathcal{TS}}$ is a tuple of polynomial-time algorithms $\mathcal{TS} := (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$, where $\mathsf{Setup}(1^\lambda)$ outputs common public parameters param; $\mathsf{KeyGen}(\mathsf{param})$ outputs a pair of secret/verification keys $(\mathsf{sk}, \mathsf{vk})$; $\mathsf{Sign}(\mathsf{sk}, \tau, m)$ outputs a signature σ on the tag τ and the message m; $\mathsf{Verify}(\mathsf{vk}, \tau, m, \sigma)$ outputs 1 if σ is a valid signature on τ and m w.r.t. the verification key vk.

Besides correctness, the security of a tagged signature [10] requires existential unforgeability under an adaptive chosen-message-tag attack which is similar to the definition of existential unforgeability of digital signatures with the difference being the winning condition is now that the adversary outputs a valid signature on a pair (τ^*, m^*) that is different from the pairs queried to the sign oracle.

In this paper, we will use two instantiations of tagged signatures which are based on two structure-preserving signature schemes [1] by Abe et al. [2]. The first instantiation (shown in Fig. 4 (Left)) is based on the re-randomizable signature scheme in [2] which signs messages in $\mathbb{G}_2^2$. We refer to this scheme as AGHO1 after its authors. The tag space of this instantiation is $\mathcal{T}_{\mathcal{TS}} := \mathbb{G}_2$, and the message space is $\mathcal{M}_{\mathcal{TS}} := \mathbb{G}_2$. The tagged signature size is $\mathbb{G}_1^2 \times \mathbb{G}_2$ and the signature is fully re-randomizable. The unforgeability of the signature scheme rests on an interactive assumption. See [2] for more details.

The second instantiation (shown in Fig. 4 (Right)) is based on the strongly unforgeable signature scheme from [2] whose unforgeability reduces to the non-interactive q-AGHO assumption (cf. Section

2). The message space of the underlying signature scheme is $\mathbb{G}_1^3$ (where the first element is chosen randomly by the signer), we refer to the underlying scheme as AGHO2. The tag space of this instantiation is $\mathcal{T}_{\mathcal{TS}} := \mathbb{G}_1$, and the message space is $\mathcal{M}_{\mathcal{TS}} := \mathbb{G}_1$. The signature size is $\mathbb{G}_1^3 \times \mathbb{G}_2$.

In both instantiations $\mathcal{TS}.\mathsf{Setup}(1^\lambda)$ outputs $\mathcal{P} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, G, \tilde{G}, e)$ which is the description of an asymmetric bilinear group.

4.3 Strongly Unforgeable One-Time Signatures

A one-time signature scheme is a signature scheme that is unforgeable against an adversary who makes a single signing query. *Strong Unforgeability* requires that the adversary cannot even forge a new signature on a message that she queried the sign oracle on. In this paper, we will instantiate the one-time signature using the full Boneh-Boyen signature scheme from Fig. 3.

4.4 Non-Interactive Zero-Knowledge Proofs

Let $\mathcal{R}$ be an efficiently computable relation on pairs (x, w), where we call x the statement and w the witness. We define the corresponding language $\mathcal{L}$ as all the statements x in $\mathcal{R}$. A Non-Interactive Zero-Knowledge (NIZK) proof system [6] for $\mathcal{R}$ is defined by a tuple of algorithms $\mathcal{NIZK} := (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Extract}, \mathsf{SimSetup}, \mathsf{SimProve})$.

Setup takes as input a security parameter 1^λ and outputs a common reference string crs and an extraction key xk which allows for witness extraction. Prove takes as input (crs, x, w) and outputs a proof π that $(x, w) \in \mathcal{R}$. Verify takes as input (crs, x, π) and outputs 1 if the proof is valid, or 0 otherwise. $\mathsf{Extract}$ takes as input $(\mathsf{crs}, \mathsf{xk}, x, \pi)$ and outputs a witness. $\mathsf{SimSetup}$ takes as input a security parameter 1^λ and outputs a simulated reference string $\mathsf{crs}_{\mathsf{Sim}}$ and a trapdoor key tr that allows for proof simulation. $\mathsf{SimProve}$ takes as input $(\mathsf{crs}_{\mathsf{Sim}}, \mathsf{tr}, x)$ and outputs a simulated proof π_{Sim} without a witness.

We require: completeness, soundness and zero-knowledge. Completeness requires that honestly generated proofs are accepted; Soundness requires that it is infeasible (but for a small probability) to produce a convincing proof for a false statement; Zero-knowledge requires that a proof reveals no information about the witness used. The formal definitions can be found in Appendix A.

Groth-Sahai Proofs. Groth-Sahai (GS) proofs [18] are efficient non-interactive proofs in the Common Reference String (CRS) model. In this paper, we will be using the SXDH-based instantiation, which is the most efficient instantiation of the proofs [16]. The language for the system has the form

$$\mathcal{L} := \{\mathsf{statement} \mid \exists\, \mathsf{witness} : E_1(\mathsf{statement}, \mathsf{witness}), \ldots, E_n(\mathsf{statement}, \mathsf{witness}) \text{ hold } \},$$

where $E_i(\mathsf{statement}, \cdot)$ is one of the types of equation summarized in Fig. 5, where $\underline{X_1}, \ldots, \underline{X_m} \in \mathbb{G}_1$, $\underline{\tilde{Y}_1}, \ldots, \underline{\tilde{Y}_n} \in \mathbb{G}_2$, $\underline{x_1}, \ldots, \underline{x_m}, \underline{\tilde{y}_1}, \ldots, \underline{\tilde{y}_n} \in \mathbb{Z}_p$ are secret variables (hence underlined), whereas $A_i, T \in \mathbb{G}_1$, $\tilde{B}_i, \tilde{T} \in \mathbb{G}_2$, $a_i, \tilde{b}_i, k_{i,j}, t \in \mathbb{Z}_p$, $t_T \in \mathbb{G}_T$ are public constants. For clarity, we also accent exponents to be mapped to group $\mathbb{G}_2$ with $\tilde{}$.

The system works by first committing to the elements of the witness and then proving that the commitments satisfy the source equations.

The proof system has perfect completeness, perfect soundness, composable witness-indistinguishability/zero-knowledge. Refer to [18] for the formal definitions and details of the instantiations.

4.5 Tag-Based Encryption

A Tag-based Public-Key Encryption (TPKE) scheme [27] is similar to a public-key encryption scheme with the only difference being that both Enc and Dec algorithms take as an additional input a tag t. Formally, a TPKE scheme for a message space $\mathcal{M}_{\mathcal{TPKE}}$ and a tag space $\mathcal{T}_{\mathcal{TPKE}}$ is a tuple of polynomial-time algorithms $\mathcal{TPKE} := (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{IsValid})$, where $\mathsf{KeyGen}(1^\lambda)$ outputs a public/secret key

- **Pairing Product Equation (PPE):** $\prod_{i=1}^{n} e(A_i, \underline{\tilde{Y}_i}) \prod_{i=1}^{m} e(\underline{X_i}, \tilde{B}_i) \prod_{i=1}^{m} \prod_{j=1}^{n} e(\underline{X_i}, \underline{\tilde{Y}_j})^{k_{i,j}} = t_T\cdot$
- **Multi-Scalar Multiplication Equation (MSME) in $\mathbb{G}_1$:** $\prod_{i=1}^{n} A_i^{\underline{\tilde{y}_i}} \prod_{i=1}^{m} \underline{X_i}^{\tilde{b}_i} \prod_{i=1}^{m} \prod_{j=1}^{n} \underline{X_i}^{k_{i,j}\underline{\tilde{y}_j}} = T\cdot$
- **Multi-Scalar Multiplication Equation (MSME) in $\mathbb{G}_2$:** $\prod_{i=1}^{n} \underline{\tilde{Y}_i}^{a_i} \prod_{i=1}^{m} \tilde{B}_i^{\underline{x_i}} \prod_{i=1}^{m} \prod_{j=1}^{n} \underline{\tilde{Y}_j}^{k_{i,j}\underline{x_i}} = \tilde{T}\cdot$
- **Quadratic Equation (QE) in $\mathbb{Z}_p$:** $\sum_{i=1}^{n} a_i\underline{\tilde{y}_i} + \sum_{i=1}^{m} \underline{x_i}\tilde{b}_i + \sum_{i=1}^{m} \sum_{j=1}^{n} \underline{x_i}\underline{\tilde{y}_j} = t\cdot$

Fig. 5. Types of equations over bilinear groups

```
Experiment: $\mathsf{Exp}^{\text{ST-WIND-CCA-b}}_{\mathcal{TPKE},\mathcal{B}}(\lambda)$:
– $(t^*, \mathsf{st}_{\mathsf{init}}) \leftarrow \mathcal{B}_{\mathsf{init}}\left(1^\lambda\right)$.
– $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.
– $(m_0, m_1, \mathsf{st}_{\mathsf{find}}) \leftarrow \mathcal{B}_{\mathsf{find}}\left(\mathsf{st}_{\mathsf{init}}, \mathsf{pk} : \mathsf{Dec}^{t^*}(\mathsf{sk}, \cdot, \cdot)\right)$, where $|m_0| = |m_1|$.
– $C_{\text{tbe},b} \leftarrow \mathsf{Enc}(\mathsf{pk}, t^*, m_b)$.
– $b^* \leftarrow \mathcal{B}_{\mathsf{guess}}\left(\mathsf{st}_{\mathsf{find}}, C_{\text{tbe},b} : \mathsf{Dec}^{t^*}(\mathsf{sk}, \cdot, \cdot)\right)$, where $\mathsf{Dec}^{t^*}$ returns $\perp$ if queried on $t^*$.
– Return $b^*$.
```

Fig. 6. The ST-WIND-CCA security game for tag-based encryption

pair $(\mathsf{pk}, \mathsf{sk})$; $\mathsf{Enc}(\mathsf{pk}, t, m)$ outputs a ciphertext C_{tbe}; $\mathsf{Dec}(\mathsf{sk}, t, C_{\text{tbe}})$ outputs a message m or the reject symbol $\perp$; $\mathsf{IsValid}(\mathsf{pk}, t, C_{\text{tbe}})$ outputs 1 if the ciphertext is valid under the tag t w.r.t. the pubic key pk or 0 otherwise.

Besides correctness, we require selective-tag weak indistinguishability under an adaptive chosen-ciphertext attack (ST-WIND-CCA), which requires for all $\lambda \in \mathbb{N}$, the advantage $\mathsf{Adv}^{\text{ST-WIND-CCA}}_{\mathcal{TPKE},\mathcal{B}}(\lambda) := \left|\Pr[\mathsf{Exp}^{\text{ST-WIND-CCA-1}}_{\mathcal{TPKE},\mathcal{B}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\text{ST-WIND-CCA-0}}_{\mathcal{TPKE},\mathcal{B}}(\lambda) = 1]\right|$ is negligible for all polynomial-time adversaries $\mathcal{B}$ where the game is shown in Fig. 6.

In order to get more efficient constructions, we use the asymmetric variant [21] of Kiltz' tag-based encryption [24] shown in Fig. 7. The security of this instantiation in Type-3 groups, which is based on the SDLIN assumption (cf. Section 2), requires a polynomial (in the security parameter) message space so that we can efficiently search when decrypting. Such a restriction is not an issue for our setting as in our constructions the adversary runs in polynomial-time and hence can only add polynomially many users in the security games. Also, note that the public key could be chosen from either $\mathbb{G}_1$ or $\mathbb{G}_2$ which determines whether SDLIN holds in $\mathbb{G}_1$ or $\mathbb{G}_2$.

5 Our Generic Construction

In this section, we present our generic construction for decentralized traceable attribute-based signatures.

```
$\mathcal{TPKE}.\mathsf{KeyGen}(1^\lambda)$
- $\mathcal{P} \leftarrow \mathsf{BGrpSetup}(1^\lambda)$.
- $\tilde{K}, \tilde{L} \leftarrow \mathbb{G}_2$; $f, h \leftarrow \mathbb{Z}_p$; $F := G^f$; $H := G^h$.
- $\mathsf{pk} := (\mathcal{P}, F, H, \tilde{K}, \tilde{L})$; $\mathsf{sk} := (f, h)$.

$\mathcal{TPKE}.\mathsf{Enc}(\mathsf{pk}, t, M)$
- $r_1, r_2 \leftarrow \mathbb{Z}_p$.
- $C_1 := F^{r_1}$; $C_2 := H^{r_2}$; $C_3 := G^{r_1+r_2} \cdot M$.
- $\tilde{C}_4 := (\tilde{G}^t \cdot \tilde{K})^{r_1}$; $\tilde{C}_5 := (\tilde{G}^t \cdot \tilde{L})^{r_2}$.
- $C_{\text{tbe}} := (C_1, C_2, C_3, \tilde{C}_4, \tilde{C}_5)$.

$\mathcal{TPKE}.\mathsf{Dec}(\mathsf{sk}, t, C_{\text{tbe}})$
- If $\mathcal{TPKE}.\mathsf{IsValid}(\mathsf{pk}, t, C_{\text{tbe}}) = 0$ Then return $\perp$.
- Parse $C_{\text{tbe}}$ as $(C_1, C_2, C_3, \tilde{C}_4, \tilde{C}_5)$.
- $M := C_3 \cdot C_1^{-1/f} C_2^{-1/h}$.

$\mathcal{TPKE}.\mathsf{IsValid}(\mathsf{pk}, t, C_{\text{tbe}})$
- Parse $C_{\text{tbe}}$ as $(C_1, C_2, C_3, \tilde{C}_4, \tilde{C}_5)$.
- If $e(F, \tilde{C}_4) \neq e(C_1, \tilde{G}^t \cdot \tilde{K})$ or
      $e(H, \tilde{C}_5) \neq e(C_2, \tilde{G}^t \cdot \tilde{L})$ Then Return 0.
- Else Return 1.
```

Fig. 7. The asymmetric instantiation [21] of the tag-based encryption by Kiltz [24]

```
$\underline{\mathsf{Setup}(1^\lambda)}$
 - $(\mathsf{crs}_1, \mathsf{xk}_1) \leftarrow \mathcal{NIZK}_1.\mathsf{Setup}(1^\lambda)$, $\mathsf{crs}_2 \leftarrow \mathcal{NIZK}_2.\mathsf{Setup}(1^\lambda)$, $(\mathsf{epk}, \mathsf{esk}) \leftarrow \mathcal{TPKE}.\mathsf{KeyGen}(1^\lambda; \rho)$.
 - Choose collision-resistant hash functions $\hat{\mathcal{H}} : \{0,1\}^* \to \mathcal{T}_{\mathcal{TPKE}}$ and $\mathcal{H} : \{0,1\}^* \to \mathcal{M}_{\mathcal{OTS}}$.
 - Let $\mathsf{tk} := \mathsf{esk}$ and $\mathsf{pp} := (1^\lambda, \mathsf{crs}_1, \mathsf{crs}_2, \mathsf{epk}, \hat{\mathcal{H}}, \mathcal{H})$. Return $\mathsf{pp}$.
$\underline{\mathsf{AKeyGen}(\mathsf{pp}, \mathsf{aid})}$
 - $(\mathsf{avk}_{\mathsf{aid}}, \mathsf{ask}_{\mathsf{aid}}) \leftarrow \mathcal{TS}.\mathsf{KeyGen}(1^\lambda)$. Return $(\mathsf{avk}_{\mathsf{aid}}, \mathsf{ask}_{\mathsf{aid}})$.
$\underline{\mathsf{UKeyGen}(\mathsf{pp})}$
 - $(\mathbf{uvk}[\mathsf{uid}], \mathbf{usk}[\mathsf{uid}]) \leftarrow \mathcal{WDS}.\mathsf{KeyGen}(1^\lambda)$. Return $(\mathbf{uvk}[\mathsf{uid}], \mathbf{usk}[\mathsf{uid}])$.
$\underline{\mathsf{AttKeyGen}(\mathsf{ask}_{\mathsf{aid}(\alpha)}, \mathsf{uid}, \mathbf{uvk}[\mathsf{uid}], \alpha)}$
 - $\mathsf{sk}_{\mathsf{uid},\alpha} \leftarrow \mathcal{TS}.\mathsf{Sign}(\mathsf{ask}_{\mathsf{aid}(\alpha)}, \mathbf{uvk}[\mathsf{uid}], \alpha)$. Return $\mathsf{sk}_{\mathsf{uid},\alpha}$.
$\underline{\mathsf{Sign}(\{\mathsf{avk}_{\mathsf{aid}(\alpha)}\}_{\alpha \in \mathcal{A}}, \mathsf{uid}, \mathbf{usk}[\mathsf{uid}], \mathbf{uvk}[\mathsf{uid}], \{\mathsf{sk}_{\mathsf{uid},\alpha}\}_{\alpha \in \mathcal{A}}, m, \mathbb{P})}$
 - Return $\perp$ if $\mathbb{P}(\mathcal{A}) = 0$.
 - $(\mathsf{otsvk}, \mathsf{otssk}) \leftarrow \mathcal{OTS}.\mathsf{KeyGen}(1^\lambda)$.
 - $C_{\mathrm{tbe}} \leftarrow \mathcal{TPKE}.\mathsf{Enc}(\mathsf{epk}, \hat{\mathcal{H}}(\mathsf{otsvk}), \mathbf{uvk}[\mathsf{uid}]; \mu)$.
 - $\sigma \leftarrow \mathcal{WDS}.\mathsf{Sign}(\mathbf{usk}[\mathsf{uid}], \hat{\mathcal{H}}(\mathsf{otsvk}))$.
 - $\pi \leftarrow \mathcal{NIZK}_1.\mathsf{Prove}(\mathsf{crs}_1, \{\mathbf{uvk}[\mathsf{uid}], \mu, z, \{\sigma_{\alpha_i}\}_{i=1}^{|\mathbb{P}|}, \sigma\} : (C_{\mathrm{tbe}}, \hat{\mathcal{H}}(\mathsf{otsvk}), \mathsf{epk}, \{\mathsf{avk}_{\mathsf{aid}(\alpha_i)}\}_{i=1}^{|\mathbb{P}|}, \{\alpha_i\}_{i=1}^{|\mathbb{P}|}) \in \mathcal{L}_1)$.
 - $\sigma_{\mathrm{ots}} \leftarrow \mathcal{OTS}.\mathsf{Sign}(\mathsf{otssk}, (\mathcal{H}(m, \mathbb{P}), \pi, C_{\mathrm{tbe}}, \mathsf{otsvk}))$.
 - Return $\Sigma := (\sigma_{\mathrm{ots}}, \pi, C_{\mathrm{tbe}}, \mathsf{otsvk})$.
$\underline{\mathsf{Verify}(\{\mathsf{avk}_{\mathsf{aid}(\alpha)}\}_{\alpha \in \mathbb{P}}, m, \Sigma, \mathbb{P})}$
 - Parse $\Sigma$ as $(\sigma_{\mathrm{ots}}, \pi, C_{\mathrm{tbe}}, \mathsf{otsvk})$.
 - Return 1 if all the following verify; otherwise, return 0:
    ○ $\mathcal{OTS}.\mathsf{Verify}(\mathsf{otsvk}, (\mathcal{H}(m, \mathbb{P}), \pi, C_{\mathrm{tbe}}, \mathsf{otsvk}), \sigma_{\mathrm{ots}}) = 1$.
    ○ $\mathcal{NIZK}_1.\mathsf{Verify}(\mathsf{crs}_1, \pi) = 1$.
    ○ $\mathcal{TPKE}.\mathsf{IsValid}(\mathsf{epk}, \hat{\mathcal{H}}(\mathsf{otsvk}), C_{\mathrm{tbe}}) = 1$.
$\underline{\mathsf{Trace}(\mathsf{tk}, m, \Sigma, \mathbb{P}, \mathbf{uvk})}$
 - Return $(\perp, \perp)$ if $\mathsf{Verify}(\{\mathsf{avk}_{\mathsf{aid}(\alpha)}\}_{\alpha \in \mathbb{P}}, m, \sigma, \mathbb{P}) = 0$.
 - $\mathsf{vk}_{\mathsf{uid}} \leftarrow \mathcal{PKE}.\mathsf{Dec}(\mathsf{tk}, \hat{\mathcal{H}}(\mathsf{otsvk}), C_{\mathrm{tbe}})$.
 - $\pi_{\mathsf{Trace}} \leftarrow \mathcal{NIZK}_2.\mathsf{Prove}(\mathsf{crs}_2, \{\mathsf{tk}, \rho\} : (C_{\mathrm{tbe}}, \hat{\mathcal{H}}(\mathsf{otsvk}), \mathsf{epk}, \mathsf{vk}_{\mathsf{uid}}) \in \mathcal{L}_2)$.
 - Return $(i, \pi_{\mathsf{Trace}})$ if $\exists i$ s.t. $\mathsf{vk}_{\mathsf{uid}} = \mathbf{uvk}[i]$. Otherwise, return $(0, \pi_{\mathsf{Trace}})$.
$\underline{\mathsf{Judge}(\{\mathsf{avk}_{\mathsf{aid}(\alpha)}\}_{\alpha \in \mathbb{P}}, m, \Sigma, \mathbb{P}, \mathsf{uid}, \mathbf{uvk}[\mathsf{uid}], \pi_{\mathsf{Trace}})}$
 - If $(\mathsf{uid}, \pi_{\mathsf{Trace}}) = (\perp, \perp)$ Then Return $\mathsf{Verify}(\{\mathsf{avk}_{\mathsf{aid}(\alpha)}\}_{\alpha \in \mathbb{P}}, m, \Sigma, \mathbb{P}) = 0$.
 - Return $\mathcal{NIZK}_2.\mathsf{Verify}(\mathsf{crs}_2, \pi_{\mathsf{Trace}})$.
```

Fig. 8. Our generic construction for DTABS

Overview of the construction. Our generic construction builds upon those in [29, 10], but with major and distinct differences. Unlike [29, 10], we dispense with relying on the so-called pseudo-attribute technique to bind the signature to the message. Thus, we eliminate the need for some of the tools used by the constructions in [29, 10]. In addition, we weaken the properties required from other building blocks, which improves the efficiency of the resulting constructions while offering stronger security.

Our construction requires two NIZK proof systems $\mathcal{NIZK}_1$ and $\mathcal{NIZK}_2$, a selective-tag weakly IND-CCA secure (i.e. ST-WIND-CCA secure) tag-based encryption scheme $\mathcal{TPKE}$, a tagged signature scheme $\mathcal{TS}$, an existentially unforgeable digital signature scheme $\mathcal{WDS}$ that is secure against a weak chosen-message attack, and a strongly unforgeable one-time signature scheme $\mathcal{OTS}$. Additionally, we require two collision-resistant hash functions $\hat{\mathcal{H}} : \{0,1\}^* \to \mathcal{T}_{\mathcal{TPKE}}$ and $\mathcal{H} : \{0,1\}^* \to \mathcal{M}_{\mathcal{OTS}}$.

We require that $\mathcal{NIZK}_1$ is a proof of knowledge. Note that it is sufficient for $\mathcal{WDS}$ to be existentially unforgeable against a weak chosen-message attack as we will use this scheme to sign the verification keys of the one-time signature scheme $\mathcal{OTS}$.

The Setup algorithm generates two separate common reference strings crs_1 and crs_2 for the NIZK systems $\mathcal{NIZK}_1$ and $\mathcal{NIZK}_2$, respectively. It also generates a key pair $(\mathsf{epk}, \mathsf{esk})$ for the tag-based encryption scheme $\mathcal{TPKE}$. The public parameters of the system is set to $\mathsf{pp} := (1^\lambda, \mathsf{crs}_1, \mathsf{crs}_2, \mathsf{epk}, \hat{\mathcal{H}}, \mathcal{H})$. The tracing authority's key is set to $\mathsf{tk} := \mathsf{esk}$.

When a new attribute authority joins the system, it creates a verification/secret key pair $(\mathsf{avk}_{\mathsf{aid}}, \mathsf{ask}_{\mathsf{aid}})$ for the tagged signature scheme $\mathcal{TS}$. When a user joins the system, she generates a verification/secret key pair $(\mathbf{uvk}[\mathsf{uid}], \mathbf{usk}[\mathsf{uid}])$ for the digital signature scheme $\mathcal{WDS}$.

To generate a signing key for attribute $\alpha \in \mathbb{A}$ for user uid, the managing attribute authority signs the user's public key $\mathbf{uvk}[\mathsf{uid}]$ (used as tag) along with the attribute α using her secret tagged signature signing key. The resulting signature σ_α is used as the secret key $\mathsf{sk}_{\mathsf{uid},\alpha}$ for that attribute by user uid.

To sign a message m w.r.t. a signing policy $\mathbb{P}$, the user chooses a fresh key pair $(\mathsf{otsvk}, \mathsf{otssk})$ for the one-time signature $\mathcal{OTS}$ and encrypts her public key $\mathbf{uvk}[\mathsf{uid}]$ using the tag-based encryption scheme $\mathcal{TPKE}$ (and some randomness μ) using $\hat{\mathcal{H}}(\mathsf{otsvk})$ as a tag to obtain a ciphertext C_{tbe}. She then signs $\hat{\mathcal{H}}(\mathsf{otsvk})$ using the digital signature scheme $\mathcal{WDS}$ and her personal secret key $\mathbf{usk}[\mathsf{uid}]$ to obtain a signature σ. Using the NIZK system $\mathcal{NIZK}_1$, she then computes a proof π that: she encrypted her public key correctly, she has a signature σ on $\hat{\mathcal{H}}(\mathsf{otsvk})$ that verifies w.r.t. her public key $\mathbf{uvk}[\mathsf{uid}]$, and she has enough attributes on her public key to satisfy the signing predicate $\mathbb{P}$. To prove the latter, we use a span program (see Section 2) represented by the matrix $\mathbf{S}$: the user proves that she knows a secret vector $\boldsymbol{z} \in \mathbb{Z}_p^{|\mathbb{P}|}$ s.t. $\boldsymbol{z}\mathbf{S} = [1, 0, \ldots, 0]$. She also needs to show that she possesses a valid tagged signature on each attribute in the signing predicate $\mathbb{P}$ for which the corresponding element in $\boldsymbol{z}$ is non-zero. For attributes appearing in $\mathbb{P}$ that the signer does not own, she chooses random signatures. Finally, she signs $(\mathcal{H}(m, \mathbb{P}), \pi, C_{\text{tbe}}, \mathsf{otsvk})$ using the one-time signature $\mathcal{OTS}$ to obtain a one-time signature σ_{ots}.

To verify the signature, one just needs to verify the proof π and the one-time signature σ_{ots}. We note here that if $\mathcal{TS}$ and/or $\mathcal{WDS}$ are re-randomizable, one can reveal in the clear the signature components which are independent of $\mathbf{uvk}[\mathsf{uid}]$ after re-randomizing them. This simplifies the NIZK proof π and subsequently improves the efficiency.

To trace a signature, the tracing authority just decrypts the ciphertext C_{tbe} to recover the user's public key $\mathsf{vk}_{\mathsf{uid}}$, and produces a proof π_{Trace} using the NIZK system $\mathcal{NIZK}_2$ to prove that the decryption was done correctly. She then searches in the public key table $\mathbf{uvk}$ to identify the entry matching $\mathsf{vk}_{\mathsf{uid}}$. It returns $(\mathsf{uid}, \pi_{\mathsf{Trace}})$ if such entry exists, or $(0, \pi_{\mathsf{Trace}})$ otherwise. To verify the tracing correctness, the judge just needs to verify the validity of the NIZK proof π_{Trace}.

The construction is in Fig. 8, whereas the languages associated with the NIZK proofs used in the construction are as follows, where for clarity we underline the elements of the witness:

$$\begin{aligned}\mathcal{L}_1 : \Big\{ &\big((C_{\text{tbe}}, \hat{\mathcal{H}}(\mathsf{otsvk}), \mathsf{epk}, \{\mathsf{avk}_{\mathsf{aid}(\alpha_i)}\}_{i=1}^{|\mathbb{P}|}, \{\alpha_i\}_{i=1}^{|\mathbb{P}|}), (\mathbf{uvk}[\mathsf{uid}], \mu, \boldsymbol{z}, \{\sigma_{\alpha_i}\}_{i=1}^{|\mathbb{P}|})\big) : \\ &\Big(\underline{\boldsymbol{z}}\mathbf{S} = [1, 0, \ldots, 0] \bigwedge_{i=1}^{|\mathbb{P}|} \text{if } \underline{z_i} \neq 0 \Rightarrow \mathcal{TS}.\mathsf{Verify}(\mathsf{avk}_{\mathsf{aid}(\alpha_i)}, \underline{\mathbf{uvk}[\mathsf{uid}]}, \alpha_i, \underline{\sigma_{\alpha_i}}) = 1\Big) \\ &\wedge\ \mathcal{WDS}.\mathsf{Verify}(\underline{\mathbf{uvk}[\mathsf{uid}]}, \hat{\mathcal{H}}(\mathsf{otsvk}), \underline{\sigma}) = 1\ \wedge\ \mathcal{TPKE}.\mathsf{Enc}(\mathsf{epk}, \hat{\mathcal{H}}(\mathsf{otsvk}), \underline{\mathbf{uvk}[\mathsf{uid}]}; \underline{\mu}) = C_{\text{tbe}} \Big\}.\end{aligned}$$

The witness consists of a user's verification key $\mathbf{uvk}[\mathsf{uid}]$, the randomness μ used in encrypting $\mathbf{uvk}[\mathsf{uid}]$, a vector $\boldsymbol{z} \in \mathbb{Z}_p^{|\mathbb{P}|}$, and signatures $\{\sigma_{\alpha_i}\}_{i=1}^{|\mathbb{P}|}$ such that: the span program $\mathbf{S}$ verifies w.r.t. to $\boldsymbol{z}$ and for every non-zero element z_i, the tagged signature σ_{α_i} on $\mathbf{uvk}[\mathsf{uid}]$ (as a tag) and the attribute α_i (as a message) verifies w.r.t. the corresponding attribute authority verification key, the signature σ on $\hat{\mathcal{H}}(\mathsf{otsvk})$ verifies w.r.t. to the user's public key $\mathbf{uvk}[\mathsf{uid}]$, and the ciphertext C_{tbe} is the encryption of $\mathbf{uvk}[\mathsf{uid}]$ using $\hat{\mathcal{H}}(\mathsf{otsvk})$ as a tag (and the randomness μ) under the public key epk.

$$\begin{aligned}\mathcal{L}_2 : \Big\{ \big((C_{\text{tbe}}, \hat{\mathcal{H}}(\mathsf{otsvk}), \mathsf{epk}, \mathsf{vk}_{\mathsf{uid}}), (\mathsf{tk}, \rho)\big) :\ &\mathcal{TPKE}.\mathsf{KeyGen}(1^\lambda; \underline{\rho}) = (\mathsf{epk}, \underline{\mathsf{tk}}) \\ &\wedge\ \mathcal{TPKE}.\mathsf{Dec}(\underline{\mathsf{tk}}, \hat{\mathcal{H}}(\mathsf{otsvk}), C_{\text{tbe}}) = \mathsf{vk}_{\mathsf{uid}} \Big\}.\end{aligned}$$

The witness consists of the tracing key, i.e. the decryption key for $\mathcal{TPKE}$, and the randomness ρ (if any) used in the key generation of $\mathcal{TPKE}$ s.t. the encryption/decryption key pair is correct and C_{tbe} decrypts to $\mathsf{vk}_{\mathsf{uid}}$.

Theorem 1. *The construction in Fig. 8 is a secure decentralized traceable attribute-based signature if the building blocks are secure w.r.t. their security requirements.*

The full proof of this Theorem can be found in Appendix B.

Next, we present two instantiations of the generic framework in the standard model.

6 Instantiations in the Standard Model

In this section, we provide two example instantiations in the standard model and compare their efficiency to existing constructions.

6.1 Instantiation I

Here we instantiate the tagged signature $\mathcal{TS}$ using the re-randomizable AGHO1 signature scheme (see Fig. 4 (Left)) and instantiate the signature scheme $\mathcal{WDS}$ and the one-time signature $\mathcal{OTS}$ using the weak and full Boneh-Boyen signature schemes, respectively. We instantiate both proof systems $\mathcal{NIZK}_1$ and $\mathcal{NIZK}_2$ using the Groth-Sahai proof system and use the Type-3 asymmetric variant of Kiltz's tag-based encryption scheme as illustrated in Fig. 7 to instantiate $\mathcal{TPKE}$.

Let $\mathbf{S} \in \mathbb{Z}_p^{|\mathbb{P}|,\beta}$ be the span program for $\mathbb{P}$. To sign, the signer provides the following proofs:

- To prove that $\boldsymbol{z}\mathbf{S} = [1, 0, \ldots, 0]$, the signer proves the following linear equations:

$$\sum_{i=1}^{|\mathbb{P}|}(\underline{z_i}\tilde{S}_{i,1}) = 1 \qquad\qquad \sum_{i=1}^{|\mathbb{P}|}(\underline{z_i}\tilde{S}_{i,j}) = 0, \text{ for } j = 2, \ldots, \beta$$

- To prove if $\underline{z_i} \neq 0 \Rightarrow \mathcal{TS}.\mathsf{Verify}(\mathsf{avk}_{\mathsf{aid}(\alpha_i)}, \underline{\mathbf{uvk}[\mathsf{uid}]}, \alpha_i, \underline{\sigma_{\alpha_i}}) = 1$, where $\sigma_{\alpha_i} = (A'_i, B'_i, \tilde{D}'_i) \in \mathbb{G}_1^2 \times \mathbb{G}_2$ and $\mathsf{avk}_{\mathsf{aid}(\alpha_i)} = (X_{i,1}, X_{i,2}, \tilde{Y}_i) \in \mathbb{G}_1^2 \times \mathbb{G}_2$. The signer re-randomizes σ_{α_i} by choosing $a' \leftarrow \mathbb{Z}_p^*$ and computing $\sigma_{\alpha_i} := (A_i, B_i, \tilde{D}_i) = (A_i'^{a'}, B_i'^{a'}, \tilde{D}_i'^{\frac{1}{a'}})$, and proves the following

$$\underline{\breve{\tilde{D}}_i} = \underline{\tilde{D}_i^{z_i}} \qquad \underline{\breve{\tilde{Y}}_i} = \underline{\tilde{Y}_i^{z_i}} \qquad \underline{\breve{\mathsf{vk}}_i} = \underline{\widetilde{\mathbf{uvk}}[\mathsf{uid}]^{z_i}} \qquad \underline{\breve{\tilde{G}}_i} = \underline{\tilde{G}^{z_i}}$$

$$e(A_i, \underline{\breve{\tilde{Y}}_i}) = e(\underline{B_i}, \underline{\breve{\tilde{G}}_i}) \qquad\qquad e(A_i, \underline{\breve{\tilde{D}}_i})e(X_{i,1}, \underline{\breve{\mathsf{vk}}_i})e(X_{i,2}, \underline{\breve{\tilde{G}}_i}^{\alpha_i}) = e(G, \underline{\breve{\tilde{G}}_i})$$

  Note here that A_i is independent of A'_i and $\mathbf{uvk}[\mathsf{uid}]$. The same applies to B_i after the re-randomization, however, we cannot afford to reveal B_i in the clear since for attributes the user does not own, she needs to simulate signatures. More precisely, choosing a random pair (A_i, B_i) satisfying the first pairing-product equation is impossible without knowledge of the secret signing key of the authority (since B_i and $\tilde{Y}_i$ lie in different groups). Thus, we hide B_i and just reveal A_i.
  Also, note that the verifier can on her own compute a Groth-Sahai commitment to the value $\breve{\tilde{G}}_i^{\alpha_i}$ by computing $\mathcal{C}_{\breve{\tilde{G}}_i}^{\alpha_i}$, where $\mathcal{C}_{\breve{\tilde{G}}_i}$ is the Groth-Sahai commitment (which is ElGamal ciphertext) to $\breve{\tilde{G}}_i$. This saves the signer producing proofs for the correct computation of such values and hence improves the efficiency. In addition, the way we express the witness of the equations only requires committing to the elements of the vector $\boldsymbol{z}$ in $\mathbb{G}_1$, which further improves the efficiency.

- To prove that $\mathcal{WDS}.\mathsf{Verify}(\underline{\widetilde{\mathbf{uvk}}[\mathsf{uid}]}, \hat{\mathcal{H}}(\mathsf{otsvk}), \underline{\sigma}) = 1$, the signer proves that

$$e(\underline{\sigma}, \underline{\widetilde{\mathbf{uvk}}[\mathsf{uid}]})e(\underline{\sigma}, \tilde{G}^{\hat{\mathcal{H}}(\mathsf{otsvk})})e(\underline{G}, \tilde{G}) = 1 \qquad\qquad \underline{G} - G = 0$$

- To prove $\mathcal{TPKE}.\mathsf{Enc}(\mathsf{epk}, \hat{\mathcal{H}}(\mathsf{otsvk}), \underline{\widetilde{\mathbf{uvk}}[\mathsf{uid}]}; (\underline{r_1}, \underline{r_2})) = C_{\mathsf{tbe}}$, the signer proves she computed the ciphertext $(\tilde{C}_1, \tilde{C}_2, \tilde{C}_3, C_4, C_5) = \big(\tilde{F}^{r_1}, \tilde{H}^{r_2}, \tilde{G}^{r_1+r_2} \cdot \widetilde{\mathbf{uvk}}[\mathsf{uid}], (G^{\hat{\mathcal{H}}(\mathsf{otsvk})} \cdot K)^{r_1}, (G^{\hat{\mathcal{H}}(\mathsf{otsvk})} \cdot L)^{r_2}\big)$ correctly. It is sufficient to prove that $\tilde{C}_1$, $\tilde{C}_2$ and $\tilde{C}_3$ were computed correctly and the rest can be verified by checking that $e(C_4, \tilde{F}) = e(G^{\hat{\mathcal{H}}(\mathsf{otsvk})} \cdot K, \tilde{C}_1)$ and $e(C_5, \tilde{H}) = e(G^{\hat{\mathcal{H}}(\mathsf{otsvk})} \cdot L, \tilde{C}_2)$. Thus, this requires proving

  $$\tilde{C}_1 = \tilde{F}^{\underline{r_1}} \qquad \tilde{C}_2 = \tilde{H}^{\underline{r_2}} \qquad \tilde{C}_3 = \tilde{G}^{\underline{r_1}} \cdot \tilde{G}^{\underline{r_2}} \cdot \underline{\widetilde{\mathbf{uvk}}[\mathsf{uid}]}$$

  Note that we have transposed the roles of $\mathbb{G}_1$ and $\mathbb{G}_2$ in the tag-based encryption instantiation since the plaintext (i.e. the user's public key) is in $\mathbb{G}_2$ rather than $\mathbb{G}_1$. Thus, the security of the underlying tag-based encryption scheme used here rests on SDLIN in $\mathbb{G}_2$ instead of $\mathbb{G}_1$.

The total size of the signature is $\mathbb{G}_1^{26\cdot|\mathbb{P}|+21} + \mathbb{G}_2^{24\cdot|\mathbb{P}|+13} + \mathbb{Z}_p^{\beta+3}$.

The proof π_{Trace} is of size $\mathbb{G}_1^4 \times \mathbb{G}_2^3$ and requires proving the following equations

$$\tilde{G}^{\underline{f}} = \tilde{F} \qquad \tilde{G}^{\underline{h}} = \tilde{H} \qquad \tilde{C}_3 \cdot \tilde{C}_1^{-1/\underline{f}} \cdot \tilde{C}_2^{-1/\underline{h}} = \tilde{\mathsf{vk}}_{\mathsf{uid}}$$

The proof for the following Theorem follows from that of Theorem 1.

Theorem 2. *The instantiation is secure if the AGHO1 signature scheme is unforgeable and the assumptions SDLIN in $\mathbb{G}_2$, SXDH, and q-SDH all hold.*

6.2 Instantiation II

Our aim here is to get an efficient instantiation that is based on falsifiable intractability assumptions [30]. We instantiate the tagged signature using the AGHO2 signature scheme which is based on a falsifiable assumption, as shown in Fig. 4 (Right). Since the tag space (i.e. the user's public key) of this instantiation of the tagged signature is $\mathbb{G}_1$, we needed to transpose the groups from which the public key and the signature components of $\mathcal{WDS}$ are chosen. More precisely, we instantiate $\mathcal{WDS}$ with the weak Boneh-Boyen signature scheme where the public key is $X := G^x$ and the signature is $\tilde{\sigma} := \tilde{G}^{\frac{1}{x+m}}$. To this end, we also transpose the groups used in the q-SDH assumption. We instantiate the tag-based encryption scheme using the scheme in Fig. 7. The rest of the tools remain the same as in Instantiation I.

Let $\mathbf{S} \in \mathbb{Z}_p^{|\mathbb{P}|,\beta}$ be the span program for $\mathbb{P}$. To sign, the signer provides the following proofs:

- To prove that $\boldsymbol{z}\mathbf{S} = [1, 0, \ldots, 0]$, the signer proves the following linear equations:

  $$\sum_{i=1}^{|\mathbb{P}|} (\underline{\tilde{z}_i} S_{i,1}) = 1 \qquad \sum_{i=1}^{|\mathbb{P}|} (\underline{\tilde{z}_i} S_{i,j}) = 0, \text{ for } j = 2, \ldots, \beta$$

- To prove if $\underline{z_i} \neq 0 \Rightarrow \mathcal{TS}.\mathsf{Verify}(\mathsf{avk}_{\mathsf{aid}(\alpha_i)}, \underline{\mathbf{uvk}[\mathsf{uid}]}, \alpha_i, \underline{\sigma_{\alpha_i}}) = 1$, where $\sigma_{\alpha_i} = (A_i, B_i, R_i, \tilde{D}_i) \in \mathbb{G}_1^3 \times \mathbb{G}_2$ and $\mathsf{avk}_{\mathsf{aid}(\alpha_i)} = (\tilde{W}_i, \tilde{X}_i, \tilde{Y}_{i,1}, \tilde{Y}_{i,2}, \tilde{Y}_{i,3}) \in \mathbb{G}_2^5$, the signer proves:

  $$\underline{\breve{A}_i} = \underline{A_i}^{\underline{z_i}} \qquad \underline{\breve{B}_i} = \underline{B_i}^{\underline{z_i}} \qquad \underline{\breve{R}_i} = \underline{R_i}^{\underline{z_i}} \qquad \underline{\breve{G}_i} = G^{\underline{z_i}} \qquad \underline{\breve{\mathsf{vk}}_i} = \underline{\mathbf{uvk}[\mathsf{uid}]}^{\underline{z_i}}$$

  $$e(\underline{\breve{A}_i}, \underline{\tilde{D}}) = e(\underline{\breve{G}_i}, \tilde{G}) \qquad e(\underline{\breve{G}_i}, \tilde{X}_i) = e(\underline{\breve{A}_i}, \tilde{W}_i) e(\underline{\breve{B}_i}, \tilde{G}) e(\underline{\breve{R}_i}, \tilde{Y}_{i,1}) e(\underline{\breve{\mathsf{vk}}_i}, \tilde{Y}_{i,2}) e(\underline{\breve{G}_i}^{\alpha_i}, \tilde{Y}_{i,3})$$

  Again, the verifier can on her own compute a Groth-Sahai commitment to the value $\breve{G}_i^{\alpha_i}$. Also, the way we express the witness of the equations only requires committing to the elements of the vector $\boldsymbol{z}$ in $\mathbb{G}_2$. This further improves the efficiency.

- To prove that $\mathcal{WDS}.\mathsf{Verify}(\underline{\mathbf{uvk}[\mathsf{uid}]}, \hat{\mathcal{H}}(\mathsf{otsvk}), \underline{\tilde{\sigma}}) = 1$, the signer needs to prove that

$$e(\underline{\mathbf{uvk}[\mathsf{uid}]}, \underline{\tilde{\sigma}})e(G^{\hat{\mathcal{H}}(\mathsf{otsvk})}, \underline{\tilde{\sigma}})e(\underline{G}, \tilde{G}) = 1 \qquad\qquad \underline{G} - G = 0$$

- To prove $\mathcal{TPKE}.\mathsf{Enc}(\mathsf{epk}, \hat{\mathcal{H}}(\mathsf{otsvk}), \underline{\mathbf{uvk}[\mathsf{uid}]}; (\underline{r_1}, \underline{r_2})) = C_{\mathsf{tbe}}$, the signer proves she computed the ciphertext $(C_1, C_2, C_3, \tilde{C}_4, \tilde{C}_5) = \big(F^{r_1}, H^{r_2}, G^{r_1+r_2} \cdot \mathbf{uvk}[\mathsf{uid}], (\tilde{G}^{\hat{\mathcal{H}}(\mathsf{otsvk})} \cdot \tilde{K})^{r_1}, (\tilde{G}^{\hat{\mathcal{H}}(\mathsf{otsvk})} \cdot \tilde{L})^{r_2}\big)$ correctly. It is sufficient to prove that C_1, C_2 and C_3 were computed correctly and the rest can be verified by checking that $e(F, \tilde{C}_4) = e(C_1, \tilde{G}^{\hat{\mathcal{H}}(\mathsf{otsvk})} \cdot \tilde{K})$ and $e(H, \tilde{C}_5) = e(C_2, \tilde{G}^{\hat{\mathcal{H}}(\mathsf{otsvk})} \cdot \tilde{L})$. Thus, this requires proving

$$C_1 = F^{\underline{\tilde{r}_1}} \qquad\qquad C_2 = H^{\underline{\tilde{r}_2}} \qquad\qquad C_3 = G^{\underline{\tilde{r}_1}} \cdot G^{\underline{\tilde{r}_2}} \cdot \underline{\mathbf{uvk}[\mathsf{uid}]}$$

The total size of the signature is $\mathbb{G}_1^{30\cdot|\mathbb{P}|+16} + \mathbb{G}_2^{30\cdot|\mathbb{P}|+18} + \mathbb{Z}_p^{\beta+3}$.

The proof π_{Trace} is of size $\mathbb{G}_1^3 \times \mathbb{G}_2^4$ and requires proving the following equations

$$G^{\underline{\tilde{f}}} = F \qquad\qquad G^{\underline{\tilde{h}}} = H \qquad\qquad C_3 \cdot C_1^{-1/\underline{\tilde{f}}} \cdot C_2^{-1/\underline{\tilde{h}}} = \mathsf{vk}_{\mathsf{uid}}$$

The proof for the following Theorem follows from that of Theorem 1.

Theorem 3. *The instantiation is secure if the assumptions SDLIN in $\mathbb{G}_1$, q-SDH, q-AGHO, and SXDH all hold.*

We end the section by noting that the verification algorithms in the above two instantiations can be made more efficient by applying batch verification techniques to Groth-Sahai proofs [15, 5].

6.3 Efficiency Comparison

We compare the efficiency of our instantiations with that of existing constructions in Table 1. Note here that the construction in [11] is for the single attribute-authority setting and that our constructions are secure w.r.t. to a stronger security model than those in [11, 10].

Construction	Signature Size	Model	Setting	No. of Authorities
[11]	$\mathbb{G}^{\vert\mathbb{P}\vert+\beta+7}$	ROM	Composite Order	Single
[10]	$\mathbb{G}_1^{34\cdot\vert\mathbb{P}\vert+28} + \mathbb{G}_2^{32\cdot\vert\mathbb{P}\vert+32} + \mathbb{Z}_p^{\beta+1}$	Standard	Prime Order	Multiple
Instantiation I	$\mathbb{G}_1^{26\cdot\vert\mathbb{P}\vert+21} + \mathbb{G}_2^{24\cdot\vert\mathbb{P}\vert+13} + \mathbb{Z}_p^{\beta+3}$	Standard	Prime Order	Multiple
Instantiation II	$\mathbb{G}_1^{30\cdot\vert\mathbb{P}\vert+16} + \mathbb{G}_2^{30\cdot\vert\mathbb{P}\vert+18} + \mathbb{Z}_p^{\beta+3}$	Standard	Prime Order	Multiple

Table 1. Efficiency comparison

7 Conclusion

We have presented a new security model for decentralized traceable attribute-based signatures that is stronger than existing models. In doing so, we have circumvented some shortcomings in existing models. We have also provided a generic framework for obtaining constructions secure w.r.t. our strong model and provided concrete instantiations in the standard model which outperform existing constructions.

Acknowledgments. The author of this work was supported by ERC Advanced Grant ERC-2010-AdG-267188-CRIPTO and EPSRC via grant EP/H043454/1.

References

1. M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev and M. Ohkubo. Structure-preserving signatures and commitments to group elements. In *Crypto 2010*, Springer LNCS 6223, 209–236, 2010.
2. M. Abe, J. Groth, K. Haralambiev and M. Ohkubo. Optimal Structure-Preserving Signatures in Asymmetric Bilinear Groups. In *Crypto 2011*, Springer LNCS 6841, 649–666, 2011.
3. M. Bellare and P. Rogaway. Random oracles are practical: A Paradigm for Designing Efficient Protocols. In *ACM-CCS 1993*, ACM, pp. 62–73.
4. M. Bellare, H. Shi and C. Zhang. Foundations of group signatures: The case of dynamic groups. In *CT-RSA 2005*, Springer LNCS 3376, 136–153, 2005.
5. O. Blazy, G. Fuchsbauer, M. Izabach'ene, A. Jambert, H. Sibert and D. Vergnaud. Batch Groth-Sahai. In *ACNS 2010*, Springer LNCS 6123, 218–235, 2010.
6. M. Blum, P. Feldman and S. Micali. Non-interactive zero-knowledge and its applications. In *STOC 1988*, 103–112, 1988.
7. D. Boneh and X. Boyen. Short Signatures Without Random Oracles and the SDH Assumption in Bilinear Groups. In Journal of Cryptology, volume 21(2), 149–177, 2008.
8. R. Bobba, O. Fatemieh, F. Khan, C.A. Gunter and H. Khurana. Using Attribute-Based Access Control to Enable Attribute-Based Messaging. In *ACSAC 2006*, IEEE Computer Society 3027, 403–413, 2006.
9. D. Boneh, X. Boyen and H. Shacham. Short Group Signatures. In *CRYPTO 2004*, Springer LNCS 3152, 227–242, 2004.
10. A. El Kaafarani, E. Ghadafi and D. Khader. Decentralized Traceable Attribute-Based Signatures. In *CT-RSA 2014*, Springer LNCS 8366, 327–348, 2014.
11. A. Escala, J. Herranz and P. Morillo. Revocable Attribute-Based Signatures with Adaptive Security in the Standard Model. In *AFRICACRYPT 2011*, Springer LNCS 6737, 224–241, 2011.
12. K.B. Frikken, J. Li and M.J. Atallah. Trust negotiation with hidden credentials, hidden policies, and policy cycles. In *NDSS 2006*, The Internet Society, 157–172, 2006.
13. Ma. Gagné, S. Narayan and R. Safavi-Naini. Short Pairing-Efficient Threshold-Attribute-Based Signature. In *Pairing 2012*, Springer LNCS 7708, 295–313, 2012.
14. S. Galbraith, K. Paterson and N.P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, **156**, 3113–3121, 2008.
15. E. Ghadafi, N.P. Smart and B. Warinschi. Practical zero-knowledge proofs for circuit evaluation. In *Coding and Cryptography: IMACC 2009*, Springer LNCS 5921, 469–494, 2009.
16. E. Ghadafi, N.P. Smart and B. Warinschi. Groth-Sahai proofs revisited. In *PKC 2010*, Springer LNCS 6056, 177–192, 2010.
17. V. Goyal, O. Pandey, A. Sahai and B. Waters. Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data. In *ACM-CCS 2006*, ACM ,89–98 , 2006.
18. J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In SIAM Journal on Computing, volume 41(5), 1193–1232, 2012.
19. J. Herranz, F. Laguillaumie, B. Libert, and C. Ráfols. Short Attribute-Based Signatures for Threshold Predicates. In *CT-RSA 2012*, Springer LNCS 7178, 51–67, 2012.
20. M. Karchmer and A. Wigderson. On span programs. In *8th IEEE Structure in Complexity Theory*, 102–111, 1993.
21. S.A. Kakvi. Efficient fully anonymous group signatures based on the Groth group signature scheme. Masters thesis, University College London, 2010. `http://www5.rz.rub.de:8032/mam/foc/content/publ/thesis_kakvi10.pdf`.
22. D. Khader. Attribute Based Group Signatures with Revocation. In *Cryptology ePrint Archive, Report 2007/241*, `http://eprint.iacr.org/2007/241.pdf`.
23. D. Khader, L. Chen, J. H. Davenport. Certificate-Free Attribute Authentication. In *Cryptography and Coding: IMACC 2009*, Springer LNCS 5921, 301–325, 2009.
24. E. Kiltz. Chosen-Ciphertext Security from Tag-Based Encryption. In *TCC 2006*, Springer LNCS 3876, 581–600, 2006.
25. J. Li, M. H. Au, W. Susilo, D. Xie and K. Ren. Attribute-based signature and its applications. In *ASIACCS '10*, ACM, 60-69, 2010.
26. J. Li and K. Kim. Attribute-Based Ring Signatures. In *Cryptology ePrint Archive, Report 2008/394*, `http://eprint.iacr.org/2008/394.pdf`.
27. P. MacKenzie, M.K. Reiter and K. Yang. Alternatives to Non-malleability: Definitions, Constructions, and Applications. In *TCC 2004*, Springer LNCS 2951, 171–190, 2004.
28. H.K. Maji, M. Prabhakaran and M. Rosulek. Attribute-Based Signatures: Achieving Attribute-Privacy and Collusion-Resistance. In *Cryptology ePrint Archive, Report 2008/328*, `http://eprint.iacr.org/2008/328.pdf`.
29. H.K. Maji, M. Prabhakaran and M. Rosulek. Attribute-Based Signatures. In *CT-RSA 2011*, Springer LNCS 6558, 376–392, 2011.
30. M. Naor. On cryptographic assumptions and challenges. In *CRYPTO 2003*, Springer LNCS 2729, 96–109, 2003.
31. T. Okamoto and K. Takashima. Efficient Attribute-Based Signatures for Non-Monotone Predicates in the Standard Model. In *PKC 2011*, Springer LNCS 6571, 35–52, ,2011.
32. T. Okamoto and K. Takashima. Decentralized Attribute-Based Signatures. In *PKC 2013*, Springer LNCS 7778, 125–142, 2012.

33. A. Sahai and B. Waters. Fuzzy Identity-Based Encryption In *EUROCRYPT 2005*, Springer LNCS 3494, 457–473, 2005.
34. Y. Sakai, J. C. N. Schuldt, K. Emura, G. Hanaoka and K. Ohta. On the Security of Dynamic Group Signatures: Preventing Signature Hijacking. In *PKC 2012*, Springer LNCS 7293, 715–732, 2012.
35. S. F. Shahandashti and R. Safavi-Naini. Threshold Attribute-Based Signatures and Their Application to Anonymous Credential Systems. In *AFRICACRYPT 2009*, Springer LNCS 5580, 198–216, 2009.

A Properties of Non-Interactive Zero-Knowledge Proofs

The properties we require from a non-interctive zero-knowledge proof system are:

- **(Perfect) Completeness:** $\forall \lambda \in \mathbb{N}$, $\forall (x, w) \in \mathcal{R}$, we have

$$\Pr\left[(\mathsf{crs}, \mathsf{xk}) \leftarrow \mathsf{Setup}(1^{\lambda}); \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w) : \mathsf{Verify}(\mathsf{crs}, x, \pi) = 1\right] = 1 \ .$$

- **Soundness:** $\forall \lambda \in \mathbb{N}$, $\forall x \notin \mathcal{L}$, we have for all adversaries $\mathcal{B}$

$$\Pr\left[(\mathsf{crs}, \mathsf{xk}) \leftarrow \mathsf{Setup}(1^{\lambda}); \pi \leftarrow \mathcal{B}(\mathsf{crs}, x) : \mathsf{Verify}(\mathsf{crs}, x, \pi) = 1\right] \leq 2^{-\lambda} \ .$$

  If the above probability is 0, we say the system has *perfect soundness*.
- **Knowledge Extraction:** A proof system is a *Proof of Knowledge* if there exists an efficient extractor algorithm Extract which can extract the witness from any proof the adversary outputs. Note that knowledge extraction implies soundness. More formally, for all adversaries $\mathcal{B}$, we have

$$\Pr\Big[(\mathsf{crs}, \mathsf{xk}) \leftarrow \mathsf{Setup}(1^{\lambda}); (x, \pi) \leftarrow \mathcal{B}(\mathsf{crs}); w \leftarrow \mathsf{Extract}(\mathsf{crs}, \mathsf{xk}, x, \pi)$$
$$: \mathsf{Verify}(\mathsf{crs}, x, \pi) = 0 \text{ OR } (x, w) \in \mathcal{R}] \leq 1 - \nu(\lambda) \ .$$

  If the above probability is 1, we say the system has *perfect knowledge extraction*.
- **Zero-Knowledge:** The system is *zero-knowledge* if $\forall (x, w) \in \mathcal{R}$, we have for all PPT adversaries $\mathcal{B}$

$$\Pr\left[(\mathsf{crs}_{\mathsf{Sim}}, \mathsf{tr}) \leftarrow \mathsf{SimSetup}(1^{\lambda}) : \mathcal{B}^{\mathsf{Sim}(\mathsf{crs}_{\mathsf{Sim}}, \mathsf{tr}, \cdot, \cdot)}(\mathsf{crs}_{\mathsf{Sim}}) = 1\right]$$
$$\approx \Pr\left[(\mathsf{crs}, \mathsf{xk}) \leftarrow \mathsf{Setup}(1^{\lambda}) : \mathcal{B}^{\mathsf{Prove}(\mathsf{crs}, \cdot, \cdot)}(\mathsf{crs}) = 1\right] ,$$

  where $\mathsf{Sim}(\mathsf{crs}_{\mathsf{Sim}}, \mathsf{tr}, x, w)$ outputs $\mathsf{SimProve}(\mathsf{crs}_{\mathsf{Sim}}, \mathsf{tr}, x)$ if $(x, w) \in \mathcal{R}$ or $\perp$ otherwise.

B Proof of Theorem 1

Proof. Correctness of the construction follows from that of the underlying building blocks.

Lemma 1. *The construction is non-frameable if $\mathcal{NIZK}_1$ and $\mathcal{NIZK}_2$ proof systems are sound, the hash functions $\hat{\mathcal{H}}$ and $\mathcal{H}$ are collision-resistant, and the one-time signature $\mathcal{OTS}$ and the digital signature $\mathcal{WDS}$ are existentially unforgeable.*

Proof. We start by initiating both $\mathcal{NIZK}_1$ and $\mathcal{NIZK}_2$ proof systems in the soundness setting which ensures that the adversary cannot break non-frameability by faking proofs for false statements. We show that if there exists an adversary $\mathcal{B}$ that breaks non-frameability, we can construct adversaries: $\mathcal{F}_1$ against the unforgeability of the digital signature scheme $\mathcal{WDS}$, adversary $\mathcal{F}_2$ against the strong unforgeability of the one-time signature scheme $\mathcal{OTS}$, and adversaries $\mathcal{F}_3$ and $\mathcal{F}_4$ against the collision-resistance of $\mathcal{H}$ and $\hat{\mathcal{H}}$, respectively.

$$\begin{aligned}\mathsf{Adv}^{\mathrm{NF}}_{\mathcal{DTABS},\mathcal{B}}(\lambda) \leq & \kappa(\lambda) \cdot \mathsf{Adv}^{\mathrm{Unforge}}_{\mathcal{WDS},\mathcal{F}_1}(\lambda) + \delta(\lambda) \cdot \mathsf{Adv}^{\mathrm{Unforge}}_{\mathcal{OTS},\mathcal{F}_2}(\lambda) + \mathsf{Adv}^{\mathrm{CR}}_{\mathcal{H},\mathcal{F}_3}(\lambda) + \mathsf{Adv}^{\mathrm{CR}}_{\hat{\mathcal{H}},\mathcal{F}_4}(\lambda) \\ & + \mathsf{Adv}^{\mathrm{Sound}}_{\mathcal{NIZK}_1,\mathcal{F}_5}(\lambda) + \mathsf{Adv}^{\mathrm{Sound}}_{\mathcal{NIZK}_2,\mathcal{F}_6}(\lambda),\end{aligned}$$

where $\kappa(\lambda)$ and $\delta(\lambda)$ are polynomials in λ representing an upper bound on the number of honest users and sign queries, respectively, $\mathcal{B}$ is allowed to make in the game.

The collision-resistance of $\mathcal{H}$ ensures that $\mathcal{B}$ has a negligible probability in finding pairs $(m^*, \mathbb{P}^*) \neq (m, \mathbb{P})$ s.t. $\mathcal{H}(m^*, \mathbb{P}^*) = \mathcal{H}(m, \mathbb{P})$. If this is not the case, we can use $\mathcal{B}$ to construct an adversary $\mathcal{F}_3$ that breaks the collision-resistance of $\mathcal{H}$. Similarly, the collision-resistance of $\hat{\mathcal{H}}$ ensures that $\mathcal{B}$ has a negligible probability in finding two different one-time signature keys $\mathsf{otsvk} \neq \mathsf{otsvk}'$ s.t. $\hat{\mathcal{H}}(\mathsf{otsvk}) = \hat{\mathcal{H}}(\mathsf{otsvk}')$. If this is not the case, we can use $\mathcal{B}$ to construct an adversary $\mathcal{F}_4$ that breaks the collision-resistance of $\hat{\mathcal{H}}$. Thus, from now on we assume that there are no hash collisions.

- Adversary $\mathcal{F}_1$: Adversary $\mathcal{F}_1$ randomly chooses $i \leftarrow \{1, \ldots, \kappa(\lambda)\}$ and guesses that $\mathcal{B}$ will frame user i. We have a probability $\frac{1}{\kappa(\lambda)}$ of guessing the correct user. Let $\eta(\lambda)$ denote the number of sign queries by user i adversary $\mathcal{B}$ makes in the game. $\mathcal{F}_1$ chooses random key pairs $\mathsf{OTSK} := \{(\mathsf{otsvk}, \mathsf{otssk})_j\}_{j=1}^{\eta(\lambda)}$ for the one-time signature. It forwards $\hat{\mathcal{H}}(\mathsf{otsvk}_1), \ldots, \hat{\mathcal{H}}(\mathsf{otsvk}_{\eta(\lambda)})$ to its game and gets back a verification key vk and signatures $\sigma_1, \ldots, \sigma_{\eta(\lambda)}$. Adversary $\mathcal{F}_1$ starts by running $(\mathsf{crs}_1, \mathsf{xk}_1) \leftarrow \mathcal{NIZK}_1.\mathsf{Setup}(1^\lambda)$, $\mathsf{crs}_2 \leftarrow \mathcal{NIZK}_2.\mathsf{Setup}(1^\lambda)$ and choosing the key pair $(\mathsf{epk}, \mathsf{esk})$ for the tag-based encryption scheme $\mathcal{TPKE}$. It then forwards $\mathsf{pp} := (1^\lambda, \mathsf{crs}_1, \mathsf{crs}_2, \mathsf{epk}, \mathcal{H}, \hat{\mathcal{H}})$ and $\mathsf{tk} := \mathsf{esk}$ to $\mathcal{B}$.
  To answer AddA queries, $\mathcal{F}_1$ chooses the secret/verification keys for the authority itself so it can answer any AddAtt queries. To answer AddU queries, for all users other than user i, $\mathcal{F}_1$ chooses the personal key pair for the user itself. However, for user i, it sets its verification key to vk it got from its game (and thus it does not know the corresponding secret key). If in the game $\mathcal{B}$ issues a $\mathsf{RevealU}$ query on user i, $\mathcal{F}_1$ aborts the game.
  To answer Sign queries $(\mathsf{uid}, \mathcal{A}, m, \mathbb{P})$ for any $\mathsf{uid} \neq i$, $\mathcal{F}_1$ first chooses a fresh key pair $(\mathsf{otsvk}', \mathsf{otssk}')$ for the one-time signature $\mathcal{OTS}$ and encrypts $\mathbf{uvk}[\mathsf{uid}]$ using $\hat{\mathcal{H}}(\mathsf{otsvk}')$ as a tag and generates the rest of the signature itself. For the j-th sign query by user i, $\mathcal{F}_1$ uses the j-th key pair in the set OTSK and encrypts $\mathbf{uvk}[\mathsf{uid}]$ using $\hat{\mathcal{H}}(\mathsf{otsvk}_j)$ as a tag, and generates the rest of the signature. The rest of $\mathcal{B}$'s queries are answered normally as in Fig. 1.
  Eventually, when $\mathcal{B}$ outputs its forgery, $\mathcal{F}_1$ uses the $\mathcal{NIZK}_1$'s extraction key xk_1 to extract the witness and returns the signature σ^* on $\hat{\mathcal{H}}(\mathsf{otsvk}^*)$ that is different from all $\mathsf{otsvk}_1, \ldots, \mathsf{otsvk}_{\eta(\lambda)}$ that $\mathcal{F}_1$ has used in answering signing queries if $\mathcal{B}$'s forgery involved framing user i as was guessed by $\mathcal{F}_1$. Otherwise, it aborts.
  By the existential unforgeability of the signature scheme $\mathcal{WDS}$, the probability of $\mathcal{B}$ winning is negligible.
- Adversary $\mathcal{F}_2$: Adversary $\mathcal{F}_2$ gets otsvk^* from its game and has access to an oracle Sign that it uses to obtain a single one-time signature that verify w.r.t. otsvk^* on a message of its choice. It runs $(\mathsf{crs}_1, \mathsf{xk}_1) \leftarrow \mathcal{NIZK}_1.\mathsf{Setup}(1^\lambda)$, $\mathsf{crs}_2 \leftarrow \mathcal{NIZK}_2.\mathsf{Setup}(1^\lambda)$. It chooses a key pair $(\mathsf{epk}, \mathsf{esk})$ for the tag-based encryption scheme $\mathcal{TPKE}$ and forwards $\mathsf{pp} := (1^\lambda, \mathsf{crs}_1, \mathsf{crs}_2, \mathsf{epk}, \mathcal{H}, \hat{\mathcal{H}})$ and $\mathsf{tk} := \mathsf{esk}$ to $\mathcal{B}$.
  To answer AddA queries, $\mathcal{F}_2$ chooses the authority keys itself. To answer AddU queries, $\mathcal{F}_2$ chooses the user's key pair itself. To answer AddAtt queries, $\mathcal{F}_2$ uses the corresponding authorities' secret keys $\mathsf{ask}_{\mathsf{aid}(\alpha)}$ to create the attributes' keys for the user.
  Adversary $\mathcal{F}_2$ randomly chooses $i \leftarrow \{1, \ldots, \delta(\lambda)\}$ and guesses that $\mathcal{B}$'s forgery will involve forging a one-time signature that verifies under otsvk^* used in answering the i-th signing query.

When asked for the j-th Sign query on $(\mathsf{uid}, \mathcal{A}, m, \mathbb{P})$, if $j \neq i$, $\mathcal{F}_2$ chooses a fresh key pair $(\mathsf{otsvk}, \mathsf{otssk})$ for the one-time signature scheme and answers the query by itself. If $j = i$, $\mathcal{F}_2$ encrypts $\mathbf{uvk}[\mathsf{uid}]$ using $\hat{\mathcal{H}}(\mathsf{otsvk}^*)$ (i.e. the verification key it got from its game) as a tag to obtain C_{tbe}. It then generates σ by signing $\hat{\mathcal{H}}(\mathsf{otsvk}^*)$ using $\mathbf{usk}[\mathsf{uid}]$ that it chose, and constructs the proof π. It then forwards $(\mathcal{H}(m, \mathbb{P}), \pi, C_{\text{tbe}}, \mathsf{otsvk}^*)$ as the message to its one-time signing oracle to get a one-time signature σ_{ots}. $\mathcal{F}_2$ then sends the signature Σ to $\mathcal{B}$.
The rest of $\mathcal{B}$'s queries are answered normally as in Fig. 1.
Eventually, when $\mathcal{B}$ outputs its forgery, $\mathcal{F}_2$ aborts if the $\mathcal{B}$'s forgery did not involve forging a one-time signature that verifies w.r.t. otsvk^* it got from its game. The probability that $\mathcal{B}$ forges a one-time signature that verifies w.r.t. otsvk^* is $\frac{1}{\delta(\lambda)}$.
By the strong existential unforgeability of the one-time signature $\mathcal{OTS}$, $\mathcal{B}$ has a negligible advantage in wining this case.

This concludes the proof.

Lemma 2. *The construction is unforgeable if $\mathcal{NIZK}_1$ and $\mathcal{NIZK}_2$ proof systems are sound, the hash functions $\hat{\mathcal{H}}$ and $\mathcal{H}$ are collision-resistant, and the tagged signature $\mathcal{TS}$, and the one-time signature $\mathcal{OTS}$ are existentially unforgeable.*

Proof. We show that if there exists an adversary $\mathcal{B}$ breaking unforgeability, we can construct adversaries: $\mathcal{F}_1$ against the unforgeability of the tagged signature scheme $\mathcal{TS}$, $\mathcal{F}_2$ against the strong unforgeability of the one-time signature scheme $\mathcal{OTS}$, adversaries $\mathcal{F}_3$ and $\mathcal{F}_4$ against the collision-resistance of the hash functions $\hat{\mathcal{H}}$ and $\mathcal{H}$, and adversaries $\mathcal{F}_5$ and $\mathcal{F}_6$ against the soundness of $\mathcal{NIZK}_1$ and $\mathcal{NIZK}_2$ respectively, such that

$$\begin{aligned}\mathsf{Adv}^{\text{Unforge}}_{\mathcal{DTABS},\mathcal{B}}(\lambda) \leq & \kappa(\lambda) \cdot \mathsf{Adv}^{\text{Unforge}}_{\mathcal{TS},\mathcal{F}_1}(\lambda) + \delta(\lambda) \cdot \mathsf{Adv}^{\text{Unforge}}_{\mathcal{OTS},\mathcal{F}_2}(\lambda) + \mathsf{Adv}^{\text{CR}}_{\hat{\mathcal{H}},\mathcal{F}_3}(\lambda) + \mathsf{Adv}^{\text{CR}}_{\mathcal{H},\mathcal{F}_4}(\lambda) \\ & + \mathsf{Adv}^{\text{Sound}}_{\mathcal{NIZK}_1,\mathcal{F}_5}(\lambda) + \mathsf{Adv}^{\text{Sound}}_{\mathcal{NIZK}_2,\mathcal{F}_6}(\lambda),\end{aligned}$$

where $\kappa(\lambda)$ and $\delta(\lambda)$ are polynomials in λ representing an upper bound on the number of honest attribute authorities and sign queries, respectively, $\mathcal{B}$ is allowed to make in the game.

We instantiate both $\mathcal{NIZK}_1$ and $\mathcal{NIZK}_2$ proof systems in the soundness setting and hence the adversary cannot break unforgeability by faking proofs for false statements. By the collision-resistance of $\hat{\mathcal{H}}$, $\mathcal{B}$ has a negligible probability in finding two different one-time signature keys $\mathsf{otsvk} \neq \mathsf{otsvk}'$ s.t. $\hat{\mathcal{H}}(\mathsf{otsvk}) = \hat{\mathcal{H}}(\mathsf{otsvk}')$. If this is not the case, we can use $\mathcal{B}$ to construct an adversary $\mathcal{F}_3$ that breaks the collision-resistance of $\hat{\mathcal{H}}$. Similarly, by the security of $\mathcal{H}$, $\mathcal{B}$ has a negligible probability in finding collisions $(m, \mathbb{P}) \neq (m^*, \mathbb{P}^*)$ s.t. $\mathcal{H}(m, \mathbb{P}) = \mathcal{H}(m^*, \mathbb{P}^*)$. If this is not the case, we can use $\mathcal{B}$ to construct an adversary $\mathcal{F}_4$ that breaks the collision-resistance of $\mathcal{H}$. Thus, from now on we assume that there are no hash collisions.

- Adversay $\mathcal{F}_1$: Adversary $\mathcal{F}_1$ gets the tagged signature scheme's verification key vk from its game and has access to an oracle Sign that it uses to obtain tagged signatures that verify w.r.t. vk on messages and tags of its choice. Adversary $\mathcal{F}_1$ starts by running $(\mathsf{crs}_1, \mathsf{xk}_1) \leftarrow \mathcal{NIZK}_1.\mathsf{Setup}(1^\lambda)$, $\mathsf{crs}_2 \leftarrow \mathcal{NIZK}_2.\mathsf{Setup}(1^\lambda)$ and choosing a key pair $(\mathsf{epk}, \mathsf{esk})$ for the tag-based encryption scheme $\mathcal{TPKE}$. It forwards $\mathsf{pp} := (1^\lambda, \mathsf{crs}_1, \mathsf{crs}_2, \mathsf{epk}, \mathcal{H}, \hat{\mathcal{H}})$ and $\mathsf{tk} := \mathsf{esk}$ to $\mathcal{B}$.
  Adversary $\mathcal{F}_1$ randomly chooses $i \leftarrow \{1, \ldots, \kappa(\lambda)\}$ and guesses that $\mathcal{B}$'s forgery will involve forging an attribute managed by the attribute authority i. To answer AddA queries, for all authorities $j \neq i$, $\mathcal{F}_1$ chooses the secret/verification keys for the authority itself. For authority i, it sets its verification key to vk it got from its game (and thus it does not know the corresponding secret key). If in the game $\mathcal{B}$ asks a RevealA query on authority i, $\mathcal{F}_1$ aborts the game.
  To answer AddU queries, $\mathcal{F}_1$ chooses the user's key pair itself. To answer AddAtt queries, if the user has attributes managed by authority i, it forwards such a query to its Sign oracle; Otherwise, it answers the query itself by using the authorities' secret keys available to it.

To answer Sign queries on $(\mathsf{uid}, \mathcal{A}, m, \mathbb{P})$, $\mathcal{F}_1$ first chooses a fresh key pair $(\mathsf{otsvk}, \mathsf{otssk})$ for the one-time signature $\mathcal{OTS}$ and encrypts $\mathbf{uvk}[\mathsf{uid}]$ using $\hat{\mathcal{H}}(\mathsf{otsvk})$ as a tag and generates the rest of the signature Σ which it then forwards to $\mathcal{B}$. The rest of $\mathcal{B}$'s queries are answered normally as in Fig. 1.
Eventually, when $\mathcal{B}$ outputs its forgery, $\mathcal{F}_1$ uses the $\mathcal{NIZK}_1$'s extraction key xk_1 to extract the witness and returns the tagged signature on $\mathbf{uvk}[\mathsf{uid}^*]$ and the attribute α^* if $\mathcal{B}$'s forgery involved forging a tagged signature. Otherwise, it aborts. $\mathcal{F}_1$ also aborts if the forgery does not involve forged attributes managed by authority i that $\mathcal{F}_1$ has guessed. The probability of $\mathcal{F}_1$ guessing the correct authority is $\frac{1}{\kappa(\lambda)}$.
By the existential unforgeability of the tagged signature scheme, the probability of $\mathcal{B}$ winning is negligible.

- Adversary $\mathcal{F}_2$: Adversary $\mathcal{F}_2$ gets otsvk^* from its game and has access to an oracle Sign that it uses to obtain a single one-time signature that verifies w.r.t. otsvk^* on a message of its choice. It runs $(\mathsf{crs}_1, \mathsf{xk}_1) \leftarrow \mathcal{NIZK}_1.\mathsf{Setup}(1^\lambda)$, $\mathsf{crs}_2 \leftarrow \mathcal{NIZK}_2.\mathsf{Setup}(1^\lambda)$. It also chooses a key pair $(\mathsf{epk}, \mathsf{esk})$ for the tag-based encryption scheme $\mathcal{TPKE}$. It forwards $\mathsf{pp} := (1^\lambda, \mathsf{crs}_1, \mathsf{crs}_2, \mathsf{epk}, \mathcal{H}, \hat{\mathcal{H}})$ and $\mathsf{tk} := \mathsf{esk}$ to $\mathcal{B}$.
To answer AddA queries, $\mathcal{F}_2$ chooses the authority keys itself. To answer AddU queries, $\mathcal{F}_2$ chooses the user's key pair itself. To answer AddAtt queries, $\mathcal{F}_2$ uses the corresponding authorities' secret keys $\mathsf{ask}_{\mathsf{aid}(\alpha)}$ to create the attribute key for the user.
Adversary $\mathcal{F}_2$ randomly chooses $i \leftarrow \{1, \ldots, \delta(\lambda)\}$ and guesses that $\mathcal{B}$'s forgery will involve forging a one-time signature that verifies under otsvk^* used in answering the i-th signing query.
When asked for the j-th Sign query on $(\mathsf{uid}, \mathcal{A}, m, \mathbb{P})$, if $j \neq i$, $\mathcal{F}_2$ chooses a fresh key pair $(\mathsf{otsvk}, \mathsf{otssk})$ for the one-time signature scheme and answers the query by itself. If $j = i$, $\mathcal{F}_2$ encrypts $\mathbf{uvk}[\mathsf{uid}]$ using $\hat{\mathcal{H}}(\mathsf{otsvk}^*)$ as a tag to obtain C_{tbe} and generates the proof π. It then forwards $(\mathcal{H}(m, \mathbb{P}), \pi, C_{\mathrm{tbe}}, \mathsf{otsvk}^*)$ as the message to its one-time signature signing oracle to get a one-time signature σ_{ots}. $\mathcal{F}_2$ then sends the signature $\Sigma := (\sigma_{\mathrm{ots}}, \pi, C_{\mathrm{tbe}}, \mathsf{otsvk}^*)$ to $\mathcal{B}$. The rest of $\mathcal{B}$'s queries are answered normally as in Fig. 1.
Eventually, when $\mathcal{B}$ outputs its forgery, $\mathcal{F}_2$ aborts if the $\mathcal{B}$'s forgery did not involve forging a one-time signature that verifies w.r.t otsvk^* it got from its game. The probability that $\mathcal{B}$ forges a one-time signature that verifies w.r.t otsvk^* is $\frac{1}{\delta(\lambda)}$.
By the strong existential unforgeability of the one-time signature $\mathcal{OTS}$, $\mathcal{B}$ has a negligible advantage in wining.

This concludes the proof.

Lemma 3. *The construction is traceable if the $\mathcal{NIZK}_1$ proof system is sound and the tagged signature $\mathcal{TS}$ is existentially unforgeable.*

Proof. Since the NIZK proof system $\mathcal{NIZK}_1$ is sound, the adversary has a negligible advantage in succeeding by faking proofs for false statements. We show that if there exists an adversary $\mathcal{B}$ breaking traceability, we can construct an adversary $\mathcal{F}_1$ attacking the unforgeability of the tagged signature scheme $\mathcal{TS}$ such that

$$\mathsf{Adv}^{\mathrm{Trace}}_{\mathcal{DTABS},\mathcal{B}}(\lambda) \leq \kappa(\lambda) \cdot \mathsf{Adv}^{\mathrm{Unforge}}_{\mathcal{TS},\mathcal{F}_1}(\lambda) + \mathsf{Adv}^{\mathrm{Sound}}_{\mathcal{NIZK}_1,\mathcal{F}_2}(\lambda),$$

where $\kappa(\lambda)$ is a polynomial in λ representing an upper bound on the number of honest attribute authorities $\mathcal{B}$ is allowed to use in the game.

- Adversay $\mathcal{F}_1$: Adversary $\mathcal{F}_1$ gets the tagged signature scheme's verification key vk from its game and has access to an oracle Sign that it uses to obtain tagged signatures that verify w.r.t. vk on messages and tags of its choice. Adversary $\mathcal{F}_1$ starts by running $(\mathsf{crs}_1, \mathsf{xk}_1) \leftarrow \mathcal{NIZK}_1.\mathsf{Setup}(1^\lambda)$,

$\mathsf{crs}_2 \leftarrow \mathcal{NIZK}_2.\mathsf{Setup}(1^\lambda)$ and choosing a key pair $(\mathsf{epk}, \mathsf{esk})$ for the tag-based encryption scheme $\mathcal{TPKE}$. It forwards $\mathsf{pp} := (1^\lambda, \mathsf{crs}_1, \mathsf{crs}_2, \mathsf{epk}, \mathcal{H}, \hat{\mathcal{H}})$ and $\mathsf{tk} := \mathsf{esk}$ to $\mathcal{B}$.

Adversary $\mathcal{F}_1$ randomly chooses $i \leftarrow \{1, \ldots, \kappa(\lambda)\}$ and guesses that $\mathcal{B}$'s forgery will involve forging an attribute managed by the attribute authority i. To answer AddA queries, for all authorities $j \neq i$, $\mathcal{F}_1$ chooses the secret/verification keys for the authority itself. For authority i, it sets its verification key to vk it got from its game. If in the game, $\mathcal{B}$ issues $\mathsf{RevealA}$ query on authority i, $\mathcal{F}_1$ aborts the game. To answer AddU queries, $\mathcal{F}_1$ chooses the user's key pair itself. Whenever asked AddAtt queries, if the user has attributes managed by authority i, it forwards such a query to its Sign oracle; Otherwise, it answers the query itself.

To answer Sign queries on $(\mathsf{uid}, \mathcal{A}, m, \mathbb{P})$, $\mathcal{F}_1$ first chooses a fresh key pair $(\mathsf{otsvk}, \mathsf{otssk})$ for the one-time signature $\mathcal{OTS}$ and encrypts $\mathbf{uvk}[\mathsf{uid}]$ using $\hat{\mathcal{H}}(\mathsf{otsvk})$ as a tag and generates the rest of the signature by itself. $\mathcal{F}_1$ forwards the signature Σ to $\mathcal{B}$. The rest of $\mathcal{B}$'s queries are answered normally as in Fig. 1.

Eventually, when $\mathcal{B}$ outputs its forgery, $\mathcal{F}_1$ uses the $\mathcal{NIZK}_1$'s extraction key xk_1 to extract the witness and returns the tagged signature on the $\mathbf{uvk}[\mathsf{uid}^*]$ and the attribute α^* if $\mathcal{B}$'s forgery involved forging a tagged signature that verifies w.r.t. vk it got from its game. Otherwise, it aborts. The probability that $\mathcal{F}_1$ guesses the correct authority is $\frac{1}{\kappa(\lambda)}$.

By the existential unforgeability of the tagged signature, the probability of $\mathcal{B}$ winning is negligible.

This concludes the proof

Lemma 4. *If the $\mathcal{NIZK}_1$ and $\mathcal{NIZK}_2$ proof systems are zero-knowledge, the tag-based encryption scheme $\mathcal{TPKE}$ is selective-tag weakly IND-CCA secure, the one-time signature $\mathcal{OTS}$ is strongly existentially unforgeable, and the hash functions $\hat{\mathcal{H}}$ and $\mathcal{H}$ are collision-resistant then the construction is fully anonymous (against full-key exposure).*

Proof. We show that if there exists an adversary $\mathcal{B}$ breaking anonymity, we can construct adversaries: $\mathcal{F}_1$ against the collision-resistance of the hash function $\hat{\mathcal{H}}$, $\mathcal{F}_2$ against the strong unforgeability of the one-time signature $\mathcal{OTS}$, $\mathcal{F}_3$ against the collision-resistance of the hash function $\mathcal{H}$, $\mathcal{F}_4$ against the NIZK property of the proof system $\mathcal{NIZK}_1$, $\mathcal{F}_5$ against the NIZK property of the proof system $\mathcal{NIZK}_2$, and $\mathcal{F}_6$ against the selective-tag weakly IND-CCA security of the tag-based encryption scheme $\mathcal{TPKE}$.

By the collision-resistance of $\hat{\mathcal{H}}$, $\mathcal{B}$ has a negligible probability in finding otsvk' s.t. $\hat{\mathcal{H}}(\mathsf{otsvk}')$ collides with the tag $\hat{\mathcal{H}}(\mathsf{otsvk}^*)$ we use for the challenge signature. If this is not the case, we can use $\mathcal{B}$ to construct an adversary $\mathcal{F}_1$ that breaks the collision-resistance of $\hat{\mathcal{H}}$.

The strong existential unforgeability of $\mathcal{OTS}$ ensures that $\mathcal{B}$ has a negligible probability in forging a one-time signature under otsvk^* we use in the challenge signature. If this is not the case, we can construct an adversary $\mathcal{F}_2$ that wins the strong unforgeability game of $\mathcal{OTS}$.

By the collision-resistance of $\mathcal{H}$, $\mathcal{B}$ has a negligible probability in finding pairs $(m^*, \mathbb{P}^*) \neq (m, \mathbb{P})$ s.t. $\mathcal{H}(m^*, \mathbb{P}^*) = \mathcal{H}(m, \mathbb{P})$. If this is not the case, we can use $\mathcal{B}$ to construct an adversary $\mathcal{F}_3$ which breaks the collision-resistance of the hash function $\mathcal{H}$. Thus, from now on we assume that there are no hash collisions.

We instantiate $\mathcal{NIZK}_1$ in the simulation setting which is, by the security of $\mathcal{NIZK}_1$, is indistinguishable from the soundness setting. The proof π is thus now zero-knowledge and hence does not reveal any information about the witness. Also, we instantiate $\mathcal{NIZK}_2$ in the simulation setting which is indistinguishable from the soundness setting. The proof π_{Trace} is now also zero-knowledge and hence $\mathcal{B}$ cannot tell simulated proofs from real proofs.

We now proceed to construct an adversary $\mathcal{F}_6$ against the selective-tag weakly IND-CCA security of $\mathcal{TPKE}$ using adversary $\mathcal{B}$. Adversary $\mathcal{F}_6$ runs the Setup algorithm where it starts by randomly choosing a key pair $(\mathsf{otsvk}^*, \mathsf{otssk}^*)$ for $\mathcal{OTS}$ that it will use when producing the challenge signature. We needed to choose the key pair beforehand as the tag-based encryption scheme is only selective-tag secure and hence the challenger in the ST-WIND-CCA game needs to know the challenge tag before sending the public-key epk for $\mathcal{TPKE}$. $\mathcal{F}_6$ sends $\hat{\mathcal{H}}(\mathsf{otsvk}^*)$ to its challenger and gets back epk. In its game, $\mathcal{F}_6$ has access to a decryption oracle Dec which it can query on any ciphertext under any tag different from $\hat{\mathcal{H}}(\mathsf{otsvk}^*)$. $\mathcal{F}_6$ chooses crs_1 and crs_2 as simulation reference strings and forwards $\mathsf{pp} := (1^\lambda, \mathsf{crs}_1, \mathsf{crs}_2, \mathsf{epk}, \hat{\mathcal{H}}, \mathcal{H})$ to $\mathcal{B}$.

To answer AddU queries, $\mathcal{F}_6$ chooses the secret/verification keys of the user itself. To answer AddA queries, $\mathcal{F}_6$ chooses the secret/verification keys for the authorities itself. Thus, $\mathcal{F}_6$ can answer any AddAtt queries itself.

To answer the challenge query $\mathsf{CH}_b((\mathsf{uid}_0, \mathcal{A}_0), (\mathsf{uid}_1, \mathcal{A}_1), m, \mathbb{P})$, $\mathcal{F}_6$ sends $(\mathbf{uvk}[\mathsf{uid}_0], \mathbf{uvk}[\mathsf{uid}_1])$ as its challenge in its ST-WIND-CCA game and gets a ciphertext under the tag $\hat{\mathcal{H}}(\mathsf{otsvk}^*)$ of either the plaintext $\mathbf{uvk}[\mathsf{uid}_0]$ or $\mathbf{uvk}[\mathsf{uid}_1]$ which it needs to distinguish. $\mathcal{F}_6$ can now construct the rest of the challenge signature by simulating the proof π and signing the whole thing with otssk^* to obtain σ_{ots}.

To answer Trace queries, $\mathcal{F}_6$ just uses its decryption oracle to get the decryption of C_{tbe} which is part of the signature and then simulates proof π_{Trace}. Note that since we have chosen the challenge tag otsvk^* uniformly at random and since we already eliminated any case where any signature sent to Trace uses the same tag as that we used for the challenge signature, such a query will be accepted by $\mathcal{F}_6$'s decryption oracle because the tag is different from the tag used in the challenge ciphertext. The rest of $\mathcal{B}$'s queries are answered as in Fig. 1.

Eventually, when $\mathcal{B}$ halts, $\mathcal{F}_6$ outputs whatever $\mathcal{B}$ outputs. By the ST-WIND-CCA property of the tag-based encryption scheme, $\mathcal{B}$ has a negligible probability in winning.

This concludes the proof.

Lemma 5. *The construction satisfies tracing soundness if the NIZK systems $\mathcal{NIZK}_1$ and $\mathcal{NIZK}_2$ are sound.*

Proof. The soundness of $\mathcal{NIZK}_2$ ensures correct decryption of the tag-based ciphertext C_{tbe} and hence the adversary has a negligible advantage in producing a proof for a false statement that C_{tbe} decrypts to a different plaintext.

Similarly, the soundness of $\mathcal{NIZK}_1$ guarantees that the ciphertext C_{tbe} is constructed correctly. By the correctness of the tag-based encryption scheme, we have that a correctly encrypted ciphertext will always decrypt to the same plaintext that was encrypted.

This concludes the proof.