Hierarchical Attribute-based Signatures

Constantin-Cătălin Drăgan, Daniel Gardham and Mark Manulis

Surrey Centre for Cyber Security, University of Surrey, UK

`c.dragan@surrey.ac.uk, d.gardham@surrey.ac.uk, mark@manulis.eu`

Abstract. Attribute-based Signatures (ABS) are a powerful tool allowing users with attributes issued by authorities to sign messages while also proving that their attributes satisfy some policy. ABS schemes provide a flexible and privacy-preserving approach to authentication since the signer's identity and attributes remain hidden within the anonymity set of users sharing policy-conform attributes. Current ABS schemes exhibit some limitations when it comes to the management and issue of attributes. In this paper we address the lack of support for hierarchical attribute management, a property that is prevalent in traditional PKIs where certification authorities are organised into hierarchies and signatures are verified along roots of trust.

Hierarchical Attribute-based Signatures (HABS) introduced in this work support delegation of attributes along paths from the top-level authority down to the users while also ensuring that signatures produced by these users do not leak their delegation paths, thus extending the original privacy guarantees of ABS schemes. Our generic HABS construction also ensures unforgeability of signatures in the presence of collusion attacks and contains an extended traceability property allowing a dedicated tracing authority to identify the signer and reveal its attribute delegation paths. We include a public verification procedure for the accountability of the tracing authority.

We anticipate that HABS will be useful for privacy-preserving authentication in applications requiring hierarchical delegation of attribute-issuing rights and where knowledge of delegation paths might leak information about signers and their attributes, e.g., in intelligent transport systems where vehicles may require certain attributes to authenticate themselves to the infrastructure but remain untrackable by the latter.

1 Introduction

Attribute-based Signatures. ABS schemes, introduced independently in [30] and [29], offer a flexible, privacy preserving primitive for authenticating messages. When presented with an attribute-based signature, verifiers are convinced that the signer owns a set of attributes satisfying the signing policy, however, they do not learn the signer's identity nor the set of attributes and hence provide for signer's anonymity within a set of users holding policy-conform attributes. Users who are not in possession of such attributes are not able to produce valid

ABS signatures, even if they collude and try to pool their attributes together. We note that while [30] enjoys an expressive policy, [29] drops this fine-grained control in favour of a more efficient threshold-based construction and such trade-off has commonly been seen in some later ABS schemes, e.g. [29, 20, 25]. ABS schemes can also be generalised to policy-based signatures [4].

Existing ABS constructions typically rely on zero-knowledge proofs in the signature generation phase, with a vast majority of schemes, e.g. [18, 33, 32, 4, 17, 21, 16], being proposed in the standard model based on bilinear maps and Groth-Sahai proofs [23]. A notable exception is the scheme in [25], which uses the RSA setting yet requires random oracles.

While anonymity makes ABS schemes interesting for fine-grained privacy-preserving authentication, the ability to trace and identify signers for accountability is another useful property that has been considered in the context of several ABS schemes [17, 15, 21].

Hierarchy and Delegation. Requiring that all signers obtain their attributes from a single authority introduces scalability issues if the universe of signers or attributes becomes large. Therefore, some ABS schemes, e.g. [30, 32, 17, 21], explicitly consider the case where multiple, possibly independent authorities can issue attributes directly to the signers. On the other hand, if the ability to issue attributes requires authorisation then some additional control mechanisms for *delegation* of attribute-issuing rights would be needed.

From a more general view, it would require an ABS scheme to support a hierarchy of attribute authorities, managed by some top-level (root) authority, and enable delegation of issuing rights (for subsets of attributes) along various delegation paths consisting of intermediate authorities. With such hierarchy, signers would be able to obtain attributes only from authorities that are authorised to issue them. For practical purposes the hierarchy should be dynamically expandable, i.e., it should be possible to add intermediate authorities (at any level) at any time. This setting resembles conceptual similarity with traditional PKIs where certification authorities form a hierarchy. The main difference is that, in the context of ABS schemes, such hierarchical delegation imposes additional challenges on the privacy of signers since leakage of the delegation path from the ABS signature may compromise the signer's anonymity by leaking information about the subsets of attributes that the signer might possess, e.g. if some authority is only responsible for a small number of attributes or otherwise has some identifying information (for example, geographic location) that could also be used to identify the signer. We note that a hybrid solution whereby the hierarchy is defined by the CAs, delegation is performed via traditional PKI certification and credentials issued to signers are standard ABS credentials would suffer from the same privacy-leaking problems. Therefore, hierarchical ABS schemes must incorporate a proof of validity of the delegation paths without disclosing any information about intermediate authorities, implying that verification must be done in relation to the public key of the top-level authority and not of the intermediate authorities.

This brings further challenges when it comes to accountability. The associated tracing mechanism needs to identify not only the signer (as in the case of existing ABS constructions) but also delegation paths through which attributes were obtained. This is because the root authority may not be aware that some particular user was issued attributes by an intermediate authority. In addition, malicious intermediate authorities in the hierarchy can try to misuse their delegation rights and create further fake authorities or users to issue rogue ABS signatures. The delegation paths disclosed by the tracing algorithm would be able to detect this behaviour, thus extending accountability to intermediate authorities.

Our Contribution: Hierarchical ABS. In this paper we solve the aforementioned challenges by proposing *Hierarchical ABS (HABS)* to enable hierarchical management by a root authority and delegation of attribute-issuing rights. HABS extends the anonymity guarantees by hiding not only the signers and their attributes but also the delegation paths (i.e., intermediate authorities) that were used for these attributes. We call this new property *path anonymity.* HABS supports dynamic hierarchies, enables delegation of attribute-issuing rights to new authorities, and allows signers to obtain attributes from multiple authorities within the hierarchy with guarantees that the entire delegation path (incl. the signer) can be revealed by an independent tracing authority. Moreover, we require public verifiability for the output of this tracing procedure to address the case where the tracing authority tries to cheat. Needless to say, HABS offers extended unforgeability guarantees, expressed through the *non-frameability* requirement, ensuring that only holders of policy-conform attributes can sign, and in particular, preventing collusions between signers and authorities. We formally define HABS and propose its generic construction from public key encryption, digital signatures and non-interactive zero-knowledge proofs of knowledge. Our HABS scheme can be instantiated using bilinear maps in the standard model under the DLIN, q-SDH and Simultaneous Flexible Pairing (SFP) [1] assumptions. Our HABS scheme supports a more general scenario where intermediate authorities and users can become part of multiple independent hierarchies, each managed by a separate root authority. We also discuss the revocation of delegation and signing rights in the context of HABS, which is known to be notoriously challenging for all hierarchical signature schemes (incl. PKIs).

Applications. HABS can find applications in traditional ABS scenarios (cf. [30]) while enabling hierarchical delegation of attributes. Due to their distinctive path-anonymity and path-traceability properties we anticipate further applications in privacy-preserving authentication where there is a need to also hide the intermediate authorities that were involved in issuing the attributes.

For example, in intelligent transport systems [37] there is a challenge to authenticate messages sent by vehicles to other parties (e.g. other vehicles, infrastructure, police, etc) while preventing that vehicles can be tracked. Existing approaches, based either on pseudonymous PKI certificates [26, 22], group signatures [36, 24] or identity-based signatures [27, 38] have all their limitations with regard to scalability and/or limited expressivity (cf. survey in [34]). As noted in [31] an attribute-based approach would bring substantial benefits and while [31] aims at realisations with heavier attribute-based credential systems, we believe that HABS could offer a more lightweight alternative. For example, the root authority can be some regulatory authority while manufacturers, authorized dealerships, local garages or testing facilities would define the hierarchy. Assume some town has a policy that bans diesel vehicles that did not pass an emission test. By viewing a vehicle's fuel type and its emission test results as attributes issued by the manufacturer and some local testing facility, respectively, the vehicle would be able to prove its compliance with the town's policy without disclosing any other information such as its make or which local facility performed the test.

Further related work. As explained in [30], group signatures [14] and ring signatures [35] can be viewed as special cases of ABS satisfying policies that would contain only disjunctions over the attributes (identities). The proposed constructions of hierarchical group [39] and ring [28] signatures, seen as special cases of HABS, also lack this richer expressivity of the attribute-based setting. Mesh signatures [7], a generalization of ring signatures to monotone access structures that can be satisfied by combinations of atomic signatures would not provide unforgeability against colluding signers and would leak verification keys (attributes) for all atomic signatures used in the clauses. As discussed in [30], anonymous credentials (AC) [13, 12], used for privacy-preserving attribute-based credentials [11], are a more powerful, yet also less efficient, primitive than ABS. AC schemes require costly zero-knowledge proofs during attribute acquisition since their goal is to prevent that authorities can link users to whom they issue attributes. This property is not provided by (H)ABS and may not even be needed for its applications (as in our example above where it does not make sense to hide the manufacturer of a vehicle from the local testing authority that carries out the emission test). Nonetheless, we note that the concept of delegation has also been explored for AC schemes [3] where some delegation mechanisms also require zero-knowledge proofs to provide similar guarantees as in the issue of anonymous credentials and in addition to prove that the delegator is L levels away from the (top-level) authority, a property that is not needed for HABS where intermediate authorities know their delegation paths. Anonymous credentials in this setting have also been proposed [9]. Whilst the construction is more efficient, it only supports attribute issuance along one delegation path and in particular, all attributes owned by an authority in the path are required for verification. We also note anonymous proxy signatures [19] that bear similarities with HABS in that the delegation path for the proxy signer is not revealed upon verification of proxy signatures but can be traced through a dedicated authority. These signatures are not attribute-based since tasks delegated to the proxy signer, when viewed as attributes, are not hidden. Functional signatures [8, 2] are another primitive that allow for controlled delegation of signing rights. A signing key is created w.r.t a function f, and one can only sign a message if it is in the range of f, and in the case of delegatable schemes [2] allow signatures to be modified by another specified party. When the message m is viewed as a set of attributes satisfying some policy f, signers require separate signings keys for each possible policy in the system, which is impractical. Another related primitive are homomorphic signatures, which have been claimed to be equivalent to ABS [40]. This implication has only been shown in the single user setting and thus does not capture the strong unforgeability requirements provided by HABS. Similarly, in relation to policy-based signatures [4] (which imply ABS), we observe that, so far, the only delegation mechanism proposed for these schemes in [4] neither supports separation between users and authorities nor distinguishes between the signers. This allows authorities to forge signatures on behalf of users (cf. Remark 1) and excludes the possibility of tracing signatures.

2 Model of Hierarchical Attribute-based Signatures

In this section we describe the entities, their roles and define the algorithms for HABS. The involved entities are attribute authorities, users (signers), and a dedicated tracing authority.

Attribute Authorities. The *root authority (RA)* with its key pair (ask_0, apk_0) is at the top of HABS hierarchy and defines its universe of attributes $\mathbb{A}$. The RA can delegate subsets from $\mathbb{A}$ to other authorities in the scheme, who are then able to delegate attributes from their subsets further, creating a dynamically expandable hierarchy of attributes (see Fig. 1). In order to be admitted to the hierarchy, each *intermediate authority (IA)* needs to generate its own key pair (ask_i, apk_i), $i > 0$ and become authorised by some already admitted authority by obtaining a subset of attributes. In addition to delegation of attributes, each authority can issue attributes from its set to the end users (signers). It is assumed that admitting authorities make sufficient checks on whether the entities they admit are eligible to receive the attributes.

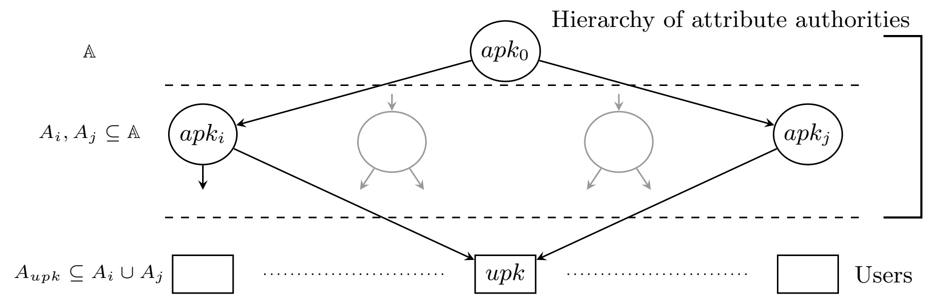
Hierarchy of attribute authorities
$\mathbb{A}$
apk_0
$A_i, A_j \subseteq \mathbb{A}$
apk_i
apk_j
$A_{upk} \subseteq A_i \cup A_j$
upk
Users

Fig. 1. Example of a HABS hierarchy where a user with public key upk receives its set of attributes $A_{upk} \subseteq A_i \cup A_j$ from two IAs i and j, who in turn receive their attribute sets A_i and A_j from the RA.

Users. Upon joining, each user generates its own key pair (usk, upk) and is issued with a subset of attributes by one or more authorities from the hierarchy. Any admitted user can generate a valid HABS signature on a message m with respect to some *predicate* Ψ using the secret key usk and the set of issued attributes A as long as this set contains some subset $A' \subseteq A$ that are needed to satisfy Ψ, i.e. $\Psi(A') = 1$. HABS signatures will be verified with respect to Ψ and the RA's public key apk_0. Note that, unlike authorities, users cannot delegate their attributes to other entities and can be viewed as the lowest level of the hierarchy (see Fig. 1). We make use of a label $\star$ to denote the end of the delegation path, and prevent the user from further delegating his attributes. Note we sometimes use j to differentiate users.

Warrants. For delegation of attributes to intermediate authorities and for their issue to the signers it is convenient to use warrants. That is, upon admission each HABS entity (IAs and signer) obtains a *warrant* **warr** that contains *all* attributes $a \in \mathbb{A}$ (along with their delegation paths) that the entity receives from the authority. However, upon signing we assume that a "reduced warrant" **warr** containing only a reduced attribute set A' for which $\Psi(A') = 1$ will be used by the signer. Note that by $\mathbf{warr}[a]$, for $a \in \mathbf{warr}$, we denote the *delegation path* $(apk_0, \dots, \{apk_i, \{upk, \star\}\})$, starting with RA's public key apk_0 and ending with the entity's public key, i.e. apk_i for the IA i or upk followed by a fixed label $\star$ (which is used to denote that this path cannot be extended further) for the corresponding user. We use $|\mathbf{warr}|$ to denote the total number of attributes in **warr** and use $|\mathbf{warr}[a]|$ to refer to the length of the delegation path for the attribute a.

Tracing Authority. A dedicated *tracing authority (TA)* with its own key pair (tsk, tpk) is responsible for tracing HABS signatures. The extended tracing procedure in HABS outputs **warr** used by the signer. This means that tracing reveals all attributes and their delegation paths from used **warr** and also the identity of the signer since its delegation paths include upk. Note that since users can use reduced warrants, the tracing procedure does not necessarily reveal all attributes held by the user but only those that were used to produce the signature. For accountability purposes we require that the output of TA is publicly verifiable, i.e. is accompanied by some proof that can be verified using a public judgment procedure.

Definition 1 (Hierarchical ABS Scheme). *A* $\mathtt{HABS} := (\mathtt{Setup}, \mathtt{KGen}, \mathtt{AttIssue}, \mathtt{Sign}, \mathtt{Verify}, \mathtt{Trace}, \mathtt{Judge})$ *scheme consists of the following seven processes:*

- $\mathtt{Setup}(1^\lambda)$ *is the initialisation process where based on some security parameter* $\lambda \in \mathbb{N}$*, the public parameters* pp *of the scheme are defined, and the root and tracing authority independently generate their own key pair, i.e. RA's* (ask_0, apk_0) *and TA's* (tsk, tpk)*. In addition, RA defines the universe* $\mathbb{A}$ *of attributes, and a label* $\star$ *for users. We stress that due to dynamic hierarchy, the system can be initialised by publishing (*pp*,* $apk_0, tpk)$ *with* $\mathbb{A}$ *and* $\star$ *contained in* pp*.*
- $\mathtt{KGen}(\mathrm{pp})$ *is a key generation algorithm executed independently by intermediate authorities and users. Each entity generates its own key pair, i.e.,* (ask_i, apk_i) *for* $i > 0$ *or* (usk, upk)*.*
- $\mathtt{AttIssue}\ (ask_i, \boldsymbol{warr}_i, A, \{apk_j|upk_j\})$ *is an algorithm that is used to delegate attributes to an authority with* apk_j *or issue them to the user with* upk_j*.*

On input of an authority's secret key ask_i, $i \in \mathbb{N}_0$, its warrant $\boldsymbol{warr}_i$, a subset of attributes A from $\boldsymbol{warr}_i$, and the public key of the entity to which attributes are delegated or issued, it outputs a new warrant $\boldsymbol{warr}$ for that entity.

- `Sign` $((usk, \boldsymbol{warr}), \mathrm{m}, \Psi)$ *is the signing algorithm. On input of the signer's* usk *and (possibly reduced)* $\boldsymbol{warr}$, *a message* m *and a predicate* Ψ *it outputs a signature* σ.
- `Verify` $(apk_0, (\mathrm{m}, \Psi, \sigma))$ *is a deterministic algorithm that outputs 1 if a candidate signature* σ *on a message* m *is valid with respect to the predicate* Ψ *and 0 otherwise.*
- `Trace` $(tsk, apk_0, (\mathrm{m}, \Psi, \sigma))$ *is an algorithm executed by the TA on input of its private key* tsk *and outputs either a triple* $(upk, \boldsymbol{warr}, \hat{\pi})$ *if the tracing is successful or* $\perp$ *to indicate its failure. Note that* $\boldsymbol{warr}$ *contains attributes and delegation paths that were used by the signer.*
- `Judge` $(tpk, apk_0, (\mathrm{m}, \Psi, \sigma),\ (upk, \boldsymbol{warr}, \hat{\pi}))$ *is a deterministic algorithm that checks a candidate triple* $(upk, \boldsymbol{warr}, \hat{\pi})$ *from the tracing algorithm and outputs* 1 *if the triple is valid and* 0 *otherwise.*

The *correctness* property of HABS requires that any signature σ output by any signer with usk in possession of a legitimately issued warrant $\mathbf{warr}$ that contains attributes $a \in \mathbb{A}$ satisfying Ψ can be successfully verified and traced, and that a triple $(upk, \mathbf{warr}, \hat{\pi})$ output by the tracing algorithm for such σ passes the public judgment procedure.

2.1 Security Properties

In this section, we define three security properties of HABS schemes: *path anonymity*, *non-frameability*, and *path traceability* and use game-based definitions assuming some PPT adversary $\mathcal{A}$ that interacts with the entities using oracles. We note that our definitions extend earlier definitions for (multi-authority) ABS schemes, e.g. [30, 4, 17, 21], to account for the hierarchical setting and potential corruptions within the hierarchy. In addition, our definitions of *path-anonymity* and *path-traceability* focus on hiding resp. verifiable traceability of delegation paths from the signer's warrant, a distinctive feature of HABS. Our modeling techniques for these properties are inspired by definitions behind anonymous proxy signatures [19] which do not apply directly to the attribute-based setting.

Oracles for $\mathcal{A}$. The oracles available to a PPT adversary $\mathcal{A}$ are defined in Fig. 2 and their high-level description is provided in the following. In our oracles we take into account that only authorities can delegate and issue attributes whereas only signers can generate HABS signatures.

- O_{Reg}: $\mathcal{A}$ can register new IAs and users for whom, in response, a key pair will be honestly generated and the public key given to $\mathcal{A}$. The oracle uses lists to keep track of the established entities and their keys. Upon registration all entities are initially considered to be honest.
- O_{Corr}: $\mathcal{A}$ can corrupt established entities. On input of the entity's public key $\mathcal{A}$ receives the corresponding private key, as long as this entity has been previously established. The oracle keeps track of entities who were corrupted.

```
$O_{\text{Reg}}(\,\cdot\,)$ with $(\,\cdot\,) = (\,i\,)$ and $i \notin HU$
1 : $(sk_i, pk_i) \leftarrow \texttt{KGen}(\text{pp})$
2 : $List \leftarrow List \cup \{(i, pk_i, sk_i)\}$
3 : $HU \leftarrow HU \cup \{i\}$
4 : return $pk_i$

$O_{\text{Corr}}(\,\cdot\,)$ with $(\,\cdot\,) = (\,i\,)$
1 : if $i \in HU$ then
2 :     $HU \leftarrow HU - \{i\}$
3 : return $sk_i$ from $List$

$O_{\text{Tr}}(\,\cdot\,)$ with $(\,\cdot\,) = (\text{m}, \Psi, \sigma)$
1 : return $\texttt{Trace}(tsk, apk_0, (\text{m}, \Psi, \sigma))$

$O_{\text{Att}}(\,\cdot\,)$
$(\,\cdot\,) = (i, \mathbf{warr}_i, a, \{apk_j | upk_j\})$
1 : $L := \{a | (a, pk_a, sk_a) \in List\}$
2 : if $i \in L \;\wedge j \in L$ then
3 :     $\mathbf{warr} \leftarrow \texttt{AttIssue}(ask_i,$
            $\mathbf{warr}_i, a, \{apk_j | upk_j\})$
4 :     return $\mathbf{warr}$
5 : return $\perp$

$O_{\text{Sig}}(\,\cdot\,)$ with $(\,\cdot\,) = (i, \mathbf{warr}, \text{m}, \Psi)$
1 : $A \leftarrow \{a|\; a \in \mathbf{warr}\}$
2 : if $i \in HU \;\wedge\; \Psi(A)$ then
3 :     $\sigma \leftarrow \texttt{Sign}((usk_i, \mathbf{warr}), \text{m}, \Psi)$
4 :     return $\sigma$
5 : return $\perp$
```

Fig. 2. Oracles used in the HABS security experiments.

- O_{Att}: $\mathcal{A}$ can ask an authority to either delegate attributes for another IA or to issue attributes to an user, as long as both involved entities are registered. Note that $\mathcal{A}$ can define which attributes the oracle should use. If both entities are registered and the issuing entity has rights to issue attributes provided by $\mathcal{A}$ then the output warrant **warr** is given to $\mathcal{A}$.
- O_{Sig}: $\mathcal{A}$ can ask a signer to produce a HABS signature using the input warrant **warr**, a message m and a predicate Ψ. If the provided warrant contains a set of attributes A satisfying Ψ and the signer is not corrupted then the signature will be given to $\mathcal{A}$.
- O_{Tr}: $\mathcal{A}$ can ask the TA to perform the tracing procedure on its input, in which case its output (which can also be $\perp$) is returned to $\mathcal{A}$.

Path Anonymity. For HABS we extend the anonymity property of traditional ABS schemes to achieve privacy of the delegation path, i.e., not only to hide the signer but also all intermediate authorities that were involved in the delegation of attributes for that signer. Our game for *path anonymity* in Fig. 3 requires the adversary to decide which user's warrant and private key were used in the generation of the challenge HABS signature σ_b. We consider a powerful two-stage PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, who knows the private keys of the candidate signers and can moreover establish its own HABS hierarchy (with IAs and users) by learning the secret key ask_0 of the root authority. This also means that the adversary comes up with the candidate warrants $\mathbf{warr}_0$ and $\mathbf{warr}_1$ for the two users in the challenge phase. Since HABS signatures do not aim to hide the length of the delegation paths nor the number of attributes used to satisfy the policy we require that both warrants are of the same size and that they both

```
$\mathbf{Exp}^{\text{pa-}b}_{\mathtt{HABS},\mathcal{A}}(\lambda)$
1 : $(\text{pp}, ask_0, tsk) \leftarrow \mathtt{Setup}(1^\lambda)$
2 : $(\mathsf{st}, (usk_0, \mathbf{warr}_0), (usk_1, \mathbf{warr}_1), m, \Psi) \leftarrow \mathcal{A}_1(\text{pp}, ask_0 : O_{\text{Reg}}, O_{\text{Corr}}, O_{\text{Tr}})$
3 : if $|\mathbf{warr}_0| = |\mathbf{warr}_1|$ then
4 :   $\sigma_0 \leftarrow \mathtt{Sign}((usk_0, \mathbf{warr}_0), \text{m}, \Psi),\ \sigma_1 \leftarrow \mathtt{Sign}((usk_1, \mathbf{warr}_1), \text{m}, \Psi)$
5 :   if $\mathtt{Verify}(apk_0, (m, \Psi, \sigma_0)) = 1$ and $\mathtt{Verify}(apk_0, (m, \Psi, \sigma_1)) = 1$ then
6 :     $b' \leftarrow \mathcal{A}_2(\mathsf{st}, \sigma_b : O_{\text{Tr}})$
7 :     return $b' \ \wedge\ \mathcal{A}_2$ did not query $O_{\text{Tr}}(tsk, (m, \Psi, \sigma_b))$
8 : return 0
```

Fig. 3. Path-Anonymity Experiment

satisfy the predicate Ψ output by the adversary. Since attributes are contained in warrants our definition also implies attribute-hiding.

Definition 2 (Path Anonymity). *A* `HABS` *scheme offers path anonymity if no PPT adversary $\mathcal{A}$ can distinguish between $\mathbf{Exp}^{\text{pa-0}}_{\mathtt{HABS},\mathcal{A}}$ and $\mathbf{Exp}^{\text{pa-1}}_{\mathtt{HABS},\mathcal{A}}$ defined in Fig. 3, i.e., the following advantage is negligible in λ:*

$$\mathsf{Adv}^{\text{pa}}_{\mathtt{HABS},\mathcal{A}}(\lambda) = \left|\Pr\left[\mathbf{Exp}^{\text{pa-0}}_{\mathtt{HABS},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{pa-1}}_{\mathtt{HABS},\mathcal{A}}(\lambda) = 1\right]\right|.$$

Non-Frameability. Another fundamental property for HABS is *non-frameability* that extends unforgeability to ensure only authorized authorities can delegate and only attribute policy-compliant users can sign. This property is formalized in Fig. 4, and requires the adversary $\mathcal{A}$ to produce valid authorizations for attributes he does not satisfy: either as a valid HABS signature σ for some honest user with upk, or as a valid tracing information that includes a warrant $\mathbf{warr}$ issued by an honest authority. In the latter, it is enough for $\mathcal{A}$ to provide a single attribute a for which the delegation path contains one honest authority i or an honest user with upk without querying O_{Att} for that authority or user. We consider a PPT adversary $\mathcal{A}$, who can admit IAs to the HABS hierarchy using the RA's private key ask_0, and act on behalf of the TA using tsk.

Definition 3 (Non-Frameability). *A* `HABS` *scheme is non-frameable, if no PPT adversary $\mathcal{A}$ can win the experiment $\mathbf{Exp}^{\text{nf}}_{\mathtt{HABS},\mathcal{A}}$ defined in Fig. 4, i.e., the following advantage is negligible in λ:*

$$\mathsf{Adv}^{\text{nf}}_{\mathtt{HABS},\mathcal{A}}(\lambda) = \Pr\left[\mathbf{Exp}^{\text{nf}}_{\mathtt{HABS},\mathcal{A}}(\lambda) = 1\right].$$

Remark 1 In the non-frameability experiment we consider a strong adversary that has full control of the hierarchy through the O_{Reg} and O_{Att} oracles. We capture the notion that malicious authorities and colluding users should not be able to produce signatures on behalf of, and therefore framing, honest users. This is a stronger notion of security than considered in some existing ABS schemes [30, 4].

```
$\mathbf{Exp}^{\text{nf}}_{\texttt{HABS},\mathcal{A}}(\lambda)$
1 : $(\text{pp}, ask_0, tsk) \leftarrow \texttt{Setup}(1^\lambda)$
2 : $((\sigma, m, \Psi), (upk_j, \mathbf{warr}, \hat{\pi})) \leftarrow \mathcal{A}(\text{pp}, ask_0, tsk : O_{\text{Att}}, O_{\text{Sig}}, O_{\text{Corr}}, O_{\text{Reg}})$
3 : if $\texttt{Verify}(apk_0, (\text{m}, \Psi, \sigma)) \wedge \texttt{Judge}(tpk, apk_0, (\text{m}, \Psi, \sigma), (upk_j, \mathbf{warr}, \hat{\pi}))$ then
4 :   if $j \in HU \wedge \mathcal{A}$ did not query $O_{\text{Sig}}((usk_j, \mathbf{warr}), \text{m}, \Psi)$ then,   return 1
5 :   if $\exists a.\ a \in \mathbf{warr} \implies (apk_0, apk_1, \ldots, apk_n, upk_j, \star) = \mathbf{warr}[a] \wedge$
6 :         $((\exists i \in [0, n-1].\ \mathcal{A}$ did not query $O_{\text{Att}}(i, \cdot, a, apk_{i+1}) \wedge i \in HU) \vee$
7 :          $(\mathcal{A}$ did not query $O_{\text{Att}}(n, \cdot, a, upk_j) \wedge n \in HU)$ ) then, return 1
8 : return 0
```

Fig. 4. Non-Frameability Experiment

Path Traceability. The final property we consider for HABS is *path traceability* in Fig. 5 that offers accountability for the entire delegation path and the tracing authority, but also validity of the entities in that delegation path. The adversary $\mathcal{A}$ is required to produce a valid HABS signature σ that either cannot be traced, or can be traced to a warrant **warr** that contains at least one "rogue" entity (some authority i or user with upk) within any of its delegation paths that has not been previously registered through the registration oracle, i.e., is not contained in *List*. For honest and registered authorities $\mathcal{A}$ can use the attribute-issuing oracle, which internally checks whether the public key of the entity for which the warrant needs to be issued has been registered before. This excludes a trivial attack where $\mathcal{A}$ obtains a legitimate warrant for some rogue entity from some honest authority. In its attack we also equip $\mathcal{A}$ with the TA's private key.

```
$\mathbf{Exp}^{\text{tr}}_{\texttt{HABS},\mathcal{A}}(\lambda)$
1 : $(\text{pp}, ask_0, tsk) \leftarrow \texttt{Setup}(1^\lambda)$
2 : $((\sigma, m, \Psi), (upk, \mathbf{warr}, \hat{\pi})) \leftarrow \mathcal{A}(\text{pp}, tsk : O_{\text{Att}}, O_{\text{Corr}}, O_{\text{Reg}})$
3 : if $\texttt{Verify}(apk_0, (m, \Psi, \sigma))$ then
4 :   if $\texttt{Trace}(tsk, (m, \Psi, \sigma)) = \perp$ then,   return 1
5 :   if $\texttt{Judge}(tpk, apk_0, (m, \Psi, \sigma), (upk, \mathbf{warr}, \hat{\pi})) \wedge$
6 :     $(\exists a.\ a \in \mathbf{warr} \implies (apk_0, apk_1, \ldots, apk_n, upk, \star) = \mathbf{warr}[a] \wedge$
7 :         $( (\exists i \in [0, n-1].\ i \in HU \wedge (i+1, apk_{i+1}, ask_{i+1}) \notin List) \vee$
8 :           $(n \in HU \wedge (\cdot, upk, usk) \notin List)$ ) ) then,   return 1
9 : return 0
```

Fig. 5. Path-Traceability Experiment.

Definition 4 (Path Traceability). *A* HABS *scheme offers path traceability if no PPT adversary* $\mathcal{A}$ *can win the experiment* $\mathbf{Exp}^{\text{tr}}_{\texttt{HABS},\mathcal{A}}$ *defined in Fig. 5, i.e.,* *the following advantage is negligible in λ:*

$$\mathsf{Adv}^{\mathrm{tr}}_{\mathtt{HABS},\mathcal{A}}(\lambda) = \Pr\left[\mathbf{Exp}^{\mathrm{tr}}_{\mathtt{HABS},\mathcal{A}}(\lambda) = 1\right].$$

3 Construction

In this section we describe and analyse our general construction for HABS that we build from several well-known building blocks.

3.1 Building Blocks

Our construction relies on standard notions of IND-CCA2 secure *public key encryption* $\mathtt{PKE} := (\mathtt{KGen}, \mathtt{Enc}, \mathtt{Dec})$ [10] and an unforgeable *digital signature* $\mathtt{DS}{:=}(\mathtt{KGen}, \mathtt{Sign}, \mathtt{Verify})$ [1] that withstands chosen-message attacks. We rely further on an unforgeable *tagged signature* $\mathtt{TS} := (\mathtt{KGen}, \mathtt{Sign}, \mathtt{Verify})$ [1] that can sign blocks of messages, also used in [17], where an additional tag t is used as input to the signing algorithm and the signature will not verify unless the verifier uses the same tag. The adversary is allowed to query its signing oracle on tags that it can use later to create a forgery. Although any unforgeable $\mathtt{DS}$ scheme can be used as a tagged signature if its message space admits signing pairs (t, m), the explicit separation of t allows usage of different spaces for tags and messages. Our HABS scheme further relies on a strongly unforgeable *one-time signature* $\mathtt{OTS} := (\mathtt{KGen}, \mathtt{Sign}, \mathtt{Verify})$ [6], for which the signing oracle can be queried only once and the adversary succeeds even if it can output a different signature on the message that it queried. Finally, our HABS construction uses *non-interactive zero-knowledge proofs* $\mathtt{NIZK} = (\mathtt{Setup}, \mathtt{Prove}, \mathtt{Verify}, \mathtt{SimSetup}, \mathtt{Sim})$ [5, 23] for a language $\mathcal{L} = \{x \mid \exists w.\ R(w, x) = 1\}$, where R is some relation over a witness w and a statement x. Typically, NIZK proofs require a common reference string crs output during the setup phase. From NIZK we require the standard properties of completeness, soundness, and zero-knowledge.

3.2 Generic Construction

High-Level Overview. We use the above general building blocks to construct our $\mathtt{HABS}$ scheme, which is specified in Fig. 6. In the following we provide a high-level intuition behind its construction. Attribute authorities (RA and IAs) generate their key pairs (ask_i, apk_i), $i \in \mathbb{N}_0$ for the tagged signature scheme $\mathtt{TS}$. The TA holds a key pair $(tsk,\ tpk)$ for the public key encryption scheme $\mathtt{PKE}$. The public parameters pp of the scheme also contain trusted common reference strings crs_1 and crs_2 for the corresponding NIZK proofs.

Attributes $a \in \mathbb{A}$ are viewed as tags of the $\mathtt{TS}$ scheme whereas delegation paths $attL := (apk_0, \ldots, \{apk_i, \{upk_{i+1}, \star\}\})$ are treated as messages. In order to create a warrant **warr** for some authority or signer, the corresponding IA with its ask_i will produce a $\mathtt{TS}$ signature on each attribute a and its delegation path and include this signature into $\mathbf{warr}[a]$ as part of the list $sigL$. Thus, a separate TS signature is used for each attribute and its path such that the signer can later reduce its **warr** to attributes that are needed for a policy Ψ.

Each signer, after initialisation, holds a key pair (usk, upk) for the digital signature scheme DS. A signer with usk and a reduced **warr** that satisfies Ψ can generate a HABS signature σ for some message m. The reduced **warr** together with the signer's public key upk and a digital signature σ_s with message $otsvk$ are encrypted in a PKE ciphertext C under the TA's public key tpk with randomness μ. The signer generates a key pair $(otssk, otsvk)$ for the one-time signature scheme OTS and uses its usk to compute a digital signature σ_s on $otsvk$.

We model Ψ as a span program $\mathbf{S}$ with a labelling function ρ that maps rows from $\mathbf{S}$ to attributes in $\mathbb{A}$. The signer attests the satisfiability of Ψ w.r.t its attributes from the reduced **warr** by computing a vector of integers $\mathbf{z}$ such that $\mathbf{zS} = [1, 0, \ldots, 0]$ and for any $z_i \neq 0$ we have $\rho(i) \in \mathbf{warr}$. Then, the signer computes a $\texttt{NIZK}_1$ proof π for the statement $(\mathrm{C}, otsvk, tpk, apk_0, \Psi)$ using as witness previously computed $(upk, \mu, \mathbf{z}, \mathbf{warr}, \sigma_s)$ such that the following relation is satisfied:

$$\begin{gathered}
\texttt{PKE.Enc}(tpk, (upk, \mathbf{warr}, \sigma_s, otsvk); \mu) = \mathrm{C} \wedge \texttt{DS.Verify}(upk, otsvk, \sigma_s) \\
\wedge\ \mathbf{zS} = [1, 0, \ldots, 0]\ \wedge \big(\forall i.\ z_i \neq 0 \implies a_i = \rho(i) \\
\wedge\ ((apk_0, apk_{i_1}, \ldots, apk_{i_n}, upk, \star)(\sigma_{i_1}, \ldots, \sigma_{i_n}, \sigma_u)) = \mathbf{warr}[a_i] \\
\wedge (\forall 1 \leq j \leq n.\ \texttt{TS.Verify}(apk_{i_{(j-1)}}, \sigma_{i_j}, a_i, (apk_0, apk_{i_1}, \ldots, apk_{i_j}))) \\
\wedge \texttt{TS.Verify}(apk_{i_n}, \sigma_u, a_i, (apk_0, apk_{i_1}, \ldots, apk_{i_n}, upk, \star))\big).
\end{gathered}$$

The resulting HABS signature σ contains the aforementioned C, π, and $otsvk$, along with an OTS signature σ_o, generated using $otssk$ to bind these value together with the message m and Ψ. The validity of such HABS signature σ can be verified using public parameters of the scheme and RA's public key apk_0 by checking the validity of the $\texttt{NIZK}_1$ proof π and the OTS signature σ_o.

The tracing algorithm, on input of a valid HABS signature $((\sigma_o, \mathrm{C}, \pi, otsvk), \mathrm{m}, \Psi)$ uses tsk to decrypt the warrant **warr** from the ciphertext C. The decrypted warrant contains all attributes and delegation paths, incl. signer's public key upk, and signature σ_s.In addition, TA outputs a $\texttt{NIZK}_2$ proof $\hat{\pi}$ for the statement $(otsvk, \mathrm{C}, tpk, (apk_0, \mathbf{warr}, \sigma_s))$ using tsk as its witness to prove the following relation:

$$\texttt{PKE.Dec}(tsk, \mathrm{C}) = (upk, \mathbf{warr}, \sigma_s, otsvk).$$

The output of TA on a valid HABS signature can be publicly judged by checking the validity of the $\texttt{NIZK}_2$ proof $\hat{\pi}$.

4 Security Proofs

In this section we prove our main theorem by showing the construction in Fig. 6 satisfies *path anonymity*, *non-frameability*, and *path traceability* from assumptions underlying its cryptographic building blocks.

```
AttIssue($ask_i$, $\mathbf{warr}_i$, $A$, $\{apk_j|upk_j\}$)

1 :  $\mathbf{warr} := \emptyset$
2 :  for $a \in A$ do
3 :     $\sigma_a \leftarrow \texttt{TS.Sign}(ask_i, a, (apk_1, \ldots, apk_i, \{apk_j|\{upk_j, \star\}\}))$
4 :     $(attL, sigL) \leftarrow \mathbf{warr}_i[a]$
5 :     $\mathbf{warr}[a] \leftarrow (attL \cup \{apk_j|\{upk_j, \star\}\}, sigL \cup \{\sigma_a\})$
6 :  return warr

Sign(($usk$, $\mathbf{warr}$), m, $\Psi$)

1 :  $(otsvk, otssk) \leftarrow \texttt{OTS.KGen}(1^\lambda)$
2 :  $\sigma_s \leftarrow \texttt{DS.Sign}(usk, otsvk)$
3 :  $\mathrm{C} \leftarrow \texttt{PKE.Enc}(tpk, (upk, \mathbf{warr}, \sigma_s, otsvk); \mu)$
4 :  compute $\mathbf{z}$ $s.t.$ $\mathbf{zS} = [1, 0, ..., 0]$
5 :  $\pi \leftarrow \texttt{NIZK}_1\texttt{.Prove}\big((upk, \mu, \mathbf{z}, \mathbf{warr}, \sigma_s) : (\mathrm{C}, otsvk, tpk, apk_0, \Psi)\big)$
6 :  $\sigma_o \leftarrow \texttt{OTS.Sign}(otssk, (\mathrm{m}, \Psi, \mathrm{C}, \pi))$
7 :  $\sigma \leftarrow (\sigma_o, \mathrm{C}, \pi, otsvk)$,  return $\sigma$

Verify($apk_0$, (m, $\Psi$, $\sigma$)) with $\sigma = (\sigma_o, \mathrm{C}, \pi, otsvk)$

return $\texttt{NIZK}_1\texttt{.Verify}((\mathrm{C}, otsvk, tpk, apk_0, \Psi), \pi) \wedge$
         $\texttt{OTS.Verify}(otsvk, (m, \Psi, \mathrm{C}, \pi), \sigma_o)$

Trace($tsk$, $apk_0$, (m, $\Psi$, $\sigma$)) with $\sigma = (\sigma_o, \mathrm{C}, \pi, otsvk)$

1 :  if $\texttt{Verify}(apk_0, (\sigma, \mathrm{m}, \Psi))$ then
2 :     $(upk, \mathbf{warr}, \sigma_s, otsvk') \leftarrow \texttt{PKE.Dec}(tsk, \mathrm{C})$
3 :     $\hat{\pi} \leftarrow \texttt{NIZK}_2\texttt{.Prove}(tsk\ : (otsvk, \mathrm{C}, tpk, upk, \mathbf{warr}, \sigma_s))$
4 :     if $otsvk' = otsvk$ then return $(upk, \mathbf{warr}, (\hat{\pi}, \sigma_s))$
5 :  return $\perp$

Judge($tpk$, $apk_0$, (m, $\Psi$, $\sigma$), ($upk$, $\mathbf{warr}$, $\hat{\pi}$)) with $\sigma = (\sigma_o, \pi, \mathrm{C}, otsvk)$

return $\texttt{Verify}(apk_0, (\sigma, \mathrm{m}, \Psi)) \wedge \texttt{NIZK}_2\texttt{.Verify}(tpk, (otsvk, \mathrm{C}, upk, \mathbf{warr}, \sigma_s), \hat{\pi})$
```

Fig. 6. Algorithms of our general HABS scheme.

Lemma 1 *The generic* HABS *construction from Fig. 6 offers path anonymity, if* $\texttt{NIZK}_1$ *and* $\texttt{NIZK}_2$ *are zero-knowledge,* PKE *is IND-CCA2, and* OTS *is strongly unforgeable.*

Proof. We follow a game-based approach and show that the advantage of the PPT adversary $\mathcal{A}$ in the path-anonymity experiment for the HABS construction from Fig. 6, is bounded by the advantages of the constructed adversaries for the underlying primitives. For simplicity, we assume that adversary $\mathcal{A}$ asks n user registration queries and the probability for sampling one of these users is $1/n$.

Game G_0: Let this be the experiment corresponding to $\mathbf{Exp}^{\text{pa-}b}_{\mathtt{HABS},\mathcal{A}}(\lambda)$ in Fig. 3, where the adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is required to distinguish between the signatures $\sigma_0 = (\sigma_o^0, \mathrm{C}_0, \pi_0, otsvk_0)$ and $\sigma_1 = (\sigma_o^1, \mathrm{C}_1, \pi_1, otsvk_1)$.

Game G_1: This game is obtained from the game G_0 where the restriction "$\mathcal{A}_2$ did not query $O_{\mathrm{Tr}}(m, \Psi, \sigma_b)$" is enforced by the O_{Tr} oracle available to $\mathcal{A}_2$. This is done by aborting the game, if $\mathcal{A}_2$ queries (m, Ψ, σ_b). We model this by adding the line "if $(\sigma_o, C, \pi, otsvk){=}(\sigma_{o,b}, C_b, \pi_b, otsvk_b)$ then return **abort**", when the adversary calls $O_{\mathrm{Tr}}(m, \Psi, (\sigma_o, C, \pi, otsvk))$ and $\sigma_b = (\sigma_{o,b}, C_b, \pi_b, otsvk_b)$. The games G_0 and G_1 preserve the same probability with respect to the adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$.

$$\Pr[G_1 = 1] = \Pr[G_0 = 1] .$$

Game G_2: We define G_2 as game G_1 except on the outputs of O_{Tr}, where we replace the $\mathtt{NIZK}_2$ proof $\hat{\pi}$ with a proof $\hat{\pi}'$, provided by the simulator $\mathtt{NIZK}_2.\mathtt{Sim}$. Additionally, in game G_2 for $\mathtt{NIZK}_2$ we replace `Setup` by `SimSetup`. These changes are done to avoid the case where $\mathcal{A}$ may "extract" tsk from $\mathtt{NIZK}_2$ proofs. Thus, for all future O_{Tr} oracle call we make use of a simulated $\mathtt{NIZK}_2$ proof. We bind the probability of $\mathcal{A}$ to distinguish between these games by the advantage of the (multi-theorem) zero-knowledge adversary $\mathcal{B}_{nizk2}$ for $\mathtt{NIZK}_2$. Hence,

$$|\Pr[G_2 = 1] - \Pr[G_1 = 1]| \leq \mathsf{Adv}^{\mathrm{zk}}_{\mathcal{B}_{zk},\mathtt{NIZK}_2}(\lambda).$$

Game G_3: Let G_3 be the game obtained from G_2 where the real $\mathtt{NIZK}_1$ proof π_b from the challenge signature $\sigma_b = (\sigma_{o,b}, C_b, \pi_b, otsvk_b)$ is replaced with the simulated proof π_b' by calling $\mathtt{NIZK}_1.\mathtt{Sim}$ on the inputs $(\mathrm{C}_b, otsvk_b, tpk, apk_0, \Psi)$. Similar to the previous step, by now for $\mathtt{NIZK}_1$ we replace `Setup` by `SimSetup`. We bound the capabilities of the adversary $\mathcal{A}$ to distinguish between games G_2 and G_3 by the advantage of the zero-knowledge adversary $\mathcal{B}_{nizk1}$ for the $\mathtt{NIZK}_1$ proof system. The adversary $\mathcal{B}_{nizk1}$ executes all steps from game G_3, except for the calls to $\mathtt{NIZK}_1$ that he replaces with oracle requests. There is only one sign challenge request performed only when the adversary $\mathcal{B}_{nizk1}$ constructs the signature σ_b. Hence,

$$|\Pr[G_3 = 1] - \Pr[G_2 = 1]| \leq \mathsf{Adv}^{\mathrm{zk}}_{\mathcal{B}_{nizk1},\mathtt{NIZK}_1}(\lambda).$$

Game G_4: Game G_4 is identical to game G_3, except we abort if $\mathcal{A}_2$ queries $O_{\mathrm{Tr}}(m, \Psi, (\sigma_o, \mathrm{C}, \pi, otsvk))$ if $(\mathrm{C}, otsvk) = (\mathrm{C}_b, otsvk_b)$. The adversary $\mathcal{A}$ is able to distinguish between G_3 and G_4, only if he can produce a valid `OTS` signature σ_o for a statement (C_b, π, m, Ψ) and verification key $otsvk_b$, without knowledge of the signing key $otssk_b$. Essentially, breaking the strong unforgeability of `OTS`.

We bound the capabilities of the adversary $\mathcal{A}$ to distinguish between this two games by the advantage of the unforgeability adversary $\mathcal{B}_{ots}$ for the `OTS` scheme that uses $otsvk_b$ as the public key. The adversary $\mathcal{B}_{ots}$ on input $otsvk_b$ does all the steps described in game G_4, except he does not call O_{Sig} to produce σ_b, instead he uses his `OTS` signing oracle to receive a signature σ_o^b for $(m, \Psi, \mathrm{C}_b, \pi_b)$ and pass this to $\mathcal{A}_2$. Then, $\mathcal{B}_{ots}$ waits for $\mathcal{A}_2$ to provide a valid input to O_{Tr} that contains C_b, $otsvk_b$ and $\sigma_o \neq \sigma_o^b$ or $\pi \neq \pi_o$, and use this as his forgery.

$$|\Pr[G_4 = 1] - \Pr[G_3 = 1]| \leq \mathsf{Adv}^{\text{euf-cma}}_{\mathcal{B}_{ots},\mathtt{OTS}}(\lambda).$$

Game G_5: This game G_5 is the same as G_4, except we abort if $\mathcal{A}_2$ queries $O_{\mathrm{Tr}}(m, \Psi, (\sigma_o, \mathrm{C}, \pi, otsvk))$ when $\mathrm{C} = \mathrm{C}_b$. The output of O_{Tr} remains unchanged between these two games, as the oracle return $\perp$ if $otsvk_b$ from C is different from $otsvk$ received as input. Game G_5 preserves the same probability as G_4:

$$\Pr[G_5 = 1] = \Pr[G_4 = 1].$$

Game G_6: Let G_6 be the game obtained from G_5 where the ciphertext C_b from the challenge signature $\sigma_b = (\sigma_o^b, C_b, \pi_b', otsvk_b)$ is replaced with the C_0. The distinguishing capabilities of the adversary $\mathcal{A}_2$, is bounded by the advantage of the IND-CCA2 adversary $\mathcal{B}_{ind}$ for the `PKE` encryption scheme. This only applies for the case $b = 1$. The adversary $\mathcal{B}_{ind}$ performs all the steps in game G_6 except the ones that require interaction with `PKE`, where he uses the oracle he has access to. A single challenge query is performed between the calls to $\mathcal{A}_1$ and $\mathcal{A}_2$ calls. $\mathcal{A}_2$ may ask O_{Tr} queries for any signature that does not contain the challenge ciphertext, and $\mathcal{B}_{ind}$ answers them by calling the decryption oracle.

$$|\Pr[G_5 = 1] - \Pr[G_6 = 1]| \leq \mathsf{Adv}^{\text{ind-cca2}}_{\mathcal{B},\mathtt{PKE}}(\lambda).$$

The experiment G_6 provides as challenge to $\mathcal{A}$ the exact same values independent of the random bit b that $\mathcal{A}$ is asked to guess. Additionally, due to zero-knowledge of $\mathtt{NIZK}_2$ used in G_1, $\mathcal{A}$ does not have access to tsk. Therefore, the probability the adversary wins game G_6 is $\frac{1}{2}$ and hence the advantage of $\mathcal{A}$ to win this experiment is 0.

$$\Pr[G_6 = 1] = \frac{1}{2}.$$

From the sequence of games above, the result of this lemma follows. □

Lemma 2 *The generic* `HABS` *construction from Fig. 6 is non-frameable, if* $\mathtt{NIZK}_1$ *is sound,* `TS` *and* `DS` *are unforgeable, and* `OTS` *is strongly unforgeable.*

Proof. We model our proof by dividing the non-frameability experiment from Fig. 4 into two experiments based on the winning condition of the adversary $\mathcal{A}$. The first experiment, denoted E_1, is defined in Fig. 7, and captures the probability of the adversary $\mathcal{A}$ to create a forgery. For readability, the algorithms `Verify` and `Judge` are inlined. The second experiments E_2 follows the exact same steps as E_1 except that "$j \in HU \ \wedge \mathcal{A}$ did not query $O_{\mathrm{Sig}}((usk_j, \mathbf{warr}), \Psi, \mathrm{m})$" is replaced by

$''\exists a.\ a \in \mathbf{warr} \implies (apk_0, apk_1, \ldots, apk_n, upk_j, \star) = \mathbf{warr}[a]\ \wedge$
$(\ (\exists 0 \le i \le n-1.\ \mathcal{A}$ did not call $O_{\mathrm{Att}}(i,\ \cdot\ , a, apk_{i+1}) \wedge i \in HU)\ \vee$
$(\mathcal{A}$ did not call $O_{\mathrm{Att}}(n,\ \cdot\ , a, upk_u)\ \wedge n \in HU))''$

The probability of winning the non-frameability experiment is bounded by the probability of $\mathcal{A}$ winning either E_1 or E_2:

$$\Pr\left[\mathbf{Exp}^{\mathrm{nf}}_{\mathtt{HABS},\mathcal{A}}(\lambda)\right] \le \Pr[E_1 = 1] + \Pr[E_2 = 1]\,.$$

E_1 - The $\mathbf{Exp}^{\mathrm{nf}}_{\mathtt{HABS},\mathcal{A}}(\lambda)$ where $\mathcal{A}$ did not query $O_{\mathrm{Sig}}((usk, \mathbf{warr}), \Psi, \mathrm{m})$

```
1 :  $(\mathrm{pp}, ask_0, tsk) \leftarrow \mathtt{Setup}(1^\lambda)$
2 :  $((m, \Psi, \sigma), (upk_j, \mathbf{warr}, (\pi, \hat{\sigma}_s))) \leftarrow \mathcal{A}(\mathrm{pp}, ask_0, tsk : O_{\mathrm{Att}}, O_{\mathrm{Sig}}, O_{\mathrm{Corr}}, O_{\mathrm{Reg}})$
3 :  $(\sigma_o, \mathrm{C}, \pi, otsvk) = \sigma$
4 :  if $\mathtt{NIZK}_1.\mathtt{Verify}((\mathrm{C}, otsvk, tpk, apk_0, \Psi), \pi)\ \wedge$
5 :    $\mathtt{OTS}.\mathtt{Verify}(otsvk, (m, \Psi, \mathrm{C}, \pi), \sigma_o)\ \wedge$
6 :    $\mathtt{NIZK}_2.\mathtt{Verify}(tpk, (otsvk, \mathrm{C}, upk_j, \mathbf{warr}, \sigma_s), \hat{\pi}) \wedge$
7 :    $j \in HU\ \wedge$
8 :    $\mathcal{A}$ did not query $O_{\mathrm{Sig}}((usk_j, \mathbf{warr}), \Psi, \mathrm{m})$ then
9 :    return 1
10 : return 0
```

Fig. 7. Experiment E_1

We start with the first experiment E_1 that we will show has a negligible probability of success. Intuitively, we want to argue over the values $(upk', \mathbf{warr}', m', \Psi'), (\sigma'_o, \mathrm{C}', \pi', otsvk')$ that correspond to the input and output of the O_{Sig} oracle. We show that they are not sufficient for the adversary $\mathcal{A}$ to create valid proofs and signatures $(\sigma_o, \mathrm{C}, \pi, otsvk)$ for the values $(upk, \mathbf{warr}, m, \Psi)$ different from $(upk', \mathbf{warr}', m', \Psi')$. More precisely, we take each element of the tuple $(upk_j, \mathbf{warr}, m, \Psi)$ and try to reason about their relation with their prime counterpart from $(upk', \mathbf{warr}', m', \Psi')$.

The first step considers if "there has been any O_{Sig} request that contains upk_j", which sets the direction for the rest of the proof. Next, we follow the same methodology by reasoning that the values $\mathbf{warr}', m', \Psi'$ and $otsvk'$ have to coincide with $\mathbf{warr}, m, \Psi, otsvk$ for $\mathcal{A}$ to actually produce valid proofs and signatures that pass the verification conditions in E_1. For simplicity, we consider the probability of any adversary to guess which oracle constructs the keys for a particular user is $1/n$, given n user registration oracle calls.

Game G_0. The game G_0 is defined exactly as E_1 except on line "$\mathcal{A}$ did not query $O_{\mathrm{Sig}}((usk_j,\ \mathbf{warr}), m, \Psi)$" that is replaced with a membership check $(upk_u, \mathbf{warr}, m, \Psi) \notin sL$ for the list sL. This list sL is initialized empty at the beginning of the experiment, and gets updated with the inputs of the O_{Sig} oracle. Additionally, we introduce the list spL that stores the input and output of the O_{Sig} oracle. We have that E_1 and G_0 have the same probability.

$$\Pr[E_1 = 1] = \Pr[G_0 = 1] .$$

Game G_1. This game is defined exactly as G_0 with the exception the additional test $\texttt{DS.Verify}(upk_j, otsvk, \sigma_s)$ performed over the output of the adversary $(((\sigma_o, \mathrm{C}, \pi, otsvk), m, \Psi), (upk_j, \mathbf{warr}, (\hat{\pi}, \sigma_s)))$. G_1 is indistinguishable from G_0 due to the soundness of $\texttt{NIZK}_1$: the probability of generating a valid $\texttt{NIZK}_1$ proof for a false statement (that does not pass $\texttt{DS}$ verification).

Next, we reason if the adversary has submitted any O_{Sig} request that contains upk, and use the list sL to check this. We split the probability in game G_1 along two cases [1]:

$$\Pr[G_1 = 1] = \Pr[G_1 = 1 \wedge (upk_j, \star, \star, \star) \in sL] + \Pr[G_1 = 1 \wedge (upk_j, \star, \star, \star) \notin sL] .$$

Game G_2. The game G_2 is obtained from G_1 by adding the condition $(upk_j, \star, \star, \star) \notin sL$. The adversary in G_2 managed to create a valid digital signature σ_s for upk_u that passes $\texttt{DS}$ verification without having access to the user's secret key (as $j \in HU$).

The capabilities of $\mathcal{A}$ in this experiment are bounded by the advantage of an unforgeability adversary $\mathcal{B}_s$ against the digital scheme $\texttt{DS}$ that uses upk_j. The coefficient $1/n$ is due to linking upk_j with the user for whom $\mathcal{A}$ produces the forgery.

$$\Pr[G_2 = 1] = \Pr[G_1 = 1 \wedge (upk_j, \star, \star, \star) \notin sL] \leq \frac{1}{n} \times \mathsf{Adv}^{\text{euf-cma}}_{\mathcal{B}_{ds}, \texttt{DS}}(\lambda).$$

Game G_3. This game uses the exact steps performed by game G_1, but in the setting where $\mathcal{A}$ requested at least one signature that contains user upk_j. There exists an adversary query $((upk_j, \mathbf{warr}', m', \Psi'), (\sigma_o', \mathrm{C}', \pi', otsvk')) \in spL$ with $(\mathbf{warr}, m, \Psi) \neq (\mathbf{warr}', m', \Psi')$.[2] Hence,

$$\begin{aligned}\Pr[G_3 = 1] &= \Pr[G_1 = 1 \wedge (upk_j, \star, \star, \star) \in sL] \\ &= \Pr\left[G_1 = 1 \wedge (upk_j, \mathbf{warr}', m', \Psi'), (\sigma_o', \mathrm{C}', \pi', otsvk')) \in spL\right] .\end{aligned}$$

Using the method applied on G_1, we reason on the relation between the $\texttt{OTS}$ public keys $otsvk$ and $otsvk'$. We split game G_3 based on $otsvk = otsvk'$ and $otsvk \neq otsvk'$.

$$\Pr[G_3 = 1] = \Pr[G_3 = 1 \wedge otsvk = otsvk'] + \Pr[G_3 = 1 \wedge otsvk \neq otsvk'] .$$

Game G_4. We define G_4 as the game G_3 where $otsvk \neq otsvk'$. In this case, the adversary $\mathcal{A}$ is able to provide a forgery for the $\texttt{DS}$ scheme by signing

[1] We use $\star$ to symbolize the existence of variables whose values we are not interested.
[2] This is due to $(upk_j, \mathbf{warr}, m, \Psi) \notin sL$ and $(upk_j, \mathbf{warr}', m', \Psi') \in sL$.

$otsvk'$ without knowledge of usk_j. This is similar to the method of computing the bound for G_2, except that now $\mathcal{A}$ asks signature queries for upk.

We can bind the capabilities of adversary $\mathcal{A}$ in this game, by constructing a forger $\mathcal{B}'_{ds}$ for the DS signing scheme. $\mathcal{B}'_{ds}$ is identical to $\mathcal{B}_{ds}$ except on O_{Sig} queries for upk_j, that $\mathcal{B}'_{ds}$ answers using his DS.Sign oracle queries. We have,

$$\Pr[G_4 = 1] = \Pr[G_3 = 1 \wedge otsvk \neq otsvk'] \leq \frac{1}{n} \times \mathsf{Adv}^{\text{euf-cma}}_{\mathcal{B}'_{ds},\mathtt{DS}}(\lambda).$$

Game G_5**.** The game G_5 uses the steps of G_3 with the additional restriction $otsvk' = otsvk$. With the $upk_j = upk'$ restriction from G_3 we further transform this game in the view of G_1.

$$\begin{aligned}\Pr[G_5 = 1] &= \Pr[G_3 = 1 \wedge otsvk' = otsvk] \\ &= \Pr\left[G_1 = 1 \wedge (upk_j, \mathbf{warr}', m', \Psi'), (\sigma'_o, \mathrm{C}', \pi', otsvk)) \in spL\right].\end{aligned}$$

Currently, we reduced that $\mathcal{A}$ has made a O_{Sig} query for $(upk_j, \mathbf{warr}', m', \Psi')$ different from $(upk_j, \mathbf{warr}, m, \Psi)$, but with the same OTS signature public key $otsvk$. We split the probability in G_5 based on the equality test between (m, Ψ) and (m', Ψ').

$$\Pr[G_5 = 1] = \Pr[G_5 = 1 \wedge (m, \Psi) = (m', \Psi')] + \Pr[G_5 = 1 \wedge (m, \Psi) \neq (m', \Psi')].$$

Game G_6**.** We define game G_6 as the game G_5 where $(m, \Psi) \neq (m', \Psi')$. That is, the adversary $\mathcal{A}$ is able to provide a forgery for the OTS scheme by signing a message that contains (m', Ψ') without knowledge of $otssk$.

The capabilities of adversary $\mathcal{A}$ in this case, are bounded by the advantage of the unforgeability adversary $\mathcal{B}_{ots}$ for the OTS signature scheme that uses $otssk$ as the secret key. There might be a slight loss of accuracy as the $\mathcal{B}_{ots}$ needs to identify which is the O_{Sig} query that uses (m', Ψ') among all O_{Sig} queries for upk_j. He is able to do this with probability $1/k$ if $\mathcal{A}$ makes k sign queries for the same upk_j value. The factor $1/n$ is given by the guess $\mathcal{B}_{ots}$ makes on which is the user registration oracle with upk.

$$\Pr[G_6 = 1] = \Pr[G_5 = 1 \wedge (m, \Psi) \neq (m', \Psi')] \leq \frac{1}{n} \times \frac{1}{k} \times \mathsf{Adv}^{\text{euf-cma}}_{\mathcal{B}_{ots},\mathtt{OTS}}(\lambda).$$

Game G_7**.** We define game G_7 as the game G_5 where $(m, \Psi) = (m', \Psi')$. Because of the $(\mathbf{warr}, m, \Psi) \neq (\mathbf{warr}', m', \Psi')$ restriction, this leads to $\mathbf{warr}' \neq \mathbf{warr}$. Going a little bit further, and including the condition added by game G_5 with respect to game G_1, we have

$$\begin{aligned}\Pr[G_7 = 1] &= \Pr[G_5 = 1 \wedge (m, \Psi) = (m', \Psi')] \\ &= \Pr\left[G_1 = 1 \wedge (upk_j, \mathbf{warr}', m, \Psi), (\sigma'_o, \mathrm{C}', \pi', otsvk)) \in spL\right].\end{aligned}$$

Given $\mathbf{warr} \neq \mathbf{warr}'$, we now show that $\mathrm{C} \neq \mathrm{C}'$. This is guaranteed by the correctness property of the encryption scheme PKE that builds C'. Let $m_0 = (upk_j, \mathbf{warr}, \sigma_s, otsvk)$ and $m_1 = (upk_j, \mathbf{warr}', \sigma'_s, otsvk)$ be two different messages that both encrypt to C'. According to the correctness of PKE, C' must decrypt with overwhelming probability to one of the two message. We divide the probability of G_6 based on the equality of C and C$'$.

$$\Pr[G_7 = 1] = \Pr\left[G_7 = 1 \wedge \mathrm{C} = \mathrm{C}'\right] + \Pr\left[G_7 = 1 \wedge \mathrm{C} \neq \mathrm{C}'\right].$$

Game G_8. Let G_8 be the game defined by G_7 with $\mathrm{C} \neq \mathrm{C}'$. In such a case, the adversary $\mathcal{A}$ is able to create a forgery without knowledge of *otssk*, that passed the verification in the body of experiment G_7.

The probability of success for adversary $\mathcal{A}$ in this game, is bounded by the advantage of the OTS forger $\mathcal{B}'_{ots}$ which behaves exactly as $\mathcal{B}_{ots}$ from game G_6. The difference in this case is given by the output of the adversary. Here, $\mathcal{A}$ provides an OTS signature that satisfies $\mathrm{C} \neq \mathrm{C}'$, while in G_6 the one-time signature is for $(m, \Psi) \neq (m', \Psi')$. Hence,

$$\Pr[G_8 = 1] = \Pr\left[G_6 = 1 \wedge \mathrm{C} \neq \mathrm{C}'\right] \leq \mathsf{Adv}^{\text{euf-cma}}_{\mathcal{B},\mathtt{OTS}}(\lambda).$$

Game G_9. The game G_9 is defined as G_7 where $\mathrm{C} = \mathrm{C}'$. In such a case, the adversary $\mathcal{A}$ has managed to produce a ciphertext C that decrypts to two different messages $m_0 = (upk_j, \mathbf{warr}, \sigma_s, otsvk)$ and $m_1 = (upk_j, \mathbf{warr}', \sigma'_s, otsvk)$. We build $\mathcal{B}_{corr}$ that performs the steps in G_7 and waits for $\mathcal{A}$ to provide an output $(((\sigma_o, \mathrm{C}, \pi, otsvk), m, \Psi), (upk_j, \mathbf{warr}, (\hat{\pi}, \sigma_s)))$. Then, he uses that output to construct message m_0, and looks through the list *spL* for the query the adversary $\mathcal{A}$ has made that produced the same ciphertext C and builds m_1. $\mathcal{B}_{corr}$ outputs the message that does not appears when he does a decryption. For simplicity, this adversary also provides the randomness needed to produce the same ciphertext in the body of the correctness experiment.We have,

$$\Pr[G_9 = 1] = \Pr\left[G_7 = 1 \wedge \mathrm{C} = \mathrm{C}'\right] \leq \mathsf{Adv}^{\text{correctness}}_{\mathcal{B}_{corr},\mathtt{PKE}}(\lambda).$$

From the sequence of games starting $G_0, \ldots, G_9$, it follows that the probability of E_1 is bounded by unforgeability of DS, OTS, and zero-knowledge of $\mathtt{NIZK}_1$.

The experiment E_2 deals with the case where the adversary $\mathcal{A}$ is able to provide a forged TS signature for an honest authority apk_i and some attribute a. The capabilities of the adversary $\mathcal{A}$ in this case is bounded by the unforgeability adversary $\mathcal{B}_{ts}$ for TS. Hence,

$$\Pr[E_2 = 1] \leq \frac{1}{t} \times \mathsf{Adv}^{\text{euf-cma}}_{\mathcal{B}_{ts},\mathtt{TS}}(\lambda).$$

□

Lemma 3 *The generic* HABS *construction from Fig. 6 offers path traceability, if* $\mathtt{NIZK}_1$ *is sound and* TS *is unforgeable.*

Proof. We divide the advantage of the path traceability adversary $\mathcal{A}$ for the experiment $\mathbf{Exp}^{\text{tr}}_{\mathtt{HABS},\mathcal{A}}$ in Fig. 5 by the two winning conditions for the adversary:

```
$E_1$
1: $(\mathrm{pp}_, ask_0, tsk) \leftarrow \mathtt{Setup}(1^\lambda)$
2: $(((\sigma_o, \mathrm{C}, \pi, otsvk), m, \Psi), (upk, \mathbf{warr}, (\hat{\pi}, \sigma_s))) \leftarrow \mathcal{A}(\mathrm{pp}, tsk : O_{\mathrm{Att}}, O_{\mathrm{Corr}}, O_{\mathrm{Reg}})$
3: if $\mathtt{NIZK}_1.\mathtt{Verify}((\mathrm{C}, otsvk, tpk, apk_0, \Psi), \pi) \ \wedge\ \mathtt{OTS.Verify}(otsvk, (m, \Psi, \mathrm{C}, \pi), \sigma_o) \ \wedge$
4:     $\big(\mathtt{PKE.Dec}(tsk, \mathrm{C}) = \bot \vee\ \mathtt{NIZK}_2.\mathtt{Prove}(tsk : otsvk, \mathrm{C}, tpk, upk, \sigma_s) = \bot\big)$ then
5:     return 1
6: return 0

$E_2$
1: $(\mathrm{pp}_, ask_0, tsk) \leftarrow \mathtt{Setup}(1^\lambda)$
2: $(((\sigma_o, \mathrm{C}, \pi, otsvk), m, \Psi), (upk, \mathbf{warr}, (\hat{\pi}, \sigma_s))) \leftarrow \mathcal{A}(\mathrm{pp}, tsk : O_{\mathrm{Att}}, O_{\mathrm{Corr}}, O_{\mathrm{Reg}})$
3: if $\mathtt{NIZK}_1.\mathtt{Verify}((\mathrm{C}, otsvk, tpk, apk_0, \Psi), \pi) \ \wedge\ \mathtt{OTS.Verify}(otsvk, (m, \Psi, \mathrm{C}, \pi), \sigma_o) \ \wedge$
4:    $\mathtt{NIZK}_2.\mathtt{Verify}((otsvk, \mathrm{C}, tpk, upk, \mathbf{warr}, \sigma_s), \hat{\pi}) \wedge$
5:    $(\exists a.\ a \in \mathbf{warr} \implies (apk_0, apk_1, \ldots, apk_n, upk, \star) = \mathbf{warr}[a] \ \wedge$
6:            $(\ (\exists 0 \le i \le n-1.\ i \in HU \wedge (i+1, apk_{i+1}, ask_{i+1}) \notin List) \vee$
7:              $(n \in HU \wedge (\ \cdot\ , upk, usk) \notin List)\ )\ )$ then,   return 1
8: return 0
```

Fig. 8. Experiment Traceability

1. `Trace` fails for a valid `HABS` signature, in experiment E_1; and
2. there exists a signature in the warrant, for some attribute introduced for a "rogue" entity, that is not registered, by an honest authority, in experiment E_2.

We have,

$$\Pr\left[\mathbf{Exp}^{\mathrm{tr}}_{\mathtt{HABS},\mathcal{A}}(\lambda)\right] \le \Pr[E_1 = 1] + \Pr[E_2 = 1]\,.$$

We start with experiment E_1, where we use the soundness of $\mathtt{NIZK}_1$ to show that `PKE.Dec` and $\mathtt{NIZK}_2$ cannot fail.

Game G_0. The game G_0 is defined exactly as E_1 in Fig. 8. From this, it immediately follows that

$$\Pr[G_0 = 1] = \Pr[E_1 = 1]\,.$$

Game G_1. We define game G_1 as G_0, except that we add the line

$$''\exists \mu.\ \mathrm{C} = \mathtt{PKE.Enc}(tpk, (upk, \mathbf{warr}, \sigma_s, otsvk)) \wedge''$$

between lines 3 and 4. This new check performed by G_1 is one of the conditions encoded in the relation for $\mathtt{NIZK}_1$, and $\mathcal{A}$ is able to notice the difference between G_0 and G_1 if he can produce a valid $\mathtt{NIZK}_1$ proof for a false statement (that does not have a witness μ). We bind the probability of $\mathcal{A}$ to distinguish this games, by the advantage of the soundness adversary $\mathcal{B}_{sound}$ for $\mathtt{NIZK}_1$: $\mathcal{B}_{sound}$ performs all steps in G_1 with the exception of those that require interaction with $\mathtt{NIZK}_1$ where he uses his oracle access. Then, uses the $\mathtt{NIZK}_1$ proof π from the output of $\mathcal{A}$ as his output. Hence,

$$|\Pr[G_0 = 1] - \Pr[G_1 = 1]| \leq \mathsf{Adv}^{\text{sound}}_{\mathcal{B}_{sound},\mathtt{NIZK}_1}(\lambda).$$

Game G_2. This game uses the correctness of the PKE to replace the test performed over the encryption in G_1 with the test over the decryption

$$''\exists\mu.\ \mathtt{PKE}.Enc(tpk, (upk, \mathbf{warr}, \sigma_s, otsvk); \mu) = \mathrm{C} \text{ by}$$
$$\mathtt{PKE}.\mathrm{Dec}(tsk, \mathrm{C}) = (upk,\ \mathbf{warr},\ \sigma_s, otsvk)''.$$

Additionally, we remove the line "$\mathtt{PKE}.\mathsf{Dec}(tsk, \mathrm{C}) = \perp$" as this can be canceled by the change above. The difference between these two games is bounded by the probability of an decryption to fail after an encryption, even on adversarial valid inputs. We construct adversary $\mathcal{B}_{cor}$ for PKE that calls Setup and $\mathcal{A}$. Then, it uses $(tpk, (upk, \mathbf{warr}, \sigma_s)$ as the message and μ as the randomness.

$$|\Pr[G_1 = 1] - \Pr[G_2 = 1]| \leq \mathsf{Adv}^{\text{corr}}_{\mathcal{B}_{cor},\mathtt{PKE}}(\lambda).$$

Game G_3. This game returns directly 0. The basis for this is that it should be hard for a $\mathtt{NIZK}_2$ proof created over a statement and witness that belong to the relation, to be declared invalid by the verification algorithm. We quantify this difficulty, by building an adversary $\mathcal{B}_{com}$ that tries to win the completeness experiment for $\mathtt{NIZK}_2$. $\mathcal{B}_{com}$ calls Setup and $\mathcal{A}$, then builds the statement $(otsvk, \mathrm{C}, tpk, upk, \mathbf{warr}, \sigma_s)$ that is used to create a proof that will fail verification.

$$\Pr[G_2 = 1] = |\Pr[G_2 = 1] - \Pr[G_3 = 1]| \leq \mathsf{Adv}^{\text{complete}}_{\mathcal{B}_{com},\mathtt{NIZK}_2}(\lambda).$$

From the sequence of games starting $G_0, \ldots, G_3$, it follows that the probability of E_1 is bounded by the soundness of $\mathtt{NIZK}_1$.

Experiment E_2 models the setting where adversary $\mathcal{A}$ is able to provide a valid **warr** that contains valid TS signatures from an honest authority apk_i to an entity, apk_{i+1} or upk, not registered - not in $List$. This means, he is able to forge signatures under the name of apk_i without knowledge of the corresponding secret key ask_i. We bind the probability of $\mathcal{A}$ to win by the advantage of the forger $\mathcal{B}_{ts}$ for TS. First, $\mathcal{B}_{ts}$ needs to guess which authority $\mathcal{A}$ would use to forge a signature in the warrant, and does that with probability $1/t$ given exactly t registration queries for authorities. Then, he uses the challenge public key for that authority, and answers all queries that $\mathcal{A}$ makes for that authority with his TS signing oracle calls. Finally, $\mathcal{B}_{ts}$ looks at the output of $\mathcal{A}$ for signatures that contain that authority and has not been provided by his signing oracle.

$$\Pr[E_2 = 1] \leq \frac{1}{t} \times \mathsf{Adv}^{\text{euf-cma}}_{\mathcal{B}_{ts},\mathtt{TS}}(\lambda).$$

$\square$

Theorem 1. *The* HABS *construction in Fig. 6 offers path anonymity, non-frameability, and path traceability under the assumptions that* PKE *is IND-CCA2 secure,* TS *and* DS *are unforgeable,* OTS *is strongly unforgeable,* $\mathtt{NIZK}_1$ *and* $\mathtt{NIZK}_2$ *are both sound zero-knowledge proofs.*

Proof. The proof follows from Lemmas 1, 2 and 3.

4.1 Instantiating the HABS Building Blocks

Instantiation. We instantiate HABS in the bilinear group setting. For the digital signature DS we use the constant-sized structure preserving scheme by Abe et al. [1], whereas for TS we use their unbounded-message version of their scheme. These are unforgeable under the Simultaneous Flexible Pairing (SFP) [1] assumption. We use an encryption scheme by Camenish et al. [10] that is capable of encrypting message vectors for our IND-CCA2 PKE, which relies on the DLIN assumption. Finally, for the one-time signature OTS we use the full Boneh-Boyen signature scheme [6], which is strongly unforgeable under the q-Strong Diffie-Hellman (q-SDH) assumption.

For the proofs $\mathtt{NIZK}_1$ and $\mathtt{NIZK}_2$, we use Groth-Sahai (GS) proof systems [23], the security of which is also based on the DLIN assumption in the symmetric setting. These are efficient, non-interactive proof systems in the CRS model that are complete, sound, and zero-knowledge. Briefly, the GS proof system works by commiting to the elements of the witness and then showing they satisfy the source equation. The equation must take the form of either a Pairing Product Equation (PPE), a Multi-Scalar Multiplication Equation (MSME) or a Quadratic Equation (QE). We refer to [23] for full details and give an overview of our constructions for $\mathtt{NIZK}_1$ and $\mathtt{NIZK}_2$ in Appendix A.

Efficiency. We briefly consider the efficiency of our HABS scheme. For our instantiation of OTS, the public key requires 4 group elements and the short signature only requires 3 elements from $\mathbb{G}$ and one element from $\mathbb{Z}_p$. The ciphertext C computed using PKE requires $n + 8$ elements from $\mathbb{G}$, where n is the number of elements in the public keys and tagged-signatures from the delegation paths in the warrant. However, the size of TS used to delegate and issue attributes depends linearly on the distance of the intermediate authority from the root authority in the delegation path, simply because the number of messages (authorities' public keys) increases by one with each delegation. Therefore, the proof $\mathtt{NIZK}_1$ that includes a proof that the warrant contains a valid path also grows linearly in this parameter.

To prove satisfiability of the signing predicate Ψ, a proof containing 2β elements from $\mathbb{Z}_p$ is constructed, where β is the size of the span program $\mathbf{S}$. The proof that DS verifies is of constant size and requires 72 elements of $\mathbb{G}$.

Finally, the size of the proof in $\mathtt{NIZK}_1$ that C was encrypted correctly is linear in the number of delegations in the warrant, this is inevitable since we need to prove the validity for each authority-signature pair on the delegation path. Similarly, this is also the case for the proof of correctness for decryption of C in $\mathtt{NIZK}_2$.

We note that if we consider HABS in the setting where the maximum delegation path of an attribute has length 1, then the size of a HABS signature is linear in the size of the policy Ψ, which is consistent with other ABS schemes that also offer flexible signing policies, e.g., [30, 17, 21].

4.2 Other properties

In the following we discuss some further properties that can be adopted within our general HABS construction.

Revocation. Our generic HABS construction can be extended to support revocation of attribute authorites and users by means of public revocations lists RL authenticated by the root authority. These lists would include public keys of revoked authorities and users. To enable detection of revoked entities upon verification of HABS signatures, the proof $\mathtt{NIZK}_1$ can be extended to prove that for all attributes used to satisfy the policy none of the public keys in the corresponding delegation paths within the signer's warrant **warr** is included into these lists. Since HABS signatures hide delegation paths this approach would preserve privacy by ensuring that no verifier can identify the revoked signer. Due to its complexity, $O(r \sum_a |\mathbf{warr}[a]|)$ where r is the number of revoked public keys, this method might not scale well and hence finding more efficient revocation mechanisms can be seen as an interesting open problem.

Independent hierarchies. Assume there are multiple HABS hierarchies, each managed by an independent root authority, and any (intermediate) authority or user should be able to receive attributes from different such hierarchies. Our general HABS construction naturally supports this scenario. In particular, warrants can include attributes (along with their signed delegation paths) that were issued to the entity by authorities belonging to other hierarchies and consequently the proof $\mathtt{NIZK}_1$ can enable generation of HABS signatures for predicates Ψ requiring possession of attributes from these hierarchies.

5 Conclusion

The notion of Hierarchical ABS (HABS) introduced in this paper extends the functionality for existing (multi-authority) ABS schemes with some useful properties that can help to expand the application domain of ABS signatures, e.g. to intelligent transport systems. The extended properties of HABS include: (1) support for dynamically expandable hierarchical formation of attribute authorities, managed by some root authority, (2) hierarchical delegation of attribute-issuing rights amongst the authorities, (3) the ability to issue attributes to signers by multiple authorities, possibly located at different levels of the hierarchy, (4) generated ABS signatures that hide signers, their attributes together with their delegation paths, (5) support for a publicly verifiable tracing procedure that enables accountability for all entities that were involved in the delegation and issue of an attribute to a signer. This brings ABS schemes closer to traditional hierarchically-organised PKIs while preserving the valuable privacy properties and security guarantees of the attribute-based setting. The proposed generic HABS construction makes use of standard cryptographic building blocks that can be instantiated in the setting of bilinear maps based on the DLIN, q-SDH and SFP assumptions. We discussed further how our HABS construction offers natural support for scenarios where the same authority or user is admitted to multiple, independently managed hierarchies and needs to bundle attributes obtained in these hierarchies to satisfy some predicate, and how it can be extended to revoke attribute authorities and signers.

Acknowledgements. Daniel Gardham was supported by the UK Government PhD studentship scheme. Constantin-Cătălin Drăgan was supported by the EPSRC project TAPESTRY (EP/N02799X).

References

1. M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev, and M. Ohkubo. Structure-Preserving Signatures and Commitments to Group Elements. In *CRYPTO 2010*, pages 209–236, 2010. https://eprint.iacr.org/2010/133.
2. M. Backes, S. Meiser, and D. Schröder. Delegatable functional signatures. In *PKC (1)'16*, pages 357–386, 2016.
3. M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Lysyanskaya, and H. Shacham. Randomizable Proofs and Delegatable Anonymous Credentials. In *CRYPTO 2009*, pages 108–125. LNCS 5677.
4. M. Bellare and G. Fuchsbauer. Policy-based signatures. In *PKC 2014*, pages 520–537, 2014. LNCS 8383.
5. M. Blum, P. Feldman, and S. Micali. Non-interactive Zero-knowledge and Its Applications. In *STOC'88*, pages 103–112, 1988.
6. D. Boneh and X. Boyen. Short Signatures Without Random Oracles. In *EUROCRYPT 2004*, pages 56–73. LNCS 3027, 2004.
7. X. Boyen. Mesh Signatures. In *EUROCRYPT 2007*, pages 210–227. LNCS 4515.
8. E. Boyle, S. Goldwasser, and I. Ivan. Functional signatures and pseudorandom functions. In *PKC 2014*, pages 501–519, 2014.
9. J. Camenisch, M. Drijvers, and M. Dubovitskaya. Practical uc-secure delegatable credentials with attributes and their application to blockchain. In *ACMCCS' 17*, pages 683–699, 2017.
10. J. Camenisch, K. Haralambiev, M. Kohlweiss, J. Lapon, and V. Naessens. Structure Preserving CCA Secure Encryption and Its Application to Oblivious Third Parties. Cryptology ePrint Archive, Report 2011/319, 2011.
11. J. Camenisch, I. Krontiris, A. Lehmann, G. Neven, C. Paquin, K. Rannenberg, and H. Zwingelberg. H2.1 abc4trust architecture for developers. abc4trust.eu, 2011.
12. J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In *SCN 2002*, SCN 2002, pages 268–289. LNCS 2576, 2003.
13. D. Chaum. Security without identification: Transaction systems to make big brother obsolete. *Commun. ACM*, 28(10):1030–1044, 1985.
14. D. Chaum and E. van Heyst. Group signatures. In *EUROCRYPT 1991*, LNCS, pages 257–265. Springer, 1991.
15. S. Ding, Y. Zhao, and Y. Liu. Efficient traceable attribute-based signature. In *IEEE TRUSTCOM 2014*, pages 582–589, 2014.

16. A. El Kaafarani and E. Ghadafi. Attribute-based signatures with user-controlled linkability without random oracles. In *Cryptography and Coding*, pages 161–184, 2017.
17. A. El Kaafarani, E. Ghadafi, and D. Khader. Decentralized traceable attribute-based signatures. In *CT-RSA 2014*, pages 327–348. LNCS 8366, 2014.
18. A. Escala, J. Herranz, and P. Morillo. Revocable attribute-based signatures with adaptive security in the standard model. In *AFRICACRYPT 2011*, pages 224–241. LNCS 6737, 2011.
19. G. Fuchsbauer and D. Pointcheval. Anonymous proxy signatures. In *SCN*, volume 5229 of *LNCS*, pages 201–217. Springer, 2008.
20. M. Gagné, S. Narayan, and R. Safavi-Naini. Short Pairing-Efficient Threshold-Attribute-Based Signature. In *Pairing 2012*, pages 295–313. LNCS 7708, 2013.
21. E. Ghadafi. Stronger security notions for decentralized traceable attribute-based signatures and more efficient constructions. In *CT-RSA 2015*, pages 391–409. LNCS 9048, 2015.
22. S. Gisdakis, M. Lagana, T. Giannetsos, and P. Papadimitratos. SEROSA: service oriented security architecture for vehicular communications. In *IEEE VNC'13*, pages 111–118, 2013.
23. J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *EUROCRYPT 2008*, pages 415–432, 2008. LNCS 4965.
24. J. Guo, J. P. Baugh, and S. Wang. A group signature based secure and privacy-preserving vehicular communication framework. In *Mobile NVE 2007*, pages 103–108, 2007.
25. J. Herranz. Attribute-based Signatures from RSA. *Theor. Comput. Sci.*, 527:73–82, 2014.
26. J.-P. Hubaux, S. Čapkun, and J. Luo. The security and privacy of smart vehicles. *IEEE Security and Privacy*, 2(3):49–55, 2004.
27. P. Kamat, A. Baliga, and W. Trappe. An identity-based security framework for vanets. In *ACM VANET 2006*, pages 94–95. ACM, 2006.
28. Ł. Krzywiecki, M. Sulkowska, and F. Zagórski. Hierarchical ring signatures revisited – unconditionally and perfectly anonymous schnorr version. In *Security, Privacy, and Applied Cryptography Engineering*, pages 329–346, 2015.
29. J. Li, M. H. Au, W. Susilo, D. Xie, and K. Ren. Attribute-based Signature and Its Applications. In *ACM ASIACCS 2010*, pages 60–69. ACM, 2010.
30. H. K. Maji, M. Prabhakaran, and M. Rosulek. Attribute-based signatures. In *CT-RSA 2011*, pages 376–392, 2011.
31. G. Neven, G. Baldini, J. Camenisch, and R. Neisse. Privacy-preserving attribute-based credentials in cooperative intelligent transport systems. In *IEEE VNC'17*, pages 131–138, 2017.
32. T. Okamoto and K. Takashima. Decentralized attribute-based signatures. In *PKC 2013*, pages 125–142. LNCS 7778, 2013.
33. T. Okamoto and K. Takashima. Efficient attribute-based signatures for non-monotone predicates in the standard model. In *PKC 2011*, pages 35–52. LNCS 6571, 2011.
34. J. Petit, F. Schaub, M. Feiri, and F. Kargl. Pseudonym schemes in vehicular networks: A survey. *IEEE Comm.s Surveys and Tutorials*, 17(1):228–255, 2015.
35. R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 552–565. Springer, 2001.
36. K. Sampigethaya, M. Li, L. Huang, and R. Poovendran. AMOEBA: Robust Location Privacy Scheme for VANET. *IEEE J-SAC*, 25(8):1569–1589, 2007.

37. F. Schaub, Z. Ma, and F. Kargl. Privacy requirements in vehicular communication systems. In *CSE'09*, pages 139–145, 2009.
38. J. Sun, C. Zhang, Y. Zhang, and Y. M. Fang. An identity-based security system for user privacy in vehicular ad hoc networks. *IEEE Trans. Parallel Distrib. Syst.*, 21(9):1227–1239, 2010.
39. M. Trolin and D. Wikström. Hierarchical Group Signatures. In *ICALP 2005*, volume 3580 of *LNCS*, pages 446–458. Springer, 2005.
40. R. Tsabary. An equivalence between attribute-based signatures and homomorphic signatures, and new constructions for both. In *TCC (2)'17*, pages 489–518, 2017.

A Constructions of NIZK Proofs

Here we give an overview of the constructions for NIZK proofs $\mathtt{NIZK}_1$ and $\mathtt{NIZK}_2$ based on the instantiation introduced in Section 4.1 and refer to the literature for further details. The instantiation described requires use a public hash function $\mathcal{H}$ to map attributes into the tag space of the tagged-signature scheme.
For reference, the keys for the primitives $\mathtt{TS}$ and $\mathtt{PKE}$ are given below, respectively. All elements here belong to a group $\mathbb{G}$.

$$(ask, apk) = \big((\tilde{\alpha}, \tilde{\beta}, \gamma_Z, \delta_Z, \{\gamma_i, \delta_i\}_{i=1}^{k}),\\ (G_Z, F_Z, G_R, F_U, \{G_i, F_i\}_{i=1}^{k}, \{A_i, \tilde{A}_i, B_i, \tilde{B}_i\}_{i=0}^{1}, S_{-1}, V_{-1})\big)$$

$$(tsk, tpk) = \big((\{\alpha_i\}_{i=1}^{n}, \{\beta_i\}_{i=0}^{n+4}), (g_1, g_2, g_3, \{h_{i,1}, h_{i,2}\}_{i=1}^{n}, \{f_{i,1}, f_{i,2}\}_{i=0}^{n+4})\big)$$

Construction of $\mathtt{NIZK}_1$. To create a proof $\mathtt{NIZK}_1$, we consider each statement of the language in turn.
• For the span program $\mathbf{S} \in M_{|\Psi|,\beta}(\mathbb{F})$ representing the predicate Ψ, the signer proves they have knowledge of some $\mathbf{z}$ such that $\mathbf{zS} = [1, 0, ..., 0]$. This encodes what attributes are used to satisfy the predicate. We commit to the vector $\mathbf{z}$ and show:

$$\textstyle\sum_{i=1}^{|\Psi|}(z_i S_{i,1}) = 1 \qquad \text{and} \qquad \sum_{i=1}^{|\Psi|}(z_i S_{i,j}) = 0 \text{ for } j = 2, ..., \beta.$$

These are β linear equations in $\mathbb{Z}_p$ and in the symmetric setting, the Groth-Sahai proof consists of 2β elements in $\mathbb{Z}_p$.
• Show that for every attribute used to satisfy the predicate, we have a valid delegation path. That is, we prove:
$z_i \neq 0 \implies \forall apk_{i_j} \in \mathbf{warr}[a_i].\ \mathtt{TS.Verify}(apk_{i_j}, \mathbf{warr}[a_i][j], a_i, (apk_{i_1}, \ldots, apk_{i_j}))$.
We achieve this by raising each pairing in the verification equations for the signatures in the delegation path of a_i to $\mathbf{z}_i$. Then, if $\mathbf{z}_i \neq 0$, $\mathbf{warr}$[a] is only a valid delegation path for a if each $\mathtt{TS}$ signature verifies. If $\mathbf{z}_i = 0$ then the verification equation will trivially verify since each pairing will evaluate to $1 \in \mathbb{G}_T$.

The unbounded-message scheme we use for our instantiation amounts to chaining together an arbitrary number of constant size signatures of a scheme also given in [1]. To realise this, the message is split into blocks. The intuition is to input the n^{th} signature as part of the message of signature $n + 1$, which the structure preserving property ensures is possible. Verification requires verifying each signature in the chain w.r.t the message block, the verification key and the previous signature. In the instantiated scheme, it is actually sufficient to only sign on part of the previous signature, due to the signature binding property as detailed in original work [1]. Hence, we only need to consider how to construct the GS proof for the constant sized scheme, as the unbounded-message version amounts to verifying multiple constant-sized signatures. With this, we consider the verification equation for the constant sized signature.

The messages $m_l \in \mathbb{G}$ are the group elements of the public keys apk_j and the attribute $\mathcal{H}(a_i)$ split into message blocks w of size $k-2$. A constant sized signature is created for each message block where S_w and V_w are part of the signature for message block $w-1$, with the exception of S_{-1} and V_{-1} which are part of the verification key apk_j.

A `TS` signature verifies if it satisfies the following equations.

$$\left(\bar{G}_l = G_l^{z_i}\right)_{l=1}^{k} \quad \bar{A}_0 = A_0^{z_i} \quad \bar{A}_1 = A_1^{z_i} \quad \bar{G}_R = G_R^{z_i} \quad \bar{G}_Z = G_Z^{z_i} \quad \bar{S} = S^{z_i}$$
$$e(\bar{A}_0, \tilde{A}_0)e(\bar{A}_1, \tilde{A}_1) =$$
$$e(\bar{G}_Z, Z)e(\bar{G}_R, R)e(\bar{S}, T)e(\bar{G}_k, S_{n-1})e(\bar{G}_{k-1}, V_{n-1}) \prod_{l=1}^{k-2} e(\bar{G}_l, m_l)$$

$$\left(\bar{F}_l = F_l^{z_i}\right)_{l=1}^{k} \quad \bar{B}_0 = B_0^{z_i} \quad \bar{B}_1 = B_1^{z_i} \quad \bar{F}_Z = F_Z^{z_i} \quad \bar{F}_U = F_U^{z_i} \quad \bar{V} = V^{z_i}$$
$$e(\bar{B}_0, \tilde{B}_0)e(\bar{B}_1, \tilde{B}_1) =$$
$$e(\bar{F}_Z, Z)e(\bar{F}_U, U)e(\bar{V}, W)e(\bar{F}_k, S_{n-1})e(\bar{F}_{k-1}, V_{n-1}) \prod_{l=1}^{k-2} e(\bar{F}_l, m_l)$$

Each constant sized signature consists of 7 elements from $\mathbb{G}$, and so the size of the signature on an unbounded message is $7 \cdot \lceil \frac{n+1}{k-2} \rceil$, where k is the size of the message space and n is the number of message blocks. In the case that k does not divide the number of messages to be signed, we append $1_{\mathbb{G}}$ to complete the message in the final block. This does not increase the size of the signature. To evaluate the verification equations in zero-knowledge, we are required to prove $2k+10$ linear multi-scalar multiplication equations and 2 product pairing equations for each message block.

We prove the statement

$$\text{“}\texttt{TS.Verify}(apk_{i_n}, \sigma_u, a_i, (apk_0, apk_{i_1}, \ldots, apk_{i_n}, upk, \star)\text{”}$$

in much the same way, with upk taking the role of the final public key in the delegation path. We choose $\star$ to be some predefined fixed element from $\mathbb{G}$.

• Prove that the ciphertext was constructed correctly. We let the plaintexts $t_i \in \mathbb{G}$ for $i = 1, ..., n$ be the group elements that comprise the public keys and signatures (of `TS` and `DS`) in the warrant, and the public key of the one-time signature $otsvk$. The randomness is given by μ and μ' (in $\mathbb{Z}_p$) and is part of the witness. To prove the correctness of the ciphertext $(C, c_1, ..., c_n, u_1, u_2, u_3)$, we show the following relation holds [10].

$$u_1 = g_1^{\mu}, \ u_2 = g_2^{\mu'}, \ u_3 = g_3^{\mu+\mu'}$$
$$c_i = t_i \cdot h_{i,1}^{\mu} h_{i,2}^{\mu'}, \text{ for } i = 1, ..., n$$
$$C = \prod_{i=0}^{3} \hat{e}(f_{i,1}^{\mu} f_{i,2}^{\mu'}, u_i) \cdot \prod_{i=4}^{n+3} \hat{e}(f_{i,1}^{\mu} f_{i,2}^{\mu'}, c_{i-3})$$

For a warrant containing n group elements, this proof requires n+3 linear MSME and one PPE, hence requires $2n + 15$ elements from $\mathbb{G}$. Note n used here is equal to $n + 8$ from the efficiency discussion in Section 4.1.
• Prove `DS` is constructed correctly. Since we instantiate `DS` with the same scheme as `TS`, and hence we follow the same proof structure as above but with $k = 2$, i.e. there are only 2 messages which are comprised of the component parts of the one-time public key *otsvk*.

In addition to the proof, each GS commitment is in $\mathbb{G}^3$, the `PKE` ciphertext consists of n+4 elements from $\mathbb{G}$ while the `OTS` (including verification key) consists of 3 elements from $\mathbb{G}$ and one from $\mathbb{Z}_p$.

Construction of $\texttt{NIZK}_2$. Here we prove the following relation:

$$\texttt{PKE.Dec}(tsk, \mathrm{C}) = (upk, \mathbf{warr}, \sigma_s, otsvk).$$

For the instantiation of `PKE` above, this is achieved by showing equality of the following equations. For the ciphertext $(C, c_1..., c_n, u_1, u_2, u_3)$:

$$C = \prod_{i=0}^{3} \hat{e}(u_1^{\beta_{i,1}} u_2^{\beta_{i,2}} u_3^{\beta_{i,3}}, u_i) \cdot \prod_{i=4}^{n+3} \hat{e}(u_1^{\beta_{i,1}} u_2^{\beta_{i,2}} u_3^{\beta_{i,3}}, c_{i-3})$$
$$t_i = c_i \cdot (u_1^{\alpha_{i,1}} u_2^{\alpha_{i,2}} u_3^{\alpha_{i,3}}) \text{ for } i = 1, ..., n.$$

Computing $\texttt{NIZK}_2$ consists of n linear MSME and one PPE, which requires $2n+9$ elements from $\mathbb{G}$ where n is the number of group elements in *upk*, **warr** and σ_s. Again, each GS commitment is in $\mathbb{G}^3$.