



Hierarchical Attribute-Based Signatures

Constantin-Cătălin Drăgan, Daniel Gardham^(✉), and Mark Manulis

Surrey Centre for Cyber Security, University of Surrey, Guildford, UK
{c.dragan,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 [31, 32], 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 [32] enjoys an expressive policy, [31] 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. [21, 26, 31]. 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. [4, 17–19, 22, 34, 35], being proposed in the standard model based on bilinear maps and Groth-Sahai proofs [24]. A notable exception is the scheme in [26], 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 [15, 18, 22].

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. [18, 22, 32, 34], 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 (see also the full version in [16]) 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. [32]) 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 [39] 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 [23,27], group signatures [25,38] or identity-based signatures [29,40] have all their limitations with regard to scalability and/or limited expressivity (cf. survey in [36]). As noted in [33] an attribute-based approach would bring substantial benefits and while [33] 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 [32], group signatures [14] and ring signatures [37] 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 [41] and ring [30] 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 [32], anonymous credentials (AC) [12,13], 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 the construction of a non-delegatable AC scheme built from (homomorphic) ABS [28], where multiple root authorities issue attribute credentials directly to the signers. However, to combine attributes obtained from distinct authorities requires online collaboration of these authorities. In contrast, our scheme supports delegation and non-interactive combination of attributes

obtained from multiple authorities. Additionally, anonymous proxy signatures [20] 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 [2, 8] 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 signing 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 [42]. 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.

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

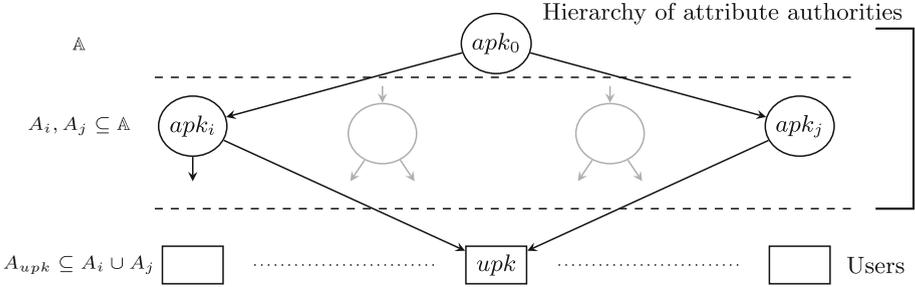


Fig. 1. Example of a HABS hierarchy where a user with public key upk receives its set of attributes $A_{upk} \subseteq A_i \cup A_j$ from two IAs i and j , who in turn receive their attribute sets A_i and A_j from the RA.

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 $a \in A$ (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 $warr[a]$, for $a \in warr$, 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 $|warr|$ to denote the total number of attributes in $warr$ and use $|warr[a]|$ 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). $\text{HABS} := (\text{Setup}, \text{KGen}, \text{AttIssue}, \text{Sign}, \text{Verify}, \text{Trace}, \text{Judge})$ consists of the following seven processes:

- $\text{Setup}(1^\lambda)$ 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 .
- $\text{KGen}(\text{pp})$ 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) .
- $\text{AttIssue}(ask_i, \mathbf{warr}_i, A, \{apk_j | upk_j\})$ is an algorithm that is used to delegate attributes to an authority with apk_j or issue them to the user with upk . On input of an authority's secret key ask_i , $i \in \mathbb{N}_0$, its warrant \mathbf{warr}_i , a subset of attributes A from \mathbf{warr}_i , and the public key of the entity to which attributes are delegated or issued, it outputs a new warrant \mathbf{warr} for that entity.
- $\text{Sign}((usk, \mathbf{warr}), m, \Psi)$ is the signing algorithm. On input of the signer's usk and (possibly reduced) \mathbf{warr} , a message m and a predicate Ψ it outputs a signature σ .
- $\text{Verify}(apk_0, (m, \Psi, \sigma))$ 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.
- $\text{Trace}(tsk, apk_0, (m, \Psi, \sigma))$ is an algorithm executed by the TA on input of its private key tsk and outputs either a triple $(upk, \mathbf{warr}, \hat{\pi})$ if the tracing is successful or \perp to indicate its failure. Note that \mathbf{warr} contains attributes and delegation paths that were used by the signer.
- $\text{Judge}(tpk, apk_0, (m, \Psi, \sigma), (upk, \mathbf{warr}, \hat{\pi}))$ is a deterministic algorithm that checks a candidate triple $(upk, \mathbf{warr}, \hat{\pi})$ 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.

$\overline{O_{\text{Reg}}(\cdot) \text{ with } (\cdot) = (i) \text{ and } i \notin HU}$ <pre style="margin: 0;"> 1 : $(sk_i, pk_i) \leftarrow \text{KGen}(\text{pp})$ 2 : $List \leftarrow List \cup \{(i, pk_i, sk_i)\}$ 3 : $HU \leftarrow HU \cup \{i\}$ 4 : return pk_i </pre>	$\overline{O_{\text{Att}}(\cdot)}$ <pre style="margin: 0;"> $(\cdot) = (i, \text{warr}_{i,a}, \{apk_j upk_j\})$ 1 : $L := \{a (a, pk_a, sk_a) \in List\}$ 2 : if $i \in L \wedge j \in L$ then 3 : $\text{warr} \leftarrow \text{AttIssue}(ask_i,$ $\text{warr}_{i,a}, \{apk_j upk_j\})$ 4 : return warr 5 : return \perp </pre>
$\overline{O_{\text{Corr}}(\cdot) \text{ with } (\cdot) = (i)}$ <pre style="margin: 0;"> 1 : if $i \in HU$ then 2 : $HU \leftarrow HU - \{i\}$ 3 : return sk_i from $List$ </pre>	$\overline{O_{\text{Sig}}(\cdot) \text{ with } (\cdot) = (i, \text{warr}, m, \Psi)}$ <pre style="margin: 0;"> 1 : $A \leftarrow \{a a \in \text{warr}\}$ 2 : if $i \in HU \wedge \Psi(A)$ then 3 : $\sigma \leftarrow \text{Sign}((usk_i, \text{warr}), m, \Psi)$ 4 : return σ 5 : return \perp </pre>
$\overline{O_{\text{Tr}}(\cdot) \text{ with } (\cdot) = (m, \Psi, \sigma)}$ <pre style="margin: 0;"> 1 : return $\text{Trace}(tsk, apk_0, (m, \Psi, \sigma))$ </pre>	

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

We note that our definitions extend earlier definitions for (multi-authority) ABS schemes, e.g. [4, 18, 22, 32], 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 [20] 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_{Att} : \mathcal{A} can ask an authority to either delegate attributes for another IA or to issue attributes to a 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} .

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

Fig. 3. Path-anonymity experiment

- O_{Sig} : \mathcal{A} can ask a signer to produce a HABS signature using the input warrant \mathbf{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 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 $\mathbf{Exp}_{\text{HABS}, \mathcal{A}}^{\text{pa-}0}$ and $\mathbf{Exp}_{\text{HABS}, \mathcal{A}}^{\text{pa-}1}$ defined in Fig. 3, i.e., the following advantage is negligible in λ :*

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

$\mathbf{Exp}_{\text{HABS}, \mathcal{A}}^{\text{nf}}(\lambda)$
1 : $(\text{pp}, \text{ask}_0, \text{tsk}) \leftarrow \text{Setup}(1^\lambda)$
2 : $((\sigma, m, \Psi), (\text{upk}_j, \mathbf{warr}, \hat{\pi})) \leftarrow \mathcal{A}(\text{pp}, \text{ask}_0, \text{tsk} : O_{\text{Att}}, O_{\text{Sig}}, O_{\text{Corr}}, O_{\text{Reg}})$
3 : if $\text{Verify}(\text{apk}_0, (m, \Psi, \sigma)) \wedge \text{Judge}(\text{tpk}, \text{apk}_0, (m, \Psi, \sigma), (\text{upk}_j, \mathbf{warr}, \hat{\pi}))$ then
4 : if $j \in \text{HU} \wedge \mathcal{A}$ did not query $O_{\text{Sig}}((\text{usk}_j, \mathbf{warr}), m, \Psi)$ then , return 1
5 : if $\exists a. a \in \mathbf{warr} \implies (\text{apk}_0, \text{apk}_1, \dots, \text{apk}_n, \text{upk}_j, \star) = \mathbf{warr}[a] \wedge$
6 : $((\exists i \in [0, n-1]. \mathcal{A}$ did not query $O_{\text{Att}}(i, \cdot, a, \text{apk}_{i+1}) \wedge i \in \text{HU}) \vee$
7 : $(\mathcal{A}$ did not query $O_{\text{Att}}(n, \cdot, a, \text{upk}_j) \wedge n \in \text{HU})$ then , return 1
8 : return 0

Fig. 4. Non-frameability experiment

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 $\mathbf{Exp}_{\text{HABS}, \mathcal{A}}^{\text{nf}}$ defined in Fig. 4, i.e., the following advantage is negligible in λ :

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

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 [4, 32].

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 \mathbf{warr} that contains at least one “rogue” entity

$\mathbf{Exp}_{\text{HABS}, \mathcal{A}}^{\text{tr}}(\lambda)$
1 : $(\text{pp}, \text{ask}_0, \text{tsk}) \leftarrow \text{Setup}(1^\lambda)$
2 : $((\sigma, m, \Psi), (\text{upk}, \mathbf{warr}, \hat{\pi})) \leftarrow \mathcal{A}(\text{pp}, \text{tsk} : O_{\text{Att}}, O_{\text{Corr}}, O_{\text{Reg}})$
3 : if $\text{Verify}(\text{apk}_0, (m, \Psi, \sigma))$ then
4 : if $\text{Trace}(\text{tsk}, (m, \Psi, \sigma)) = \perp$ then, return 1
5 : if $\text{Judge}(\text{tpk}, \text{apk}_0, (m, \Psi, \sigma), (\text{upk}, \mathbf{warr}, \hat{\pi})) \wedge$
6 : $(\exists a. a \in \mathbf{warr} \implies (\text{apk}_0, \text{apk}_1, \dots, \text{apk}_n, \text{upk}, \star) = \mathbf{warr}[a] \wedge$
7 : $(\exists i \in [0, n-1]. i \in \text{HU} \wedge (i+1, \text{apk}_{i+1}, \text{ask}_{i+1}) \notin \text{List}) \vee$
8 : $(n \in \text{HU} \wedge (\cdot, \text{upk}, \text{usk}) \notin \text{List})$) then, return 1
9 : return 0

Fig. 5. Path-traceability experiment.

(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.

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

$$\text{Adv}_{\text{HABS}, \mathcal{A}}^{\text{tr}}(\lambda) = \Pr [\mathbf{Exp}_{\text{HABS}, \mathcal{A}}^{\text{tr}}(\lambda) = 1] .$$

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* $\text{PKE} := (\text{KGen}, \text{Enc}, \text{Dec})$ [10] and an unforgeable *digital signature* $\text{DS} := (\text{KGen}, \text{Sign}, \text{Verify})$ [1] that withstands chosen-message attacks. We rely further on an unforgeable *tagged signature* $\text{TS} := (\text{KGen}, \text{Sign}, \text{Verify})$ [1] that can sign blocks of messages, also used in [18], 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 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* $\text{OTS} := (\text{KGen}, \text{Sign}, \text{Verify})$ [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* $\text{NIZK} = (\text{Setup}, \text{Prove}, \text{Verify}, \text{SimSetup}, \text{Sim})$ [5, 24] 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

We use the above general building blocks to construct our 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 $(\text{ask}_i, \text{apk}_i)$, $i \in \mathbb{N}_0$ for the tagged signature scheme TS. The TA holds a key pair (tsk, tpk) for the public key encryption scheme PKE. The public parameters pp_{of} 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 TS scheme whereas delegation paths $\text{att}L := (\text{apk}_0, \dots, \{\text{apk}_i, \{\text{upk}_{i+1}, \star\}\})$ are treated as messages. In order to create a warrant \mathbf{warr} for some authority or signer, the corresponding IA with its ask_i will produce a TS signature on each attribute a and its delegation path and include this signature into $\mathbf{warr}[a]$ as part of the list $\text{sig}L$. Thus, a separate TS signature is used for each attribute and its path such that the signer can later reduce its \mathbf{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 \mathbf{warr} that satisfies Ψ can generate a HABS signature σ for some message m . The reduced \mathbf{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 $(\text{otssk}, \text{otsvk})$ 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 \mathbf{warr} by computing a vector of integers \mathbf{z} such that $\mathbf{zS} = [1, 0, \dots, 0]$ and for any $z_i \neq 0$ we have $\rho(i) \in \mathbf{warr}$.

Then, the signer computes a NIZK₁ proof π for the statement $(C, \text{otsvk}, \text{tpk}, \text{apk}_0, \Psi)$ using as witness the previously computed $(\text{upk}, \mu, \mathbf{z}, \mathbf{warr}, \sigma_s)$ such that the following relation is satisfied:

$\text{AttIssue}(ask_i, \text{warr}_i, A, \{apk_j upk_j\})$ <hr/> <pre style="margin: 0; padding: 0;"> 1: warr := \emptyset 2: for $a \in A$ do 3: $\sigma_a \leftarrow \text{TS.Sign}(ask_i, a, (apk_1, \dots, apk_i, \{apk_j \{upk_j, \star\}\}))$ 4: $\text{warr}[a] \leftarrow \text{warr}_i[a] \cup \{apk_j \{upk_j, \star\}\} \cup \{\sigma_a\}$ 5: return warr </pre>
$\text{Sign}((usk, \text{warr}), m, \Psi)$ <hr/> <pre style="margin: 0; padding: 0;"> 1: $(otsvk, otssk) \leftarrow \text{OTS.KGen}(1^\lambda)$ 2: $\sigma_s \leftarrow \text{DS.Sign}(usk, otsvk); C \leftarrow \text{PKE.Enc}(tpk, (upk, \text{warr}, \sigma_s, otsvk); \mu)$ 3: compute \mathbf{z} s.t. $\mathbf{zS} = [1, 0, \dots, 0]$ 4: $\pi \leftarrow \text{NIZK}_1.\text{Prove}((upk, \mu, \mathbf{z}, \text{warr}, \sigma_s) : (C, otsvk, tpk, apk_0, \Psi))$ 5: $\sigma_o \leftarrow \text{OTS.Sign}(otssk, (m, \Psi, C, \pi))$ 6: $\sigma \leftarrow (\sigma_o, C, \pi, otsvk)$, return σ </pre>
$\text{Verify}(apk_0, (m, \Psi, \sigma)) \text{ with } \sigma = (\sigma_o, C, \pi, otsvk)$ <hr/> <pre style="margin: 0; padding: 0;"> return $\text{NIZK}_1.\text{Verify}((C, otsvk, tpk, apk_0, \Psi), \pi) \wedge \text{OTS.Verify}(otsvk, (m, \Psi, C, \pi), \sigma_o)$ </pre>
$\text{Trace}(tsk, apk_0, (m, \Psi, \sigma)) \text{ with } \sigma = (\sigma_o, C, \pi, otsvk)$ <hr/> <pre style="margin: 0; padding: 0;"> 1: if $\text{Verify}(apk_0, (\sigma, m, \Psi))$ then 2: $(upk, \text{warr}, \sigma_s, otsvk') \leftarrow \text{PKE.Dec}(tsk, C)$ 3: $\hat{\pi} \leftarrow \text{NIZK}_2.\text{Prove}(tsk : (otsvk, C, tpk, upk, \text{warr}, \sigma_s))$ 4: if $otsvk' = otsvk$ then return $(upk, \text{warr}, (\hat{\pi}, \sigma_s))$ 5: return \perp </pre>
$\text{Judge}(tpk, apk_0, (m, \Psi, \sigma), (upk, \text{warr}, (\hat{\pi}, \sigma_s))) \text{ with } \sigma = (\sigma_o, \pi, C, otsvk)$ <hr/> <pre style="margin: 0; padding: 0;"> return $\text{Verify}(apk_0, (m, \Psi, \sigma)) \wedge \text{NIZK}_2.\text{Verify}(tpk, (otsvk, C, upk, \text{warr}, \sigma_s), \hat{\pi})$ </pre>

Fig. 6. Algorithms of our general HABS scheme.

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

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 NIZK₁ proof π and the OTS signature σ_o .

The tracing algorithm, on input of a valid HABS signature $((\sigma_o, C, \pi, otsvk), m, \Psi)$ 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 with message $otsvk'$. Then, it checks whether $otsvk = otsvk'$. If true, TA outputs a NIZK₂ proof $\hat{\pi}$ for the statement $(otsvk, C, tpk, (apk_0, \mathbf{warr}, \sigma_s))$ using tsk as its witness to prove the following relation:

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

The output of TA on a valid HABS signature can be publicly judged by checking the validity of the NIZK₂ proof $\hat{\pi}$.

3.3 Security Analysis

In this section we show that our general HABS construction in Fig. 6 satisfies path-anonymity, non-frameability and path-traceability from assumptions of its underlying cryptographic building blocks.

Lemma 1. *HABS defined in Fig. 6 offers path anonymity, if NIZK₁ and NIZK₂ 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.

Game G_0 : Let this be the experiment corresponding to $\text{Exp}_{\text{HABS}, \mathcal{A}}^{\text{pa-b}}(\lambda)$ in Fig. 3, where the adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is required to distinguish between the signatures $\sigma_0 = (\sigma_o^0, C_0, \pi_0, otsvk_0)$ and $\sigma_1 = (\sigma_o^1, C_1, \pi_1, otsvk_1)$.

Game G_1 : This game is obtained from the game G_0 where the restriction “ \mathcal{A}_2 did not query $O_{\text{Tr}}(m, \Psi, \sigma_b)$ ” 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 $O_{\text{Tr}}(m, \Psi, (\sigma_o, C, \pi, otsvk))$ and $\sigma_b = (\sigma_{o,b}, C_b, \pi_b, otsvk_b)$. The games G_1 and G_0 preserve the exact same probability.

Game G_2 : We define G_2 as game G_1 except on the outputs of O_{Tr} , where we replace the NIZK₂ proof $\hat{\pi}$ with a proof $\hat{\pi}'$, provided by the simulator $\text{NIZK}_2.\text{Sim}$. Additionally, in game G_2 for NIZK₂ we replace **Setup** by **SimSetup**. These changes are done to avoid the case where \mathcal{A} may “extract” tsk from NIZK₂ proofs. Thus, for all future O_{Tr} oracle call we make use of a simulated NIZK₂ proof. These two games are indistinguishable due to the zero-knowledge of NIZK₂.

Game G_3 : Let G_3 be the game obtained from G_2 where the real NIZK₁ 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 $\text{NIZK}_1.\text{Sim}$ on the inputs $(C_b, otsvk_b, tpk, apk_0, \Psi)$. Similar to the previous step, by now for NIZK₁ we replace **Setup** by **SimSetup**.

We bound the capabilities of \mathcal{A} to distinguish between games G_3 and G_2 by the advantage of the zero-knowledge adversary for NIZK_1 .

Game G_4 : Game G_4 is identical to game G_3 , except we abort if \mathcal{A}_2 queries $O_{\text{Tr}}(m, \Psi, (\sigma_o, C, \pi, \text{otsvk}))$ if $(C, \text{otsvk}) = (C_b, \text{otsvk}_b)$. 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 otsk_b . Essentially, breaking the strong unforgeability of OTS.

Game G_5 : This game G_5 is the same as G_4 , except we abort if \mathcal{A}_2 queries $O_{\text{Tr}}(m, \Psi, (\sigma_o, C, \pi, \text{otsvk}))$ when $C = C_b$. 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 .

Game G_6 : Let G_6 be the game obtained from G_5 where the ciphertext C_b from the challenge signature $\sigma_b = (\sigma_{o,b}, C_b, \pi'_b, \text{otsvk}_b)$ is replaced with C_0 . The distinguishing capabilities of the adversary \mathcal{A}_2 are bounded by the advantage of the IND-CCA2 adversary for the PKE scheme.

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 NIZK_2 used in G_2 , \mathcal{A} does not have access to tsk . Therefore, the advantage of \mathcal{A} in winning this experiment is 0. \square

Lemma 2. *The generic HABS construction from Fig. 6 is non-frameable, if 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, E_1 , captures the probability of the adversary \mathcal{A} to create a forgery. The second experiments E_2 follows the exact same steps as E_1 except that “ $j \in HU \wedge \mathcal{A}$ did not query $O_{\text{Sig}}((usk_j, \mathbf{warr}), \Psi, m)$ ” is replaced by

$$\begin{aligned} \text{“}\exists a. a \in \mathbf{warr} \implies (apk_0, apk_1, \dots, apk_n, upk_j, \star) = \mathbf{warr}[a] \wedge \\ (\exists 0 \leq i \leq n-1. \mathcal{A} \text{ did not call } O_{\text{Att}}(i, \cdot, a, apk_{i+1}) \wedge i \in HU) \vee \\ (\mathcal{A} \text{ did not call } O_{\text{Att}}(n, \cdot, a, upk_j) \wedge n \in HU)\text{”} \end{aligned}$$

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

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

We start with the first experiment E_1 that we will show has a negligible probability of success. Intuitively, we want to argue over all the values $(upk', \mathbf{warr}', m', \Psi'), (\sigma'_o, C', \pi', \text{otsvk}')$ 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 $(\sigma_o, C, \pi, \text{otsvk})$ for the values $(upk_j, \mathbf{warr}, m, \Psi)$ 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 ”, which sets the direction for the rest of the proof. Next, we follow the

same methodology by reasoning that the values \mathbf{warr}' , m' , Ψ' and $otsvk'$ have to coincide with \mathbf{warr} , m , Ψ , $otsvk$ for \mathcal{A} to actually produce valid proofs and signatures that pass the verification conditions in E_1 .

Game G_0 . The game G_0 is defined exactly as E_1 except on line “ \mathcal{A} did not query $O_{\text{Sig}}((usk_j, \mathbf{warr}), m, \Psi)$ ” that is replaced with a membership check $(upk_j, \mathbf{warr}, m, \Psi) \notin sL$ 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.

Game G_1 . This game is defined exactly as G_0 with the exception the additional test $\text{DS.Verify}(upk_j, otsvk, \sigma_s)$ performed over the output of the adversary $(((\sigma_o, C, \pi, otsvk), m, \Psi), (upk_j, \mathbf{warr}, (\hat{\pi}, \sigma_s)))$. G_1 is indistinguishable from G_0 due to the soundness of NIZK_1 : the probability of generating a valid NIZK_1 proof for a false statement (that does not pass DS verification).

Game G_2 . This game uses the exact steps performed by game G_1 , except it returns false if the adversary has not queried at least one signature that contains user upk_j . The adversary \mathcal{A} is able to distinguish between these two games if he can produce a valid digital signature σ_s for upk_j that passes DS verification without having access to the user’s secret key (as $j \in HU$). Thus, breaking unforgeability for DS.

Game G_3 . We define game G_3 exactly as G_2 , except we replace the test on where \mathcal{A} has queried O_{Sig} for user upk_j : line $(upk_j, \star, \star, \star) \in sL$ with

$$\exists \mathbf{warr}'. m'. \Psi'. \sigma'_o. C'. \pi'. otsvk'. ((upk_j, \mathbf{warr}', m', \Psi'), (\sigma'_o, C', \pi', otsvk')) \in spL.$$

We have $(\mathbf{warr}, m, \Psi) \neq (\mathbf{warr}', m', \Psi')$, because of $(upk_j, \mathbf{warr}, m, \Psi) \notin sL$ and $(upk_j, \mathbf{warr}', m', \Psi') \in sL$.

Game G_4 . We define G_4 as the game G_3 , except we extend the condition “ $\exists \mathbf{warr}' \dots ((upk_j, \mathbf{warr}', m', \Psi'), (\sigma'_o, C', \pi', otsvk')) \in spL$ ” to include $otsvk = otsvk'$. The capabilities of the adversary \mathcal{A} to distinguish between G_4 and G_3 are bounded by the unforgeability of the DS.

Game G_5 . We define game G_5 as the game G_4 , where we add the restriction $(m, \Psi) = (m', \Psi')$, on the same line and in the same manner as the changes done in game G_4 . The adversary \mathcal{A} is able to distinguish between these two games, if he can provide a forgery for the OTS scheme by signing a message that contains (m', Ψ') without knowledge of $otssk$.

Game G_6 . We define game G_6 as the game G_5 , except we add $C \neq C'$ to the same line that was modified in game G_5 . Because of the restriction $(\mathbf{warr}, m, \Psi) \neq (\mathbf{warr}', m', \Psi')$, we have $\mathbf{warr}' \neq \mathbf{warr}$ in game G_6 . Given $\mathbf{warr} \neq \mathbf{warr}'$, we now show that $C \neq C'$. This is guaranteed by the correctness property of the encryption scheme PKE that builds C' . There should not be possible to find \mathbf{warr} such that C (equal to C') decrypts to it. We take $m_0 = (upk_j, \mathbf{warr}, \sigma_s, otsvk)$ 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.

The probability of adversary \mathcal{A} to win G_6 is bounded by the probability of creating a OTS forgery without knowledge of $otsk$, that passed the verification in the body of experiment G_6 .

From the sequence of games starting G_0, \dots, G_6 , it follows that the probability of E_1 are bounded by unforgeability of DS, strong unforgeability of OTS, and zero-knowledge of NIZK₁.

The experiment E_2 models 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 for TS. \square

Lemma 3. *The generic HABS construction from Fig. 6 offers path traceability, if NIZK₁ is sound and TS is unforgeable.*

Proof. See full version in [16].

Theorem 1. *The proposed 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, NIZK₁ and NIZK₂ are both sound zero-knowledge proofs.*

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

3.4 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 NIZK₁ and NIZK₂, we use Groth-Sahai (GS) proof systems [24], 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 committing 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 [24] for full details and give an overview of our constructions for NIZK₁ and NIZK₂ in the full version [16].

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 \mathbf{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 \mathbf{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 \mathbf{DS} verifies is of constant size and requires 72 elements of \mathbb{G} .

Finally, the size of the proof in \mathbf{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 \mathbf{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., [18, 22, 32].

3.5 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 authorities 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 \mathbf{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 \mathbf{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 NIZK_1 can enable generation of HABS signatures for predicates Ψ requiring possession of attributes from these hierarchies.

4 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. DG was supported by the UK Government PhD studentship scheme. CD and MM were supported by the EPSRC project TAPESTRY (EP/N02799X). The authors also thank the reviewers of CANS 2018 and Alfredo Rial for valuable comments.

References

1. Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structure-preserving signatures and commitments to group elements. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 209–236. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_12
2. Backes, M., Meiser, S., Schröder, D.: Delegatable functional signatures. In: Cheng, C.-M., Chung, K.-M., Persiano, G., Yang, B.-Y. (eds.) PKC 2016. LNCS, vol. 9614, pp. 357–386. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49384-7_14

3. Belenkiy, M., Camenisch, J., Chase, M., Kohlweiss, M., Lysyanskaya, A., Shacham, H.: Randomizable proofs and delegatable anonymous credentials. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 108–125. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03356-8_7
4. Bellare, M., Fuchsbauer, G.: Policy-based signatures. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 520–537. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54631-0_30
5. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications. In: STOC 1988, pp. 103–112 (1988)
6. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_4
7. Boyen, X.: Mesh signatures. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 210–227. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72540-4_12
8. Boyle, E., Goldwasser, S., Ivan, I.: Functional signatures and pseudorandom functions. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 501–519. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54631-0_29
9. Camenisch, J., Drijvers, M., Dubovitskaya, M.: Practical UC-secure delegatable credentials with attributes and their application to blockchain. In: ACMCCS 2017, pp. 683–699 (2017)
10. Camenisch, J., Haralambiev, K., Kohlweiss, M., Lapon, J., Naessens, V.: Structure preserving CCA secure encryption and its application to oblivious third parties. Cryptology ePrint Archive, Report 2011/319 (2011)
11. Camenisch, J., Krontiris, L., Lehmann, A., Neven, G., Paquin, C., Rannenber, K., Zwingelberg, H.: H2.1 abc4trust architecture for developers (2011). abc4trust.eu
12. Camenisch, J., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Cimato, S., Persiano, G., Galdi, C. (eds.) SCN 2002. LNCS, vol. 2576, pp. 268–289. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36413-7_20
13. Chaum, D.: Security without identification: transaction systems to make big brother obsolete. *Commun. ACM* **28**(10), 1030–1044 (1985)
14. Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991). https://doi.org/10.1007/3-540-46416-6_22
15. Ding, S., Zhao, Y., Liu, Y.: Efficient traceable attribute-based signature. In: IEEE TRUSTCOM 2014, pp. 582–589 (2014)
16. Dragan, C.-C., Gardham, D., Manulis, M.: Hierarchical attribute-based signatures. *IACR Cryptology ePrint Archive* (2018). <https://eprint.iacr.org/2018/610>
17. El Kaafarani, A., Ghadafi, E.: Attribute-based signatures with user-controlled linkability without random oracles. In: O’Neill, M. (ed.) IMACC 2017. LNCS, vol. 10655, pp. 161–184. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-71045-7_9
18. El Kaafarani, A., Ghadafi, E., Khader, D.: Decentralized traceable attribute-based signatures. In: Benaloh, J. (ed.) CT-RSA 2014. LNCS, vol. 8366, pp. 327–348. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-04852-9_17
19. Escala, A., Herranz, J., Morillo, P.: Revocable attribute-based signatures with adaptive security in the standard model. In: Nitaj, A., Pointcheval, D. (eds.) AFRICACRYPT 2011. LNCS, vol. 6737, pp. 224–241. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21969-6_14

20. Fuchsbauer, G., Pointcheval, D.: Anonymous proxy signatures. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 201–217. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85855-3_14
21. Gagné, M., Narayan, S., Safavi-Naini, R.: Short pairing-efficient threshold-attribute-based signature. In: Abdalla, M., Lange, T. (eds.) Pairing 2012. LNCS, vol. 7708, pp. 295–313. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36334-4_19
22. Ghadafi, E.: Stronger security notions for decentralized traceable attribute-based signatures and more efficient constructions. In: Nyberg, K. (ed.) CT-RSA 2015. LNCS, vol. 9048, pp. 391–409. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-16715-2_21
23. Gisdakis, S., Lagana, M., Giannetsos, T., Papadimitratos, P.: SEROSA: service oriented security architecture for vehicular communications. In: IEEE VNC 2013, pp. 111–118 (2013)
24. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_24
25. Guo, J., Baugh, J.P., Wang, S.: A group signature based secure and privacy-preserving vehicular communication framework. In: Mobile NVE 2007, pp. 103–108 (2007)
26. Herranz, J.: Attribute-based signatures from RSA. TCS **527**, 73–82 (2014)
27. Hubaux, J.-P., Čapkun, S., Luo, J.: The security and privacy of smart vehicles. IEEE Secur. Priv. **2**(3), 49–55 (2004)
28. Kaaniche, N., Laurent, M., Rocher, P.-O., Kiennert, C., Garcia-Alfaro, J.: PCS, A privacy-preserving certification scheme. In: Garcia-Alfaro, J., Navarro-Arribas, G., Hartenstein, H., Herrera-Joancomartí, J. (eds.) ESORICS/DPM/CBT -2017. LNCS, vol. 10436, pp. 239–256. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67816-0_14
29. Kamat, P., Baliga, A., Trappe, W.: An identity-based security framework for vanets. In: ACM VANET 2006, pp. 94–95. ACM (2006)
30. Krzywiecki, L., Sulkowska, M., Zagórski, F.: Hierarchical ring signatures revisited – unconditionally and perfectly anonymous schnorr version. In: Chakraborty, R.S., Schwabe, P., Solworth, J. (eds.) SPACE 2015. LNCS, vol. 9354, pp. 329–346. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24126-5_19
31. Li, J., Au, M.H., Susilo, W., Xie, D., Ren, K.: Attribute-based signature and its applications. In: ACM ASIACCS 2010, pp. 60–69. ACM (2010)
32. Maji, H.K., Prabhakaran, M., Rosulek, M.: Attribute-based signatures. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 376–392. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19074-2_24
33. Neven, G., Baldini, G., Camenisch, J., Neisse, R.: Privacy-preserving attribute-based credentials in cooperative intelligent transport systems. In: IEEE VNC 2017, pp. 131–138 (2017)
34. Okamoto, T., Takashima, K.: Decentralized attribute-based signatures. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 125–142. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36362-7_9
35. Okamoto, T., Takashima, K.: Efficient attribute-based signatures for non-monotone predicates in the standard model. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 35–52. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19379-8_3
36. Petit, J., Schaub, F., Feiri, M., Kargl, F.: Pseudonym schemes in vehicular networks: a survey. IEEE Commun. Surv. Tutor. **17**(1), 228–255 (2015)

37. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45682-1_32
38. Sampigethaya, K., Li, M., Huang, L., Poovendran, R.: AMOEBA: robust location privacy scheme for VANET. IEEE J.-SAC **25**(8), 1569–1589 (2007)
39. Schaub, F., Ma, Z., Kargl, F.: Privacy requirements in vehicular communication systems. In: CSE 2009, pp. 139–145 (2009)
40. Sun, J., Zhang, C., Zhang, Y., Fang, Y.M.: An identity-based security system for user privacy in vehicular ad hoc networks. IEEE Trans. Parallel Distrib. Syst. **21**(9), 1227–1239 (2010)
41. Trolin, M., Wikström, D.: Hierarchical Group Signatures. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 446–458. Springer, Heidelberg (2005). https://doi.org/10.1007/11523468_37
42. Tsabary, R.: An equivalence between attribute-based signatures and homomorphic signatures, and new constructions for both. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10678, pp. 489–518. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70503-3_16