Revocable Hierarchical Attribute-based Signatures from Lattices

Daniel Gardham[⋆]

Surrey Centre for Cyber Security
University of Surrey
Guildford, United Kingdom
daniel.gardham@surrey.ac.uk

Mark Manulis

Research Institute CODE
Universität der Bundeswehr München
Munich, Germany
mark@manulis.eu

Abstract. Attribute-based Signatures (ABS) allow users to obtain attributes from issuing authorities, and sign messages whilst simultaneously proving compliance of their attributes with a verification policy. ABS demands that both the signer and the set of attributes used to satisfy a policy remain hidden to the verifier. Hierarchical ABS (HABS) supporting roots of trust and delegation were recently proposed to alleviate scalability issues in centralised ABS schemes.

An important yet challenging property for privacy-preserving ABS is revocation, which may be applied to signers or some of the attributes they possess. Existing ABS schemes lack efficient revocation of either signers or their attributes, relying on generic costly proofs. Moreover, in HABS there is a further need to support revocation of authorities on the delegation paths, which is not provided by existing HABS constructions.

This paper proposes a direct HABS scheme with a Verifier-Local Revocation (VLR) property. We extend the original HABS security model to address revocation and develop a new attribute delegation technique with appropriate VLR mechanism for HABS, which also implies the *first* ABS scheme to support VLR. Moreover, our scheme supports inner-product signing policies, offering a wider class of attribute relations than previous HABS schemes, and is the first to be based on lattices, which are thought to offer post-quantum security.

Keywords: Attribute-based Signatures, Revocation, Privacy

1 Introduction

(Hierarchical) Attribute-based Signatures. To provide privacy-preserving authentication, Attribute-based Signatures (ABS), introduced in [31,40], allow users to collect attributes from authorities and produce signatures showing attribute-compliance with some signing policy. A core security property of ABS schemes is that they are attribute-hiding, and for schemes that consider multiple users, it is often required that they also remain anonymous. A second security property, unforgeability, prevents users from generating signatures for policies for which they do not have a satisfying set of attributes.

Most constructions for ABS schemes [20,45,46,6,18,23,17,51] are based on bilinear groups and make use of the flexible Groth-Sahai proof system [25] to provide anonymity guarantees. Notable exceptions include constructions from RSA [26] and recent work in the lattice setting [19,54,57,56], which are in the random oracle model. Originally, ABS schemes were proposed in the centralised model, that is, one central authority is responsible for all attribute issuance, but to allow for larger scalability, decentralised schemes [18,23] have also been developed.

More recently, Hierarchical Attribute-Based Signatures (HABS) [15,21] overcome the shortcomings of previous schemes by allowing attribute delegation to intermediate authorities. In particular, a central *Root Authority (RA)* delegates issuing rights of a subset of attributes to lower tier *Intermediate Authorities (IA)* who can delegate further, or issue directly to a user. This overcomes the bottleneck of requiring a single authority to issue all attributes in a scheme with either a large number of users or attributes, and also allows a verifier to trust a signature without having to trust each authority in the scheme, as is the case in *decentralised* constructions.

Revocation. A desirable property of any privacy-preserving signature is the support for user revocation. This would enable a trusted authority to prevent users from producing signatures that

[⋆] Some of this work was carried out whilst the author as at the Information Security Group, Royal Holloway, University of London, UK.

pass verification, without compromising the anonymity of honest participants. Revocation for a hierarchical structure of authorities would require the ability to check that a revoked authority does not appear anywhere in the delegation path of an attribute. This brings new challenges and any HABS construction would have to perform these additional checks when verifying the HABS signature. Specific to attribute-based protocols, it may also be desirable to revoke an attribute itself, rather than issuing authorities. For example, this maybe be required in the setting where attributes may depend on the time period or changed dynamically.

Revocation techniques typically follow one of few approaches. Firstly, it can be achieved by requiring signers to update their secret credentials in order to produce a valid signature. Another approach is to use a public revocation list, which is updated with some information about revoked users. When a signature is formed, the signer typically proves in zero-knowledge that its information does not appear in the list. Finally, we have *verifier-local* revocation which puts the onus on the verifier to check that signatures have not been generated by a revoked signer. This approach still requires up-to-date revocation information but has more semblance to traditional public key infrastructure that typically use Certificate Revocation Lists, and can allow for more efficient constructions as it bypasses the need for costly zero-knowledge proofs when generating signatures. Previously, VLR as a means of revocation has appeared in group signatures (introduced in [8]) but it remains an open problem for an ABS scheme to support *any* revocation technique[1]. We note that Herranz [26] proposed a scheme called Revocable Attribute-based Signatures, where revocation refers to the revocation of anonymity, which in this, and many other ABS works, is called traceability.

Contribution. In this paper we improve upon security and functionality of existing HABS constructions by proposing a lattice-based scheme which supports revocation and a wider range of signing policies. Our scheme is based on the widely used LWE and SIS assumptions over integer lattices, and supports inner-product relations which allow for conjunctive, disjunctive and threshold policies as well as polynomial evaluations of attributes [27]. Revocation in our HABS schemes uses a novel VLR mechanism to revoke signers and attributes as well as intermediate authorities. We model HABS security and use an integration of techniques from identity-based encryption, trapdoor delegation and signature schemes as well as novel techniques to realise our construction. This work also implies the *first* lattice-based (non-hierarchical) ABS scheme with the aforementioned properties.

Related Work. In this section we review related works on VLR, lattice-based signatures and signing policies in ABS schemes.

Revocation. VLR was first suggested in [3] and formalised in [7] and has been widely researched since then, for example, improving efficiency (e.g. [58]), functionality (e.g. [14]), stronger security properties (e.g. [44,9]) or basing on different hardness assumptions (e.g. lattices [30], bilinear groups [58,44]). The first scheme secure in the standard model that supported VLR was a group signature scheme by Libert and Vergnaud [32], based on the DLIN and variants of Diffie-Hellman type assumptions. In the recent lattice-based VLR group signature scheme from Langlois et al. [30], signing requires knowledge of a secret revocation token. We note that this technique cannot be transferred to the HABS setting as a signature must also include tokens for intermediate authorities, which are part of the secret, thus a new approach is needed.

Post-Quantum Security. Most ABS schemes are based on bilinear groups [20,45,46,6,18,23,17,51], or RSA [26] and do not offer post-quantum security. As lattice-based hardness assumptions are believed to be resistant to quantum adversaries, and also have provable security under worst-case hardness assumptions, this area has attracted significant research interest. As a result, there have been many privacy-preserving signature schemes constructed, such as lattice-based group signature schemes (e.g.[24,30,35,36,37]), ring signatures (e.g.[13,5]), anonymous attribute tokens [10] and even ABS schemes (e.g.[57,19,54,56]). However, whilst ABS have been proposed from lattices, current literature falls short of the delegation feature offered by HABS.

Signing Policies in (H)ABS. Constructing schemes with more expressive signing policies is an active

[1] We note here the work [53] of Su et al. that claims to propose a revocable ABS scheme, however we note that their scheme does not hide the attributes (nor takes a signing policy) so does not meet traditional definitions of ABS.

area of research for ABS, as it allows for a wider range of use-cases and offers signers more flexibility. Despite this, many schemes [23,18,40], including all known HABS constructions [15,21], utilise span programs that result in restrictive monotone boolean policies. Wang et al. [56] provide a different construction for threshold policies but benefit from shorter private key sizes over comparable schemes. There are notable exceptions that support non-monotone policies [45] and even unbounded circuits [19]. In particular, [4] offers a lattice construction for a threshold scheme in a centralised setting, and then shows how to transform this to support more expressive ($\wedge$,$\vee$)-policies. ABS from lattices supporting inner-product policies [57] have been proposed, yet without distinguishing between signers, which prevents any meaningful definition for delegation or revocation.

2 Preliminaries

We denote vectors by lower-case bold letters ($\mathbf{a}$), and use capital bold font for matrices ($\mathbf{A}$). The transpose of a matrix $\mathbf{A}$ (or vector) is denoted by $\mathbf{A}^T$, and the concatenation of matrices (or vectors) $\mathbf{A}$ and $\mathbf{B}$ by $[\mathbf{A}||\mathbf{B}]$. We use $\mathbf{I}$ to denote the identity matrix, and if we wish to be clear on the dimension then we write $\mathbf{I}_{n\times m}$, for some naturals n and m. The interval $[a,b]$ is used to denote all *integer* values x in the range $a\leqslant x\leqslant b$. Sampling a random variable $\mathbf{x}$ from a distribution $\mathcal{X}$ is written $\mathbf{x}\hookleftarrow\mathcal{X}$. The maximum number of users in the scheme is given by $N=2^d$, and we denote the security parameter by λ. The number of levels in the hierarchy is l, and denote a signing policy by Ψ, and set $\delta:=|\Psi|$, i.e. the number of attributes that form the signing policy.

Lattices. Let $n,m,q\geqslant 2$ be integers. For a matrix $\mathbf{A}\in\mathbb{Z}_q^{n\times m}$, define the m-dimensional lattice

$$\Lambda^{\perp}(\mathbf{A})=\{\mathbf{z}\in\mathbb{Z}^m\ :\ \mathbf{A}\cdot\mathbf{z}=\mathbf{0}\mod q\}\subseteq\mathbb{Z}^m.$$

For a vector $\mathbf{u}$ in the preimage of $\mathbf{A}$, define the coset $\Lambda^{\perp}_{\mathbf{u}}=\{\mathbf{z}\in\mathbb{Z}^m:\mathbf{A}\cdot\mathbf{z}=\mathbf{u}\mod q\}$.

Gaussian Distributions over Lattices. For a postive real σ, the n-dimensional Gaussian function is given as $\rho_\sigma(\mathbf{z})=\exp(-\pi||\mathbf{z}||/\sigma^2)$ for all $\mathbf{z}\in\mathbb{R}^n$. For any n-dimensional lattice Λ, define the discrete Guassian distribution over Λ as: $\mathcal{D}_{\Lambda,\sigma}(\mathbf{z})=\frac{\rho_\sigma(\mathbf{z})}{\rho_\sigma(\Lambda)}$ for all $\mathbf{z}\in\Lambda$.

Hardness Assumptions. We will introduce the LWE and SIS probelems and state their hardness assumptions.

LWE. The (Decisional) Learning With Errors ($\mathsf{LWE}\ _{n,m,q,\chi}$) problem is as follows. Let $n,m\geqslant 1$, $q\geqslant 2$ and χ be a probability distribution over $\mathbb{Z}$. Let $\mathbf{s}\in\mathbb{Z}_q^n$, then $\mathcal{D}_{\mathbf{s},\chi}$ is a distribution obtained by sampling $\mathbf{a}\leftarrow\mathbb{Z}_q^n$ and $e\leftarrow\chi$ and computing $(\mathbf{a},\mathbf{a}^T\mathbf{s}+e)\in\mathbb{Z}_q^n\times\mathbb{Z}_q$. Then the $\mathsf{LWE}_{n,m,q,\chi}$ requires an adversary to distinguish m samples chosen from χ and m samples chosen from a uniform distribution from $\mathbb{Z}_q^n\times\mathbb{Z}_q^n$. If q is a prime power, $\beta\geqslant\sqrt{n}\cdot\omega(\log n)$ and $\gamma=\mathcal{O}(nq/\beta)$ then there exists an efficient sample-able β-bounded distribution χ, usually instantiated as a discrete Gaussian $\mathcal{D}_{\mathbb{Z},\alpha}$. That is, the distribution χ outputs samples with norm at most β with overwhelming probability. Then, χ is such that the $\mathsf{LWE}_{n,m,q,\chi}$ problem is as least as hard as SIVP_γ (see [50]). We will also make use of a variant of LWE called Binary-LWE, where the domain from which $\mathbf{s}$ is sampled is restricted to $\{0,1\}^n$, which incurs an increase in parameter size by a factor of $\log n$ to maintain claimed security level [41].

SIS. The Short Integer Solution problem ($\mathsf{SIS}\ _{n,m,q,\beta}$), introduced in [1], requires an adversary who, given a uniformly matrix $\mathbf{A}\in\mathbb{Z}_q^{n\times m}$, to find a non-zero vector $\mathbf{z}\in\mathbb{Z}_q^m$ such that $||\mathbf{z}||_\infty\leqslant\beta$ and $\mathbf{Az}=\mathbf{0}$ mod q. We define the *Inhomogenous* Short Integer Solution ($\mathsf{ISIS}_{n,m,q,\beta}$) as SIS but for a non-zero syndrome, i.e. $\mathbf{Az}=\mathbf{u}\mod q$. By considering the relationship between the l_2, l_∞ norms, it is shown that $\mathsf{SIS}_{n,m,q,\beta}$ is at least as hard as SIVP_γ (in l_2) for $\gamma=\beta\cdot\mathcal{O}(\sqrt{n})$ [43].

3 VLR-HABS Model: Entities and Definitions

We start with the description of entities for the VLR-HABS ecosystem.

Attribute Authorities. The set of Attribute Authorities (AA) comprises the Root Authority (RA) and Intermediate Authorities (IAs). As the name suggests, the RA is the root of the hierarchy, and upon setup defines the universe of attributes $\mathbb{A}$. With its key pair $(\mathsf{skd}_0,\mathsf{pkd}_0)$, the RA can delegate a subset of attributes to IAs which hold their own key pairs $(\mathsf{skd}_i,\mathsf{pkd}_i)$, $i>0$. IAs can further delegate/issue attributes to other IAs or to any end user. This allows for a dynamically expandable VLR-HABS hierarchy to be established.

Users. With key pair $(\mathsf{usk},\mathsf{upk})$, a user joins the scheme by being receiving issued attributes from potentially many AAs. Then, a user can use usk to create a VLR-HABS signature, provided their issued set of attributes A satisfies the policy, i.e. $\Psi(A')=1$ for some $A'\subseteq A$ and a signing policy Ψ. Users are prevented from delegating attributes further and thus can be viewed as the lowest tier of the hierarchy. We realise this in our scheme by requiring users to obtain public keys in a different space to that of authorities.

Warrants. A warrant is used to store delegated attributes for each IA or user. It contains the attribute, the delegation information, and a list of identities that comprise the delegation path of the attribute. Warrants are updated any time a new attribute is issued by appending a new entry. We use the notation $|\mathsf{warr}|$ to denote the size of the warrant, i.e. the number of attributes stored in the warrant warr, and we use $|\mathsf{warr}\ [\mathsf{att}\ ]|$ to denote the length of the delegation path of the attribute $\mathsf{att}\in\mathbb{A}$. During the signing phase, the user submits uses a reduced warrant for an attribute set $A'\subseteq A$ that satisfies $\Psi(A')=1$. We fix the maximum depth of the delegation path to be l and stress this is not a restriction on the *minimum.*

Tracing Authority. The tracing authority (TA), independent of the hierarchy, is responsible for removing anonymity in the case of misuse. It can identify the signer and all authorities on the delegation paths for attributes that the signer used to satisfy the signing policy, and proves correctness of these identities by producing a publicly verifiable proof.

Revocation Authority. The Revocation Authority (RevA) is a trusted third party that acts independently of the hierarchy. The role of the RevA is to publish a list of revoked IDs that cause any signature generated with a corresponding revoked identity to fail verification. The RevA would require input of a user or AA identity in order to execute its function, which given the anonymity of VLR-HABS, could require extraction from a signature by the TA. In practice, it might be likely that the TA and RevA would be instantiated as a single authority whose role covers both functions, however, we present them as independent parties to cover a more general scheme.

Definition 1 (VLR-HABS). *A* $\mathsf{VLR\text{-}HABS}:=(\mathsf{Setup},\ \mathsf{UKGen},\ \mathsf{AKGen},\ \mathsf{AttIssue},\ \mathsf{Revoke},\ \mathsf{Sign},\ \mathsf{Verify},\ \mathsf{Trace},\ \mathsf{Judge})$ *consists of the following nine processes:*

- $\mathsf{Setup}(1^\lambda)$ *is the initialisation process. Based on some security parameter* $\lambda\in\mathbb{N}$*, the public parameters* pp *of the scheme are defined. In this phase, the root, tracing and revocation authorities independently generate their own key pairs, i.e. RA's* $(\mathsf{skd}_0,\mathsf{pkd}_0)$*, TA's* $(\mathsf{sk_{TA}},\mathsf{pk_{TA}})$ *and RevA's* $(\mathsf{sk_{RevA}},\mathsf{pk_{RevA}})$. *In addition, RA defines the universe of attributes* $\mathbb{A}$*, and initialises an empty list* $\mathsf{RevokeList}$. *We stress that due to dynamic hierarchy, the system can be initialised by publishing* $(\mathsf{pp},\ \mathsf{pkd}_0,\mathsf{pk_{TA}},\mathsf{pk_{RevA}})$ *with* $\mathbb{A}$ *and* $\mathsf{RevokeList}$ *contained in* pp.
- $\mathsf{UKGen}(\mathsf{pp},\mathsf{skd}_0)$ *is a key generation algorithm executed by the root authority for users and issued to users as* $(\mathsf{usk},\mathsf{upk},\mathsf{id})$.
- $\mathsf{AKGen}(\mathsf{pp})$ *is a key generation algorithm executed independently by intermediate authorities. Each IA generates its own public key, i.e.,* $\mathsf{pkd}_i,\mathsf{id}_i$ $(i>0)$.
- $\mathsf{AttIssue}(\mathsf{warr}_i,\mathsf{att},\{\mathsf{pkd}_j|\mathsf{upk}_j\})$ *is an algorithm that is used to delegate attributes to an authority* id_j *with* pkd_j *or issue them to the user* uid *with* upk. *On input of an authority's warrant* warr_i*, an attribute* att *from* warr_i*, and the public key of the entity to which attributes are delegated or issued, it outputs a new warrant* warr *for that entity.*
- $\mathsf{Revoke}(\mathsf{sk_{RevA}},\mathsf{id})$ *is an algorithm executed by the Revocation Authority. Using* $\mathsf{RevokeList}$ *from the implicit input* pp*, and on input of a User or AA ID* $(\mathsf{uid},\mathsf{id})$*, it outputs an updated* $\mathsf{RevokeList}$.
- $\mathsf{Sign}((\mathsf{usk},\mathsf{warr}),\mathsf{m},\Psi)$ *is the signing algorithm. On input of the signer's* usk *and (possibly reduced)* warr*, a message* m *and a predicate* Ψ *it outputs a signature* σ.
- $\mathsf{Verify}(\mathsf{pkd}_0,(\mathsf{m},\Psi,\sigma))$ *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 revocation list* $\mathsf{RevokeList}$ *from* pp*, and 0 otherwise.*

- $\mathsf{Trace}(\mathsf{sk}_{\mathsf{TA}},\mathsf{pkd}_0,(\mathsf{m},\Psi,\sigma))$ *is an algorithm executed by the TA on input of its private key* $\mathsf{sk}_{\mathsf{TA}}$ *and a VLR-HABS signature* σ*, it outputs either a triple* $(\mathsf{upk},\mathsf{warr},\hat{\pi})$ *if the tracing is successful or* $\perp$ *to indicate its failure. Note that* warr *contains attributes and delegation paths that were used by the signer.*
- $\mathsf{Judge}(\mathsf{pk}_{\mathsf{TA}},\mathsf{pkd}_0,(\mathsf{m},\Psi,\sigma),\ (\mathsf{upk},\mathsf{warr},\hat{\pi}))$ *is a deterministic algorithm that checks a candidate triple* $(\mathsf{upk},\mathsf{warr},\hat{\pi})$ *from the tracing algorithm and outputs* 1 *if the triple is valid and* 0 *otherwise.*

A VLR-HABS scheme satisfies the *correctness* property if any signature σ generated based on an honestly issued warrant that satisfies the signing policy, will verify and trace correctly, if and only if identities used in the warrant have not been revoked. The output $(\mathsf{upk},\mathsf{warr},\hat{\pi})$ of the tracing algorithm on such signatures will be accepted by the public judging algorithm with overwhelming probability. Formally, we have:

Definition 2 (Correctness). *A VLR-HABS scheme is correct if the following condition holds:*

$$\begin{cases} \Psi(A)=1 & (1)\\ \forall \mathsf{att}\in A, \exists \mathsf{warr}[\mathsf{att}]\in\mathsf{warr}\ \textit{s.t.}\ \mathsf{warr}[\mathsf{att}]\ \textit{is valid.} & (2)\end{cases}$$

implies

$$\mathsf{Verify}(\mathsf{pkd}_0,(\mathsf{m},\Psi,\mathsf{Sign}((\mathsf{usk},\mathsf{warr}),\mathsf{m},\Psi)))=1 \iff \forall \mathsf{id}\in\mathsf{warr},\mathsf{id}\notin\mathsf{RevokeList} \quad (3)$$

and

$$\mathsf{Judge}((\mathsf{pk}_{\mathsf{TA}},\mathsf{pkd}_0,(\mathsf{m},\Psi,\sigma),\mathsf{Trace}(\mathsf{sk}_{\mathsf{TA}},\mathsf{pkd}_0,(\mathsf{m},\Psi,\sigma)))=1 \quad (4)$$

3.1 Security Properties of VLR-HABS

Our security definitions are closely related to *path anonymity*, *path traceability*, and *non-frameability* from [15] but with modifications to allow for revocation functionality. We give new game-based definitions assuming probabilistic polynomial time (PPT) adversaries interacting with VLR-HABS entities through a set of oracles given below and formally described in Figure 1.

- O_{RegU}: $\mathcal{A}$ registers new users through this registration oracle, for which a key pair will be generated and added to List. The public key is given to the adversary. Initially, the entity is considered honest, and so the public key is also added to the list HUList.
- O_{RegA}: $\mathcal{A}$ registers new IAs through this registration oracle, for which an identity will be generated and added to AList, which is given to the adversary.
- O_{CorrU}: This oracle allows $\mathcal{A}$ to corrupt registered users. Upon input of a public key, the corresponding private key is given as output if it exists in List. The public key is removed from HUList so the oracle keeps track of corrupt entities.
- O_{CorrA}: This oracle allows $\mathcal{A}$ to corrupt registered IAs and User attribute keys. Upon input of a public key and an attribute, the corresponding private key is given as output the if the pair exists in AList. The identity is removed from HAList so the oracle keeps track of corrupt delegations.
- O_{Att}: $\mathcal{A}$ uses this oracle to invoke an attribute authority to delegate attributes to either an IA or to a user. In particular, the adversary has control over which attributes are issued and the oracle outputs a warrant warr if both parties are registered, otherwise it outputs $\perp$. The public key and attribute are added to a list HAList, that is initialised with $\{0,\perp,\perp,\perp,\mathsf{att}\}$, $\forall \mathsf{att}\in\mathbb{A}$.
- O_{Sig}: $\mathcal{A}$ uses this oracle to obtain a VLR-HABS signature from a registered user. The adversary provides the warrant (and implicitly the attributes used), signing policy, message and the public key of the signer. If the attribute set satisfies the policy, and the public key is contained in HUList then the signature will be given to $\mathcal{A}$, otherwise $\perp$ is returned.
- O_{Tr}: $\mathcal{A}$ uses the Trace oracle on a VLR-HABS signature (provided by the adversary) to extract the attributes and identities. The TA does verification checks on the signature and upon failure, will return $\perp$, otherwise it outputs the warrant warr.
- O_{RevID}: $\mathcal{A}$ uses this oracle to revoke a user. In particular, the adversary has control over which IDs (both Users and AAs) are revoked. The oracle outputs an updated revocation list $\mathsf{RevokeList}$ if the entity exists in List or AList, otherwise it outputs $\perp$.

```
$\mathcal{O}_{\mathrm{RegU}}(i),\ i \notin \mathsf{List}$
1: $(\mathsf{id},\mathsf{usk}_i,\mathsf{upk}_i) \leftarrow \mathsf{UKGen}(\mathsf{pp})$
2: $\mathsf{List} \leftarrow \mathsf{List} \cup \{(i,\mathsf{id},\mathsf{upk}_i,\mathsf{usk}_i)\}$
3: $\mathsf{HUList} \leftarrow \mathsf{HUList} \cup \{i\}$
4: return $(\mathsf{id},\mathsf{upk}_i)$
```

```
$\mathcal{O}_{\mathrm{RegA}}(i),\ i \notin \mathsf{AList}$
1: $\mathsf{id} \leftarrow \mathsf{AKGen}(\mathsf{pp})$
2: $\mathsf{AList} \leftarrow \mathsf{AList} \cup \{(i,\mathsf{id},\perp,\perp,\perp)\}$
3: return $\mathsf{id}$
```

```
$\mathcal{O}_{\mathrm{CorrU}}(i)$
1: $\mathsf{HUList} \leftarrow \mathsf{HUList} \setminus \{i\}$
2: return $\mathsf{skd}_i$ from $\mathsf{List}$
```

```
$\mathcal{O}_{\mathrm{CorrA}}(i,\mathsf{pkd}_{i,\mathsf{att}},\mathsf{att})$
1: $\mathsf{HAList} \leftarrow \mathsf{HAList} \setminus \{i,\mathsf{pkd}_{i,\mathsf{att}},\mathsf{att}\}$
2: return $\mathsf{skd}_{i,\mathsf{att}}$ from $\mathsf{AList}$
```

```
$\mathcal{O}_{\mathrm{Tr}}(\mathsf{m},\Psi,\sigma)$
1: return $\mathsf{Trace}(\mathsf{sk}_{\mathsf{TA}},\mathsf{pkd}_0,(\mathsf{m},\Psi,\sigma))$
```

```
$\mathcal{O}_{\mathrm{Att}}(i,\mathsf{warr}_i,\mathsf{att},\{\mathsf{id}_j|\mathsf{uid}_j\})$
1:  $L := \{(i,\mathsf{pkd}_{i,\mathsf{att}},\mathsf{att}) | \{i,\mathsf{id},\mathsf{pkd}_{i,\mathsf{att}},$
2:          $\mathsf{skd}_{i,\mathsf{att}},\mathsf{att}\} \in \mathsf{AList}\}$
3:  if $(i,\mathsf{warr}_i,\mathsf{att}) \in L \wedge j \in \mathsf{List} \vee \mathsf{AList}$ then
4:      $(\mathsf{skd}_{i,\mathsf{att}},\mathsf{pkd}_{i,\mathsf{att}}) \leftarrow \mathsf{AttIssue}(\mathsf{skd}_{i,\mathsf{att}},$
5:          $\mathsf{warr}_i,\mathsf{att},\{\mathsf{id}_j|\mathsf{uid}_j\})$
6:      $\mathsf{warr}_j[\mathsf{att}] \leftarrow \mathsf{warr}_i[\mathsf{att}] \cup \{\mathsf{pkd}_{i,\mathsf{att}},\mathsf{id}_j,\mathsf{att}\}$
7:      $\mathsf{AList} \leftarrow \mathsf{AList} \cup \{j,\mathsf{id}_j,\mathsf{pkd}_{j,\mathsf{att}},\mathsf{skd}_{j,\mathsf{att}},\mathsf{att}\}$
8:      $\mathsf{HAList} \leftarrow \mathsf{HAList} \cup \{j,\mathsf{pkd}_{j,\mathsf{att}},\mathsf{att}\}$
9:      return $\mathsf{warr}$
10: return $\perp$
```

```
$\mathcal{O}_{\mathrm{Sig}}(i,\mathsf{warr},\mathsf{m},\Psi)$
1: $A \leftarrow \{\mathsf{att} |\ \mathsf{att} \in \mathsf{warr}\}$
2: if $i \in \mathsf{HUList} \wedge \Psi(A)$ then
3:     $\sigma \leftarrow \mathsf{Sign}((\mathsf{usk}_i,\mathsf{warr}),\mathsf{m},\Psi)$
4:     return $\sigma$
5: return $\perp$
```

```
$\mathcal{O}_{\mathrm{RevID}}(i,\mathsf{id},\mathsf{RevokeList})$
1: if $(i,\mathsf{id},\star,\star,[\star]) \in \mathsf{List} \vee \mathsf{AList}$ then
2:     $\mathsf{tok} \leftarrow \mathsf{Revoke}(\mathsf{sk}_{\mathsf{RevA}},\mathsf{id})$
3:     $\mathsf{RevokeList} \leftarrow \mathsf{RevokeList} \cup \{\mathsf{tok}\}$
4: return $\mathsf{RevokeList}$
```

Fig. 1. Oracles for VLR-HABS security experiments.

Path Anonymity. This property guarantees anonymity of the signer as well as all intermediate authorities involved in attribute-delegation for attributes used to satisfy the signing policy. The definition for path anonymity for a VLR-HABS scheme is closely related to that given in [15], however we make adjustments to allow for the revocation feature. Our definition captures unlinkability for unrevoked signers, without considering backwards unlinkability [44,9] which splits time into different epochs and preserves unlinkability across them. The experiment for path anonymity, defined in Figure 2, requires a two-stage PPT adversary $(\mathcal{A}_1,\mathcal{A}_2)$ to distinguish which warrant and private key were used in the generation of the challenge VLR-HABS signature σ_b. Initially, $\mathcal{A}_1$ generates the authority and user hierarchy, utilising the registration and delegation oracles. A challenge VLR-HABS signature σ_b according to the predefined challenge bit b, using warrants and keys provided by the adversary. Then, with access to the tracing oracle, the adversary $\mathcal{A}_2$ guesses b'. We note that the game returns 0 if $\mathcal{A}$ revokes the identity in either of the warrants warr_0 and warr_1 that it provides the experiment. Since it does not have access to the revoke oracle in the second phase of the experiment, it cannot use this to help determine the challenge bit.

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

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

Non-frameability. Defined in Figure 3, this property captures traditional unforgeability notions, i.e., that no PPT adversary can create a VLR-HABS signature without having an honestly issued warrant for a set of attributes that satisfies the policy. It also forbids an adversary from framing another user, i.e. creating a verifiable VLR-HABS signature on behalf of a user for which the secret key is not

```
$\mathsf{Exp}^{\text{pa-}b}_{\mathsf{VLR\text{-}HABS},\mathcal{A}}(\lambda)$
1: $(\mathsf{pp},\mathsf{skd}_0,\mathsf{sk}_{\mathsf{TA}}) \leftarrow \mathsf{Setup}(1^\lambda)$
2: $((\mathsf{usk}_0,\mathsf{warr}_0),(\mathsf{usk}_1,\mathsf{warr}_1),\mathsf{m},\Psi) \leftarrow \mathcal{A}_1(\mathsf{pp},\mathsf{skd}_0 :$
       $\mathcal{O}_{\mathrm{RegU}},\mathcal{O}_{\mathrm{RegA}},\mathcal{O}_{\mathrm{CorrU}},\mathcal{O}_{\mathrm{CorrA}},\mathcal{O}_{\mathrm{Tr}},\mathcal{O}_{\mathrm{RevID}})$
3: if $|\mathsf{warr}_0| = |\mathsf{warr}_1|$ then
4:   $\sigma_0 \leftarrow \mathsf{Sign}((\mathsf{usk}_0,\mathsf{warr}_0),\mathsf{m},\Psi)$, $\sigma_1 \leftarrow \mathsf{Sign}((\mathsf{usk}_1,\mathsf{warr}_1),\mathsf{m},\Psi)$
5:   if $\mathsf{Verify}(\mathsf{pkd}_0,(\mathsf{m},\Psi,\sigma_0)) = 1$ and $\mathsf{Verify}(\mathsf{pkd}_0,(\mathsf{m},\Psi,\sigma_1)) = 1$ then
6:     $b' \leftarrow \mathcal{A}_2(\sigma_b : \mathcal{O}_{\mathrm{Tr}})$
7:     return $b' \wedge \mathcal{A}_2$ did not query $\mathcal{O}_{\mathrm{Tr}}(\mathsf{sk}_{\mathsf{TA}},(\mathsf{m},\Psi,\sigma_b))$
8: return 0
```

Fig. 2. Path Anonymity Experiment for VLR-HABS

```
$\mathsf{Exp}^{\mathsf{nf}}_{\mathsf{VLR\text{-}HABS},\mathcal{A}}(\lambda)$
1: $(\mathsf{pp},\mathsf{skd}_0,\mathsf{sk}_{\mathsf{TA}}) \leftarrow \mathsf{Setup}(1^\lambda)$
2: $((\sigma,\mathsf{m},\Psi),(\mathsf{upk}_j,\mathsf{warr},\hat{\pi})) \leftarrow \mathcal{A}(\mathsf{pp},\mathsf{pkd}_0,\mathsf{sk}_{\mathsf{TA}} :$
       $\mathcal{O}_{\mathrm{Att}},\mathcal{O}_{\mathrm{Sig}},\mathcal{O}_{\mathrm{RegU}},\mathcal{O}_{\mathrm{RegA}},\mathcal{O}_{\mathrm{CorrU}},\mathcal{O}_{\mathrm{CorrA}},\mathcal{O}_{\mathrm{RevID}})$
3: if $\mathsf{Verify}(\mathsf{pkd}_0,(\mathsf{m},\Psi,\sigma)) \wedge \mathsf{Judge}(\mathsf{pk}_{\mathsf{TA}},\mathsf{pkd}_0,(\mathsf{m},\Psi,\sigma),(\mathsf{upk}_j,\mathsf{warr},\hat{\pi}))$ then
4:   if $j \in \mathsf{HUList} \wedge \mathcal{A}$ did not query $\mathcal{O}_{\mathrm{Sig}}((\mathsf{usk}_j,\mathsf{warr}),\mathsf{m},\Psi)$ then , return 1
5:   if $\exists \mathsf{att} \in \mathsf{warr} \implies (\mathsf{pkd}_0,\mathsf{pkd}_1,\ldots,\mathsf{pkd}_{l-1},\mathsf{upk}_j) = \mathsf{warr}[\mathsf{att}] \wedge$
6:     $\forall j \in [0,l] : (j,\mathsf{pkd}_j,\mathsf{att}) \in \mathsf{HAList} \vee$
7:     $((\exists i \in [0,l-2].\ \mathcal{A}$ didn't query $\mathcal{O}_{\mathrm{Att}}(i,\cdot,\mathsf{att},\mathsf{pkd}_{i+1})$
           and $\forall j \in [0,i] : (j,\mathsf{pkd}_j,\mathsf{att}) \in \mathsf{HAList})\ \vee$
8:     $(\mathcal{A}$ did not query $\mathcal{O}_{\mathrm{Att}}(l-1,\cdot,\mathsf{att},\mathsf{upk}_j)$
           $\wedge \forall j \in [0,i] : (j,\mathsf{pkd}_j,\mathsf{att}) \in \mathsf{HAList})\ )$ then , return 1
9:   if $\Psi(A) \neq 1$, where $A := \{\mathsf{att} | \mathsf{att} \in \mathsf{warr}\}$ then , return 1
10:  if $\exists i$ s.t. $\mathsf{id}_i \in \mathsf{RevokeList} \cap \mathsf{warr}$ then , return 1
11: return 0
```

Fig. 3. Non-Frameability Experiment for VLR-HABS

known. The adversary wins if either it produces a valid VLR-HABS signature, or is able to perform delegation for at least one attribute on behalf of any honest authority that is not 'below' a corrupt authority. This trivially implies that the root authority must also remain honest, unlike the definition given in Dragan et al. [15]. We also modify the original definition to include extra winning conditions that capture the scenario the adversary is able to produce a signature that verifies despite using an ID that was revoked. This can be seen in line 10 of Figure 3. As in Dragan et al., $\mathcal{A}$ also wins if it can generate a signature for which its attribute do not satisfy the policy.

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

$$\mathsf{Adv}^{\mathsf{nf}}_{\mathsf{VLR\text{-}HABS},\mathcal{A}}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{nf}}_{\mathsf{VLR\text{-}HABS},\mathcal{A}}(\lambda) = 1]$$

Path Traceability. This property, defined in Figure 4, provides accountability for authorities in the delegation path. It ensures that any valid VLR-HABS signature can be traced (by the tracing authority) to the signer and the path of authorities that were involved in the issuance of the attributes. To win this game, the adversary $\mathcal{A}$ is required to satisfy one of two conditions. Firstly, it can output a VLR-HABS signature that verifies but cannot be traced (that is, the tracing algorithm fails), or secondly, one in which the tracing algorithm outputs a warrant containing at least one unknown IA or user, i.e., were not

```
$\mathsf{Exp}^{\mathsf{tr}}_{\mathsf{VLR\text{-}HABS},\mathcal{A}}(\lambda)$
1: $(\mathsf{pp},\mathsf{skd}_0,\mathsf{sk}_{\mathsf{TA}}) \leftarrow \mathsf{Setup}(1^\lambda)$
2: $((\sigma,\mathsf{m},\Psi),(\mathsf{upk},\mathsf{warr},\hat{\pi})) \leftarrow \mathcal{A}(\mathsf{pp},\mathsf{sk}_{\mathsf{TA}} : \mathcal{O}_{\mathrm{Att}},\mathcal{O}_{\mathrm{RegU}},\mathcal{O}_{\mathrm{RegA}},\mathcal{O}_{\mathrm{CorrU}},\mathcal{O}_{\mathrm{CorrA}},\mathcal{O}_{\mathrm{RevID}})$
3: if $\mathsf{Verify}(\mathsf{pkd}_0,(\mathsf{m},\Psi,\sigma))$ then ,
4:    if $\mathsf{Trace}(\mathsf{sk}_{\mathsf{TA}},(\mathsf{m},\Psi,\sigma)) = \perp$ then ,return 1
5:    if $\mathsf{Judge}(\mathsf{pk}_{\mathsf{TA}},\mathsf{pkd}_0,(\mathsf{m},\Psi,\sigma),(\mathsf{upk},\mathsf{warr},\hat{\pi})) \wedge$
6:       $(\exists \mathsf{att} \in \mathsf{warr} \Longrightarrow (\mathsf{pkd}_0,\mathsf{pkd}_1,\ldots,\mathsf{pkd}_{l-1},\mathsf{upk}) = \mathsf{warr}[\mathsf{att}] \wedge$
7:          $( (\exists i \in [0,l-2].\ i \in \mathsf{HAList} \wedge i+1 \notin \mathsf{AList}) \vee$
8:          $(l-1 \in \mathsf{HUList} \wedge (\cdot,\mathsf{upk},\mathsf{usk}) \notin \mathsf{List}) ) )$ then ,return 1
9: return 0
```

Fig. 4. Path Traceability Experiment for VLR-HABS

previously registered in List or AList. To prohibit trivial attacks, we require the attribute-issuing oracle to check that both entities are registered (are in List or AList) before returning a delegated attribute.

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

$$\mathsf{Adv}^{\mathsf{tr}}_{\mathsf{VLR\text{-}HABS},\mathcal{A}}(\lambda) = |\Pr[\mathsf{Exp}^{\mathsf{tr}}_{\mathsf{VLR\text{-}HABS},\mathcal{A}}(\lambda) = 1]|$$

4 Building Blocks

We now recall some useful building blocks and techniques that will be used in our VLR-HABS construction. We will use a zero-knowledge argument of knowledge to form the core of our construction, a trapdoor delegation technique to issue attributes and an identity-based encryption scheme that will allow the tracing authority to extract the warrant. First we recall a commitment scheme that will be compatible with our argument of knowledge.

KTX Commitment Scheme. Kawachi et al. [28] constructed a string commitment scheme $\mathsf{COM} : \{0,1\}^* \times \{0,1\}^{\tilde{m}/2} \to \mathbb{Z}_q^n$, such that:

- If $\tilde{m} > 2n(1+\eta)\log q$ for some positive η, then COM is statistically hiding.
- If the $\mathsf{SIS}^{\infty}_{n,\tilde{m},q,\beta}$ problem is hard, then COM is computationally binding.

We implicitly choose $\tilde{m}$ sufficiently large, e.g., $\tilde{m} = 4n\log q$, to make COM statistically hiding. It consists of 3 algorithms, Setup, COM and Open.

- Setup: Fix $n,m,\tilde{m} \in \mathbb{Z}$ and sample $\mathbf{A} \hookleftarrow \mathbb{Z}_q^{n\times(m+\tilde{m})}$. Output $\mathsf{pp} := (n,m,\tilde{m},\mathbf{A})$.
- $\mathsf{COM}_{\mathbf{A}}(\mathsf{m};\rho)$: On input of a message m and randomness $\rho \hookleftarrow \mathbb{Z}_q^{\tilde{m}}$, the commitment is computed as $\mathsf{co} := \mathsf{COM}_{\mathbf{A}}(\mathsf{m};\rho) = \mathbf{A}[\mathsf{m}||\rho]^T \mod q$.
- $\mathsf{Open}(\mathsf{pp},\mathsf{m},\mathsf{co},\rho)$. To open a commitment, reveal ρ and m, then check $\mathsf{co} \stackrel{?}{=} \mathbf{A}[\mathsf{m}||\rho]^T \mod q$.

4.1 Techniques for our Zero-Knowledge Argument of Knowledge

We recall the techniques of Ling et al. [34] to create a zero-knowledge argument of knowledge that will form the core of VLR-HABS.

Decomposition-Extension Technique. These techniques have been introduced in [34] and developed further in [30,38]. It is a permutation-based argument of knowledge that supports a range of relations, including for (I)SIS and LWE. In this work we rely on these techniques to achieve the strong anonymity properties of a VLR-HABS scheme. We introduce the notation and algorithms used to construct the argument. We start with a description of various groups and sets.

- $\mathcal{S}_n$: The permutation group of n elements. Let $\mathcal{S}_{n\times m}$ denote m copies $\mathcal{S}_n$. In particular, $\phi \hookleftarrow \mathcal{S}_{n\times m}$ operates on matrices of dimension $n \times m$ as m column-wise n-permutations.
- B_{3m} the set of all vectors in $\{-1,0,1\}^{3m}$, that have exactly m coordinates of each -1,0 and 1. We extend this notation to define $\mathsf{B}_{3m\times k}$ as k vectors from B_{3m}.
- B_{2m}. Similarly, the set of all vectors in $\{0,1\}^{2m}$, that have exactly m coordinates of each 0 and 1, i.e. with Hamming weight $m/2$. We also define $\mathsf{B}_{2m\times k}$ as k vectors from B_{2m}.
- $\mathsf{Secret}_\beta(k)$: The set of all vectors $\mathbf{z} = (\mathbf{z}_0||\mathbf{z}_1^0||\mathbf{z}_1^1||...||\mathbf{z}_d^0||\mathbf{z}_d^1) \in \mathbb{Z}^{m(2d+1)}$ consisting of $2d+1$ blocks of size m such that $||\mathbf{z}_i||_\infty \leqslant \beta$ and the d blocks, $\mathbf{z}_1^{1-k[1]},...,\mathbf{z}_s^{1-k[d]}$, are *zero-blocks* $\mathbf{0}^m$.
- $\mathsf{SecretExt}_\beta(k)$: The set of all vectors $\mathbf{z} = (\mathbf{z}_0||\mathbf{z}_1^0||\mathbf{z}_1^1||...||\mathbf{z}_d^0||\mathbf{z}_d^1) \in \{-1,0,1\}^{3m(2d+1)}$ consisting of $2d+1$ blocks of size $3m$ such that the $d+1$ blocks $\mathbf{z}_0,\mathbf{z}_1^{k[1]},...,\mathbf{z}_d^{k[d]}$ are elements of B_{3m} and the remaining blocks are *zero-blocks* $\mathbf{0}^{3m}$.
- $\overline{\mathsf{SecretExt}}_\beta(k)$: The set of all matrices such that each column, viewed as a vector $\mathbf{z} = (\mathbf{z}_0||\mathbf{z}_1^0||\mathbf{z}_1^1||...||\mathbf{z}_d^0||\mathbf{z}_d^1) \in \{-1,0,1\}^{2m(2d+1)}$ consisting of $2d+1$ blocks of size $2m$ such that the $d+1$ blocks $\mathbf{z}_0,\mathbf{z}_1^{k[1]},...,\mathbf{z}_d^{k[d]}$ are elements of B_{2m} and the remaining blocks are *zero-blocks* $\mathbf{0}^{2m}$.

Given a vector $\mathbf{z} = (\mathbf{z}_0||\mathbf{z}_1^{(0)}||\mathbf{z}_1^{(1)}||...||\mathbf{z}_d^{(0)}||\mathbf{z}_d^{(1)}) \in \mathbb{Z}^{3m(2d+1)}$ consisting of $2d+1$ blocks of size $3m$, and $\hat{\mathbf{z}} = (\mathbf{z}^{(0)}||\mathbf{z}^{(1)}||...||\mathbf{z}^{(\delta)}) \in \mathbb{Z}^{3\delta m(2d+1)}$ where $\mathbf{z}^{(i)}$ are potentially distinct vectors of the form $\mathbf{z}$. We define two sets of permutations:

- The set $\mathcal{P}$ of all permutations π that keep the arrangement of the blocks. Specifically, if $\pi \in \mathcal{P}$ then $\pi(\mathbf{z}) = (\tau_0(\mathbf{z}_0)||\tau_1^0(\mathbf{z}_1^0)||\tau_1^1(\mathbf{z}_1^1)||...||\tau_{ld}^0(\mathbf{z}_{ld}^0)||\tau_{ld}^1(\mathbf{z}_{ld}^1))$ where $\tau_0,\tau_1^0,\tau_1^1,...,\tau_{ld}^0,\tau_{ld}^1 \in \mathcal{S}_{3m}$, and $\mathcal{S}_{3m}$ is the symmetric group of $3m$ elements. We further denote $\hat{\mathcal{P}}$ to denote the restricted case that $l=1$.
- The set $\overline{\mathcal{P}}$ of all permutations π that keep the arrangement of the blocks. Specifically, if $\pi \in \overline{\mathcal{P}}$ then $\pi(\mathbf{Z}) = (\tau_0(\mathbf{Z}_0)||\tau_1^0(\mathbf{Z}_1^0)||\tau_1^1(\mathbf{Z}_1^1)||...||\tau_d^0(\mathbf{Z}_d^0)||\tau_d^1(\mathbf{Z}_d^1))$, where $\tau_0,\tau_1^0,\tau_1^1,...,\tau_d^0,\tau_d^1 \in \mathcal{S}_{2m\times n}$, and $\mathcal{S}_{2m\times n}$ is the symmetric group of $2m$ elements, sampled n times and applied to each of n columns in the matrix.
- The set $\mathcal{T} = \{T_e | e \in \{0,1\}^{ld}\}$, where for $e = e[1],...,e[ld], T_e \in \mathcal{T}$ rearranges the blocks: $T_e(\mathbf{z}) = (\mathbf{z}_0||\mathbf{z}_1^{e[1]}||\mathbf{z}_1^{1-e[1]}||...||\mathbf{z}_d^{e[ld]}||\mathbf{z}_d^{1-e[ld]})$.

In particular, given $t,c \in \{0,1\}^d, \pi \in \mathcal{P}$, and $\mathbf{z} \in \mathbb{Z}^{3m(2d+1)}$, where $\oplus$ denotes bit-wise addition mod 2, and $\circ$ is composition of permutations, it can be checked that:

$$\mathbf{z} \in \mathsf{SecretExt}_\beta(t) \Longleftrightarrow \pi(\mathbf{z}) \in \mathsf{SecretExt}_\beta(t) \Longleftrightarrow T_c \circ \pi(\mathbf{z}) \in \mathsf{SecretExt}_\beta(t \oplus c)$$

Decomposition. On input vector $\mathbf{z} = (z_1,z_2,...,z_m) \in \mathbb{Z}^m$ such that $||\mathbf{z}||_\infty \leqslant \beta$, the procedure VecDec outputs $p = \lfloor \log\beta \rfloor + 1$ vectors $\mathbf{w}_1,...,\mathbf{w}_p$ such that $\sum_{j=1}^p \beta_j \cdot \mathbf{w}_j = \mathbf{z}$. This is achieved by the following:

- For each $i \in [1,m]$, write the i^{th} element of $\mathbf{z}$ as $z_i = \beta_1 \cdot w_{i,1} + \beta_2 \cdot w_{i,2} + ... + \beta_p \cdot v_{i,p}$ where $\forall j \in [1,p] : w_{i,j} \in \{-1,0,1\}$, and $\beta_1 = \lceil \beta/2 \rceil, \beta_2 = \lceil (\beta - \beta_1)/2 \rceil, \beta_3 = \lceil (\beta - \beta_1 - \beta_2)/2 \rceil, ..., \beta_p = 1$.[2]
- For each $j \in [1,p]$, let $\mathbf{w}_j := (w_{1,j},w_{2,j},...,w_{m,j}) \in \{-1,0,1\}^m$. Output $\{\mathbf{w}_j\}_{j=1}^p$.

Extension. On input of a vector $\mathbf{w} \in \{-1,0,1\}^m$, VecExt extends the vectors $\mathbf{w}$ to a vector $\hat{\mathbf{w}} \in \mathsf{B}_{3m}$, computed as follows:

- Let $\eta^{(-1)},\eta^{(0)},\eta^{(1)}$ be the numbers of coordinates in $\mathbf{w}$ that equal -1,0,1.
- Pick a random vector $\mathbf{w}' \in \{-1,0,1\}^{2m}$ that has $m - \eta^{(-1)}$ coordinates equal to -1, $m - \eta^{(0)}$ coordinates equal to 0 and $m - \eta^{(1)}$ coordinates equal to 1. Output $\hat{\mathbf{w}} = (\mathbf{w}||\mathbf{w}') \in \mathsf{B}_{3m}$.

VecExt_2 is defined with input $\hat{\mathbf{w}} \in \{0,1\}^m$ and extends to $\mathbf{w} \in \mathsf{B}_{2m}$ of Hamming weight m, as follows.

$$\mathbf{w}[i] := \begin{cases} \hat{\mathbf{w}}[(i+1)/2] \text{ if } i \text{ is odd} \\ 1 - \hat{\mathbf{w}}[i/2] \text{ if } i \text{ is even} \end{cases}$$

[2] It was noted in [34] that any set of $\{\beta_i\}_i$ that allows any integer in the range $[0,\beta]$ to be expressed as a subset sum would suffice, but the choice here allows for no extraction gap, and thus is an optimal choice.

Preparing Secret Vectors. We define a process VecDE as follows:

- On input of a vector $\mathbf{z} \in \mathsf{Secret}_\beta(s)$ for s a bit-string of length d, parse $\mathbf{z}$ as $(\mathbf{z}_0||\mathbf{z}_1^0||\mathbf{z}_1^1||...||\mathbf{z}_d^0||\mathbf{z}_d^1) \in \mathbb{Z}^{m(2d+1)}$.
- Execute VecDec on each $\mathbf{z}_i^{s[i]}$ for $i \in [1,d]$ to obtain $(d+1)p$ vectors $\mathbf{w}_i$.
- Execute $\mathsf{VecExt}(\mathbf{w}_i), i \in [1,d]$ to compute the $(d+1)p$ vectors $\hat{\mathbf{w}} \in \mathsf{B}_{3m}$, denoted by $\{\{\hat{\mathbf{w}}_{i,j}\}_{j=1}^p\}_{i=0}^d$.
- For each $j \in [1,p]$, let $\mathbf{y}_j = (\hat{\mathbf{w}}_{0,j}||\hat{\mathbf{w}}_{1,j}^0||\hat{\mathbf{w}}_{1,j}^1||...||\hat{\mathbf{w}}_{d,j}^0||\hat{\mathbf{w}}_{d,j}^1) \in \mathbb{Z}^{3m(2d+1)}$ where $\hat{\mathbf{w}}_{i,j}^{1-s[i]}$ is the zero vector $\mathbf{0}^{3m}$.
- Output $\mathbf{y}_j \in \mathsf{SecretExt}_\beta(s)$ for $j \in [1,p]$.

Useful Functions. We now define two functions $f_k : \mathbb{Z}_q^{lmd} \to \mathbb{Z}_q^{m(2ld+1)}$ and $\hat{f}_k : \mathbb{Z}_q^{n \times lmd} \to \mathbb{Z}_q^{n \times m(2ld+1)}$, that are indexed by a vector $\mathbf{k} \in \{0,1\}^{2ld}$.

$$f_{\mathbf{k}}(\mathbf{z}) := (\mathbf{z}||k[1]\mathbf{z}_1||(1-k[1])\mathbf{z}_1||...||k[d]\mathbf{z}_d||(1-k[d])\mathbf{z}_d)^T$$

$$\hat{f}_{\mathbf{k}}(\mathbf{X}) := (f_{\mathbf{k}}(\mathbf{X}_{i,1})||f_{\mathbf{k}}(\mathbf{X}_{i,2})||...||f_{\mathbf{k}}(\mathbf{X}_{i,m})) \text{ for } i \in [1,n]$$

Matrix Decomposition. We define MatDE based on VecDE, except it calls VecExt_2 instead of VecExt. Denote $\mathbf{B}_{j,f}$ for $f \in [1,k]$ to describe the k^{th} column of $\mathbf{B}$, and s a binary string of length d:

- Input matrix $\mathbf{B} \in \mathbb{Z}_q^{m \times k}$
- Return $\{\mathbf{B}^{(i)}\}_{i=1}^p \leftarrow [\mathsf{VecDE}(\mathbf{B}_{j,1},s)||...||\mathsf{VecDE}(\mathbf{E}_{j,k},s)] \in \overline{\mathsf{SecretExt}}_\beta(s)$ for $i \in [1,p]$.

We define MatDec to be $\{\mathbf{E}^{(i)}\}_{i=1}^p \leftarrow [\mathsf{VecDec}(\mathbf{E}_{j,1})||...||\mathsf{VecDec}(\mathbf{E}_{j,k})]$ for $f \in [1,k]$.

Matrix Extension. On input of a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m(2ld+1)}$, the procedure MatExt outputs a matrix $\mathbf{A}^* \in \mathbb{Z}_q^{n \times 3m(2ld+1)}$. Parse $\mathbf{A} = [\tilde{\mathbf{A}}||\mathbf{A}_0||\mathbf{A}_1||...||\mathbf{A}_d||...||\mathbf{A}_{(l-1)d}||...||\mathbf{A}_d]$ and define
$\mathbf{A}^* = [\tilde{\mathbf{A}}||\mathbf{A}_0||\mathbf{0}^{n \times 2m}||...||\mathbf{A}_d||\mathbf{0}^{n \times 2m}||...||\mathbf{A}_{(l-1)d}||...||\mathbf{A}_d||\mathbf{0}^{n \times 2m}]$. Also define
$\overline{\mathbf{A}}^* = [\tilde{\mathbf{A}}||\mathbf{A}_0||\mathbf{0}^{n \times 2m}||...||\mathbf{A}_{ld}||\mathbf{0}^{n \times 2m}]$.

Proving consistency of identities. During the argument of knowledge, we will need to show that the same identity id is used throughout. This is particularly challenging as id is encoded differently across different sub-relations, so we are required to do more than just reusing a single commitment, as one might intuitively hope. In particular, we will be required to relate id to $\mathbf{A}_{\mathsf{id}}$ in the equation $\mathbf{A}_{\mathsf{id}}\mathbf{z} = \mathbf{u}$ by proving the relation instead showing $\mathbf{A}f_{\mathsf{id}^*}(\mathbf{z}) = \mathbf{u}$ and $f_{\mathsf{id}^*}(\mathbf{z}) \in \mathsf{Secret}_\beta(\mathsf{id}^*)$, that is, the output vector of $f_{\mathsf{id}^*}(\mathbf{z})$ has the correct structure that ensures id^* was used. At a high level, to argue that $\mathsf{id} \in \{0,1\}^d$, we first extend id to $\mathsf{id}^* \in \mathsf{B}_{2d}$ (the set of all vectors in $\{0,1\}^{2d}$ with Hamming weight d), and then show that a random permutation of id^* belongs to the set B_{2d} which implies that $\mathsf{id} \in \{0,1\}^d$.

4.2 Signatures and Basis Delegation

We recall the notion of Bonsai Trees [12] and a natural signature scheme that follows.
Delegating a Short Basis. A trapdoor for a matrix $\mathbf{A}$, for Bonsai trees, is a short basis for $\Lambda^\perp(\mathbf{A})$. Cash et al. [12] show how one can *delegate* this basis to an extended matrix $\mathbf{A}' := [\mathbf{A}||\tilde{\mathbf{A}}]$ such that the new basis $\mathbf{T}_{\mathbf{A}'}$ is also short, but for which is is hard to recover $\mathbf{T}_{\mathbf{A}}$. We note that the extension matrix $\tilde{\mathbf{A}} \in \mathbb{Z}_q^{n \times m'}$ is flexible, and in particular m' can be any integer greater than 0. We recall some algorithms from Cash et al. whose existence and security properties are proven in [12].

- $\mathsf{GenBasis}$ (n,m,q). There is a fixed constant $C > 1$ and a probabilistic polynomial-time algorithm $\mathsf{GenBasis}$ that, for poly(n)-bounded $m \geqslant Cn\log q$, outputs and $\mathbf{S} \in \mathbb{Z}^{m \times m}$ such that $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, the distribution of $\mathbf{A}$ is within negl (n) statistical distance of uniform, $\mathbf{S}$ is a basis of $\Lambda^\perp(\mathbf{A})$, and $||\mathsf{S}|| \leqslant \mathcal{O}(\sqrt{n\log q})$.

– $\mathsf{SampleD}(\mathbf{S},\mathbf{A},\mathbf{u},\beta)$. For given $\mathbf{A}\in\mathbb{Z}_q^{n\times m}$, a trapdoor $\mathbf{S}$ for $\mathbf{A}$, a parameter $\beta=\omega(n\log q\log n)$ and a vector $\mathbf{u}\in\mathbb{Z}_q^n$, there is a PPT algorithm $\mathsf{SampleD}$ that returns a $\mathbf{z}\in\Lambda_{\mathbf{u}}^{\perp}(\mathbf{A})$ with distribution statistically close to $\mathcal{D}_{\Lambda_{\mathbf{u}}^{\perp}(\mathbf{A}),\alpha}$, and norm at most β.
– $\mathsf{ExtBasis}((\mathbf{T_A}=\mathbf{S}),\mathbf{A}')$ takes as input a short-basis $\mathbf{T_A}$ for a lattice $\Lambda^{\perp}(\mathbf{A})$, and $\mathbf{A}'$ which describes a superlattice of $\mathbf{A}$, i.e. $\mathbf{A}'=[\mathbf{A}||\bar{\mathbf{A}}]$. It returns an extended basis for $\mathbf{A}'$ as $\mathbf{T}_{\mathbf{A}'}:=\mathbf{S}'\leftarrow\begin{pmatrix}\mathbf{S} & \mathbf{W}\\ \mathbf{0} & \mathbf{I}\end{pmatrix}$ where $\mathbf{W}$ is such that $\mathbf{AW}=-\bar{\mathbf{A}} \mod q$.
– $\mathsf{RandBasis}(\mathbf{S},\mathbf{A},\mathbf{y},\beta)$ takes as input a short basis $\mathbf{S}$ for $\mathbf{A}$ with respect to a syndrome $\mathbf{y}$, and returns a randomised basis $\mathbf{S}'$ with small loss in quality.

The primary function of this delegation mechanism in our scheme will be to support attribute delegation and issuance. The matrices $\mathbf{A}$ will be the public key of an authority or user, the attributes will be the syndromes, and delegation from authority i to j will consist of extending the short basis $\mathbf{T}_{\mathbf{A}_i}$ to $\mathbf{T}_{[\mathbf{A}_i||\mathbf{A}_j]}$.
Bonsai Signature [12]. The signature scheme is instantiated with the following parameters: a message m of length $k=\mathcal{O}(\log q)$, a bound $\beta=\mathcal{O}(\sqrt{n\log q})$, a Gaussian parameter $\alpha=\beta\cdot\omega(\sqrt{\log q})$, a uniform matrix $\mathbf{A}_0\in\mathbb{Z}_q^{n\times m}$, with a corresponding basis $\mathbf{S}_0$ of $\Lambda^{\perp}(\mathbf{A}_0)$ such that $||\mathbf{S}_0||\leqslant\beta$. Finally, choose $\{A_i^b\}_{i=1}^k$ and $b\in\{0,1\}$. Set $\mathsf{pk}=(\mathbf{A}_0,\{\mathbf{A}_i^b\}_{i=1}^k)$, $\mathsf{sk}=(\mathbf{S}_0)$, return $(\mathsf{sk},\mathsf{pk})$.

– $\mathsf{Sign}(\mathsf{sk},\mathsf{m})$ takes as input a mess m and secret key $\mathsf{sk}=\mathbf{S}_0$, it returns the signature σ computed as follows. Define $\mathbf{A}_{\mathsf{m}}:=[\mathbf{A}_0||\mathbf{A}_1^{\mathsf{m}[1]}||...||\mathbf{A}_k^{\mathsf{m}[k]}]$, and compute a short vector $\mathbf{z}\leftarrow\mathsf{SampleD}(\mathsf{ExtBasis}(\mathbf{S}_0,\mathbf{A}_{\mathsf{m}}),\mathbf{0},\beta)$. In the rare event that $\mathbf{z}=\mathbf{0}$ or $||\mathbf{z}||\geqslant\beta\cdot\sqrt{m'}$, then resample $\mathbf{z}$. Set $\sigma=\mathbf{z}$.
– $\mathsf{Verify}(\mathsf{pk},\sigma)$ upon input of a candidate signature $\sigma=\mathbf{z}$, output 'accept' if $\mathbf{z}\neq\mathbf{0},||\mathbf{z}||\leqslant\beta\cdot\sqrt{m'}$ and $\mathbf{A}_{\mathsf{m}}\mathbf{z}=\mathbf{0}$, else output 'reject'.

This scheme offers static unforgeability [12], which requires the adversary to submit signing queries before it learns the verification key, under the SIS assumption. Using a family of chameleon hash functions, there is a generic transformation from static to adaptive unforgeability [29]. For our construction, we will use the lattice-based Chameleon Hash Function from Ducas and Miccancio [16], which we denote $\mathcal{H}_2:\mathbb{Z}_q^*\to\{0,1\}^k$. The security properties intuitively require recovery of a short basis for $\mathbf{A}$ given a delegated basis $\mathbf{T}_{\mathbf{A}'}$ for $\mathbf{A}'$ to be hard. This is the case due to the statistical independence of $\mathbf{T_A}$ and $\mathbf{T}_{\mathbf{A}'}$ [12]. We will use Bonsai Signatures to prove attribute delegation and as a one-time signature in VLR-HABS.
Arguing in Zero-Knowledge. Proving knowledge of a valid Bonsai signature requires proving the following two relations, $||\mathbf{z}||_\infty\leqslant\beta$ and $\mathbf{A}_{\mathsf{id}}\mathbf{z}=\mathbf{u} \mod q$. To show the latter, we use the techniques of Langlois et al. [30]. We rewrite the equation so that the matrix $\mathbf{A}$ is independent of id, we do this by instead showing $\mathbf{A}f_{\mathsf{id}}(\mathbf{z})=\mathbf{u} \mod q$, and that $f_{\mathsf{id}}(\mathbf{z})$ is constructed correctly, that is, $f_{\mathsf{id}}(\mathbf{z})\in\mathsf{Secret}_\beta(\mathsf{id})$. Now that only one term forms the witness, we can proceed to efficiently prove the relation in zero-knowledge. This is done as follows:

– Extend $\mathbf{A}$ to $\mathbf{A}^*\leftarrow\mathsf{MatExt}(\mathbf{A})$, and compute $\mathbf{z}_1^*,...,\mathbf{z}_p^*\leftarrow\mathsf{VecDE}(\mathbf{z},\mathsf{id})$.

Since $\mathbf{z}^*=\sum_{j=1}^p\beta_j\cdot\mathbf{z}_j^*$, and the last two columns of $\mathbf{A}^*$ are 0, then we have the following:

$$\mathbf{A}^*\left(\sum_{j=1}^p\beta_j\cdot\mathbf{z}_j^*\right)=\mathbf{u} \mod q \Longleftrightarrow \mathbf{Az}=\mathbf{u} \mod q$$

In the Stern-like protocol, we will show the following:

$$\mathbf{A}^*\left(\sum_{j=1}^p\beta_j(\mathbf{z}_j^*+\mathbf{r}_{\mathbf{z}_j^*})\right)-\mathbf{u}=\mathbf{A}^*\left(\sum_{j=1}^p\beta_j\mathbf{r}_{\mathbf{z}_j^*}\right)$$

To complete the proof, the prover is also required to show $\mathbf{z}_1^*,...,\mathbf{z}_p^*\in\mathsf{SecretExt}_\beta(\mathsf{id})$.

4.3 Identity-based Encryption

We will use the Identity-based Encryption (IBE) scheme from Gentry, Peikert and Vaikuntanathan (GPV-IBE) [22], which has the following parameters: a security parameter λ, a message m with length l, an integer $n=\mathcal{O}(\lambda)$, a prime modulus $q=\mathcal{O}(n^2)$, an integer $m\geqslant 2n\log q$, an integer bound $\beta=\mathcal{O}(\sqrt{n})$, an efficiently sampleable distribution χ over $\mathbb{Z}$ parametrised by $\alpha\leqslant 1/\sqrt{m+1}(\omega(\sqrt{\log m\cdot\log n}))$ and a hash function $\mathcal{H}:\{0,1\}\rightarrow\mathbb{Z}_q^{n\times l}$.

The master public-secret key pair is $(\mathsf{sk},\mathsf{pk})=(\mathbf{T_D},\mathbf{D})$ generated via $\mathsf{GenBasis}(n,m,q)$. For a identity $\mathsf{id}\in\{0,1\}^*$, the extraction algorithm first hashes id to a matrix $\mathbf{N}=[\mathbf{g}_1||...||\mathbf{g}_l]\in\mathbb{Z}_q^{n\times l}$ using $\mathcal{H}$ and then runs $\mathbf{s}_i\leftarrow\mathsf{SampleD}(\mathbf{D},\mathbf{T_D},\mathbf{g}_i,\alpha)$ for $i\in[1,...,l]$. Finally, it defines the decryption key for a user id as $\mathbf{S}_{\mathsf{id}}=[\mathbf{s}_1||...||\mathbf{s}_l]\in\mathbb{Z}_q^{m\times l}$.

- $\mathsf{Enc}(\mathsf{pk}=\mathbf{D},\mathsf{m})$ outputs a ciphertext $\mathbf{f}:=(\mathbf{f}_1,\mathbf{f}_2,)$ computed as $\mathbf{f}_1\leftarrow\mathbf{D}^T\cdot\mathbf{s}+\mathbf{e}_1\in\mathbb{Z}_q^m$ and $\mathbf{f}_2\leftarrow\mathbf{N}^T\cdot\mathbf{s}+\mathbf{e}_2+\mathsf{m}\cdot\lfloor\frac{q}{2}\rfloor\in\mathbb{Z}_q^l$, where $\mathbf{e}_1,\mathbf{e}_2\leftarrow\chi$, $\mathbf{s}\hookleftarrow\mathbb{Z}_q^n$ and $\mathbf{N}=\mathcal{H}(\mathsf{id})$.
- $\mathsf{Dec}(\mathsf{sk}=\mathbf{S}_{\mathsf{id}},\mathbf{f})$ outputs the plaintext m computed via $\mathsf{m}'\leftarrow\mathbf{f}_2-\mathbf{S}_{\mathsf{id}}^T\mathbf{f}_1\in\mathbb{Z}_q^l$. For each $i\in[1,...,l],\mathsf{m}[i]=0$ if $\mathsf{m}'[i]$ is closer to 0 than $\lfloor\frac{q}{2}\rfloor$, otherwise $\mathsf{m}[i]=1$. Return $\mathsf{m}=[\mathsf{m}[1]||...||\mathsf{m}[l]]$.

This scheme in [22] is proven to be IND-CPA secure under the $\mathsf{LWE}_{n,m,q,\chi}$ assumption. We will use the CHK transform [11] to achieve a IND-CCA secure PKE.

Proving Correctness of Encryption. Here we show how to make it compatible with the Stern-like techniques of [34]. For our scheme, the signer will be required to show that each identity that appears in the delegation path of an attribute is correctly encrypted into a ciphertext under a one-time identity, usings tracing authority's public key. Using the TA's secret key, it can recover identities from a signature. For a plaintext $\mathsf{m}\in\{0,1\}^d$, using the Hermite Normal Form (HNF) variants of LWE, the relation among the related objects can be expressed as:

$$\mathbf{Pe}+(0^{g-d}||\lfloor q/2\rfloor\mathsf{m})=\mathbf{f}\mod q,\text{ where }\mathbf{P}=\begin{pmatrix}\mathbf{D}^T & \mathbf{I}\\ \mathbf{N}^T & \mathbf{I}\end{pmatrix}$$

where $\mathbf{P}\in\mathbb{Z}_q^{g\times h}$ (for $g=m+l,h=n+m+l$) and $\mathbf{I}$ is the identity matrix, $\mathbf{f}=[\mathbf{f}_1||\mathbf{f}_2]^T\in\mathbb{Z}_q^g$ is a ciphertext, $\mathbf{e}:=[\mathbf{s}||\mathbf{e}_1||\mathbf{e}_2]^T\in\mathbb{Z}_q^h$ is the encryption randomness satisfying $||\mathbf{e}||_\infty\leqslant\beta$. To construct a verifiable encryption protocol given $(\mathbf{P},\mathbf{f})$, the prover with knowledge of $(\mathbf{e},\mathsf{m})$, can argue in zero-knowledge that $\mathbf{f}$ is a correct encryption of m. Hence, we are required to show:

- To argue that $\mathsf{m}\in\{0,1\}^d$, we extend m to $\mathsf{m}^*\in\mathsf{B}_{2d}$ (the set of all vectors in $\{0,1\}^{2d}$ with Hamming weight d), then use a random permutation to show $\mathsf{m}^*\in\mathsf{B}_{2d}$.
- To argue that $||\mathbf{e}||_\infty\leqslant\beta$, we form the vectors $\mathbf{e}_1,...,\mathbf{e}_p\leftarrow\mathsf{VecDec}(\mathbf{e})$, and use random permutations ϕ_j to hide the value of $\mathbf{e}_j$, and then show $\phi_j(\mathbf{e}_j)\in\mathsf{B}_{3h}$.
- Next, we define the following two extended matrices:

$$\mathbf{P}^*=[\mathbf{P}||0^{g\times h}]\in\mathbb{Z}_q^{g\times 3h};\quad\mathbf{Q}=\begin{pmatrix}0^{(g-d)\times d} & 0^{(g-d)\times d}\\ \lfloor q/2\rfloor\mathbf{I}_{d\times d} & 0^{l\times d}\end{pmatrix}\in\{0,\lfloor q/2\rfloor\}^{g\times 2d}$$

- We then have that:

$$\mathbf{P}^*\left(\sum_{j=1}^{p}\beta_j\mathbf{e}_j\right)+\mathbf{Q}\mathsf{m}^*=\mathbf{Pe}+(0^{g-d}||\lfloor q/2\rfloor\mathsf{m})=\mathbf{f}\mod q$$

In Stern's framework, we will use masking vectors to hide the true values of the wtiness, and instead show that:

$$\mathbf{P}^*\left(\sum_{j=1}^{p}b_j(\mathbf{e}_j+\mathbf{r}_{\mathbf{e}_j})\right)+\mathbf{Q}(\mathsf{m}^*+\mathbf{r}_{\mathsf{m}^*})-\mathbf{f}=\mathbf{P}^*\left(\sum_{j=1}^{p}b_j\mathbf{r}_{\mathbf{e}_j}\right)+\mathbf{Q}\mathbf{r}_{\mathsf{m}^*}\mod q$$

where $\mathbf{r}_{\mathbf{e_j}}\in\mathbb{Z}_q^{3h}$ for $j\in[1,p]$ and $\mathbf{r}_{\mathsf{m}^*}\in\mathbb{Z}_q^{2d}$ are the uniformly random masking vectors.

Proving Correctness of Decryption. Parse a ciphertext $\mathbf{f}$ as $[\mathbf{f}_1||\mathbf{f}_2]^T$, each in $\mathbb{Z}_q^d$, we aim to show that a ciphertext correctly decrypts to a candidate message under the identity's secret key derived from the master key pair. That is, we show the relation $\mathsf{m}' = \mathbf{f}_2 - \mathbf{S}_{\mathsf{id}}\mathbf{f}_1$. For our scheme, each of the components will be public so we are not required to produce a proof, as knowledge of $\mathbf{S}_{\mathsf{id}}$ is sufficient. The Tracing Authority will perform this action part alongside producing the plaintext m, to recover the identities contained within the signature. It is publicly verifiable only when $\mathbf{S}_{\mathsf{id}}$ is published.

4.4 Inner-Product Signing Policies

In this work, we consider inner-product predicates, which capture disjunctive, conjunctive and threshold policies, and polynomial evaluations. We say that a set of attributes A satisfies the policy Ψ if the inner product of an attribute vector $\mathbf{a}$ and predicate vector $\mathbf{p}$ is 1 [3]. That is:

$$\Psi(A) = 1 \iff \langle \mathbf{a}, \mathbf{p} \rangle = 1 \text{ where } \mathbf{a} = [\mathbf{a}_1||...||\mathbf{a}_\delta]^T \text{ for } \mathbf{a}_j \in A, \text{ and } \mathbf{p} = [\mathbf{p}_1||..||\mathbf{p}_\delta]^T$$

We note that for construction, if $\delta > 1$, then the policy Ψ must have a certain structure. Precisely, this is of the form $\Psi = \wedge_{i=1}^{\delta} \Psi_i(\mathbf{a}_i)$. That is, it consists of a conjunction of *sub-policies* Ψ_i that are themselves represented by inner-products, but only over single attributes. We stress this supports Conjunctive Normal Form (CNF) formulas and therefore contains all $(\wedge,\vee)$-policies (inc. threshold). If we consider the case that $\delta = 1$, i.e. a user submits a single attribute to the signing policy, then we obtain the scenario considered in *all* existing attribute-based signatures that support inner-product policies. Thus this seemingly restricted form for the policy Ψ is actually more general than both HABS constructions in [15,21] and all existing IP-ABS schemes [46,57]. We do not consider this restriction further, as we generically refer to a policy vector $\mathbf{p}$ and make the assumption it takes this form.

We will require a zero-knowledge argument of knowledge of this relation, and we detail how we achieve this in the Decomposition-Extension framework as follows. Let $\tilde{\mathbf{a}} = \mathsf{bin}(\mathbf{a}) \in \{0,1\}^{\delta n \lceil \log q \rceil}$, then we have that $\mathbf{a} = \mathbf{G} \cdot \tilde{\mathbf{a}}$, i.e. the binary decomposition of $\mathbf{a}$, a gadget matrix $\mathbf{G} := \mathbb{I} \otimes [1,2,...,2^{\lceil \log q \rceil - 1}]$. Then we can rewrite this as

$$\langle \mathbf{G} \cdot \tilde{\mathbf{a}}, \mathbf{p} \rangle = 1 \mod q.$$

We further transform it, $\mathbf{a}' \leftarrow \mathsf{VecExt}(\tilde{\mathbf{a}}), \mathbf{G}^* \leftarrow (\mathbf{G}||\mathbf{0}^{n \times n \log q}) \in \mathbb{Z}_q^{n \times 2n \log q}$, which allows us to write:

$$\langle \mathbf{G}^* \cdot \mathbf{a}', \mathbf{p} \rangle = \langle \mathbf{a}, \mathbf{p} \rangle = 1 \mod q.$$

In the Stern-like protocol, we will use the linearity of the inner product to instead show the following equivalent identity, where $\mathbf{r}_a$ is a randomly sampled masking vector.

$$\langle \mathbf{G}^* \cdot \left(\mathbf{a}' + \mathbf{r}_{\mathbf{a}'}\right), \mathbf{p} \rangle - \langle \mathbf{G}^* \cdot \mathbf{r}_{\mathbf{a}'}, \mathbf{p} \rangle = 1 \mod q.$$

5 VLR-HABS Scheme

In this section we detail the core contributions. Firstly we introduce the VLR mechanism in Section 5.1, then the zero-knowledge protocol in Section 5.2, and present the scheme itself in Section 5.3.

5.1 New VLR Mechanism

We introduce a novel verifier-local revocation scheme that relies on the LWE and SIS hardness assumptions. For our scheme, a central authority (RevA) maintains a list of revoked identities in a list $\mathsf{RevokeList}$. A user is required to produce and publish privacy-preserving revocation tokens during the signing phase of the signature scheme. As part of verification, values from $\mathsf{RevokeList}$ are used to check whether the revocation tokens pass or fails verification.

To create a revocation token, the user samples a uniform binary matrix $\mathbf{B} \hookleftarrow \mathbb{B}^{m \times (d+1)n}$ and computes the LWE instance $\mathbf{C}_{\mathsf{id}} = \mathbf{B}\mathbf{R}_{\mathsf{id}} + \mathbf{E}$ for each id in the delegation paths, where $\mathbf{R}_{\mathsf{id}}$ is the

[3] The choice of 1 here is arbitrary, and any non-zero $c \in \mathbb{Z}_q$ can be used provided it is consistent amongst all parties.

encoded id of the entity, and $\mathbf{E}$ is a small error matrix sampled from a distribution χ. This is repeated for each identity contained in the delegation paths of the obtained attributes, i.e. $\mathsf{id} \in \mathsf{warr}$. In this work, we compute $\mathbf{R}_{\mathsf{id}} = [\mathbf{R} || \mathbf{R}_1^{\mathsf{id}[1]} || ... || \mathbf{R}_d^{\mathsf{id}[d]}]$ where $\mathsf{id}[i]$ is the i^{th} bit of id and $\mathbf{R}_i^b$ are uniformly sampled matrices with elements in $\mathbb{Z}_q$, and are published in the public parameters. Finally, $\mathbf{R}$ is the public key of the RevA, whose corresponding secret key is a trapdoor $\mathbf{T_R}$ that allows RevA to solve SIS instances with respect to $\mathbf{R}$. At a high level, due to the pseudo-randomness of binary-secret LWE, we argue that $\mathbf{C}$ is computationally close to a sample from a uniform distribution, thus no adversary can learn the identity committed to in $\mathbf{C}$, which more generally maintains the anonymity properties of the VLR-HABS scheme.

To revoke an identity, RevA uses its trapdoor for $\mathbf{R}$ to compute an extended trapdoor for $\mathbf{R}_{\mathsf{id}}$, using the $\mathsf{ExtBasis}$ and $\mathsf{RandBasis}$ algorithms. It then invokes $\mathsf{SampleD}$ to compute a small vector $\mathbf{y}$ such that $\mathbf{R}_{\mathsf{id}}\mathbf{y} = \mathbf{0}$, i.e. solving the SIS problem for $\mathbf{R}_{\mathsf{id}}$. It appends $\mathbf{y}$ to a public revocation list $\mathsf{RevokeList}$. During the verification phase, the verifier obtains $\mathsf{RevokeList}$ from the revocation authority and computes $\mathbf{C}_{\mathsf{id}}\mathbf{y} = \mathbf{B}\mathbf{R}_{\mathsf{id}}\mathbf{y} + \mathbf{E}\mathbf{y}$ for each $\mathbf{y} \in \mathsf{RevokeList}$, where $\mathbf{B}$ is a binary matrix of size $n_3 \times m_3$ and $\mathbf{E}$ is an error matrix of size $n_3 \times k_3$. If id has been revoked, then $\mathbf{R}_{\mathsf{id}}\mathbf{y} = \mathbf{0}$ for some $\mathbf{y} \in \mathsf{RevokeList}$ and hence $\mathbf{C}_{\mathsf{id}}\mathbf{y} = \mathbf{B}\mathbf{R}_{\mathsf{id}}\mathbf{y} + \mathbf{E}\mathbf{y} = \mathbf{B}\mathbf{0} + \mathbf{E}\mathbf{y} = \mathbf{E}\mathbf{y}$ and $||\mathbf{C}_{\mathsf{id}}\mathbf{y}|| = ||\mathbf{E}\mathbf{y}|| \leqslant n\beta^2$. If $||\mathbf{C}_{\mathsf{id}}\mathbf{y}|| > n\beta^2$ for all $\mathbf{y} \in \mathsf{RevokeList}$ and $\mathbf{C}_{\mathsf{id}}$ in the signature, then the verifier is assured that the signature was not generated using a revoked id.

To show the correctness of $\mathbf{C}$ and that it contains the IDs encrypted in the ciphertext, the signer is required to generate a zero-knowledge proof. In the decomposition-extension framework for Stern-like protocols, this is done by instead letting $\mathbf{R}^* := [\mathbf{R} || \mathbf{R}_1^0 || \mathbf{R}_1^{(1)} || ... || \mathbf{R}_d^{(0)} || \mathbf{R}_d^{(1)}]$ and proving the following relation:

$$\mathbf{C}_{\mathsf{id}} = \hat{f}_{\mathsf{id}}(\mathbf{B})\mathbf{R}^* + \mathbf{E}.$$

This is a matrix-extension of the trick we used earlier in Section 4.1 to remove the dependency of $\mathbf{A}_{\mathsf{id}}$ on id, which makes the resulting relation linear, and is efficiently provable. We also note that $\mathbf{R}^*$ is now public so the prover only needs to hide $\mathbf{B}, \mathbf{E}$ and id as part of the witness.

5.2 Zero-Knowledge Protocol

We now construct a Stern-like protocol that will form the core building block of our VLR-HABS scheme. The protocol will allow a signer to convince the verifier in zero-knowledge that:

1. The warrant contains a set of committed attributes that satisfy the policy.
2. For each attribute in the warrant, the signers possess a valid delegation path.
3. The ciphertext is a correct encryption of the IDs that appear in the warrant.
4. The signer's revocation token is correctly committed via an LWE function.

The protocol is instantiated with the following parameters:

– Public parameters: $\mathbf{A}, \mathbf{R}, \{\mathbf{A}_i^b\}_{i=1}^{ld}, \{\mathbf{R}_i^b\}_{i=1}^{d}, \mathbf{G}^*, \mathbf{P}^*, \mathbf{Q}, \{\mathbf{C}_{\mathsf{id}}\}_{\mathsf{id} \in \mathsf{warr}}, \{\mathbf{f}_i\}_{i=1}^{\delta}, \mathbf{p}, \mathbf{u}$.
– The prover's witness are the vectors $\mathsf{uid}, \mathbf{z}_0, \{\mathbf{z}_i, \mathbf{a}_i, \mathbf{e}_i\}_{i=1}^{\delta}, \{\mathsf{id}\}_{\mathsf{id} \in \mathsf{warr}}$ and the matrices $\{\mathbf{B}_{\mathsf{id}}, \mathbf{E}_{\mathsf{id}}\}_{\mathsf{id} \in \mathsf{warr}}$.

The goal is prove the following relation:

$$\mathcal{R}_1 := \begin{cases} \mathbf{A}_{[\mathsf{id}_1 || ... || \mathsf{id}_l]}(\mathbf{z}_i) = \mathbf{a}_i \mod q_1, i \in [1,\delta] \text{ and } \mathsf{id}_j \in \mathsf{warr}[\mathbf{a}_i] \\ \mathbf{C}_{\mathsf{id}} = \mathbf{B}\mathbf{R}_{\mathsf{id}} + \mathbf{E} \mod q_3 \text{ for } \mathsf{id} \in \mathsf{warr} \\ \mathbf{f}_{\mathsf{id}} = \mathbf{P}^*\mathbf{e} + \mathbf{Q}[\mathsf{id}_1 || ... || \mathsf{uid}] \mod q_2 \text{ for every } \mathsf{id} \in \mathsf{warr}[\mathbf{a}_i],\ i \in [1,\delta] \\ \langle \mathbf{a}, \mathbf{p} \rangle = 1 \text{ where } \mathbf{a} = [\mathbf{a}_1 || ... || \mathbf{a}_\delta] \wedge \mathbf{A}_{\mathsf{uid}}(\mathbf{z}_0) = \mathbf{u} \mod q_1 \end{cases}$$

A full and detailed description of the protocol is provided in Appendix A.

Theorem 1. *Let* COM *be a statistically hiding and computationally binding string commitment scheme. Then the protocol given in Section 5.2 is a zero-knowledge argument of knowledge with perfect completeness with soundness error* $2/3$. *Explicitly, that is:*

– *There exists a polynomial-time simulator that outputs an accepting transcript that is statistically close to a transcript produced by an honest prover with a valid witness.*
– *There exists a polynomial-time extractor, such that, on input of a commitment* CMT *and 3 responses* $(\mathsf{RSP}_1,\mathsf{RSP}_2,\mathsf{RSP}_3)$ *corresponding to each challenge* $\{1,2,3\}$*, outputs a valid witness for the relation* $\mathcal{R}_1$.

Proof. The proof follows standard arguments for Stern-like protocols, and can be found in Appendix B.

5.3 Specification of VLR-HABS

We now give high-level description of our lattice-based VLR-HABS scheme and present the formal algorithms in Figures 5 to 7.

Setup: The setup algorithm generates the public parameters and is executed by a trusted party to initiate the scheme, it begins by setting a parameter d where 2^d will be the maximum number of AAs and Users in the scheme. It samples uniformly random matrices $\{\mathbf{A}_i^{(b)}\}_{j=0}^{ld} \hookleftarrow \mathbb{Z}_q^{n_1\times m_1}$, $\{\mathbf{R}_j^{(b)}\}_{j=1}^{d} \hookleftarrow \mathbb{Z}_q^{m_3\times k_3}$ $(b\in\{0,1\})$ that generate the public keys and revocation tokens respectively. Two further matrices are computed $(\mathbf{A},\mathbf{R})$ with corresponding trapdoors $(\mathbf{T_A},\mathbf{T_R})$ according to $\mathsf{GenBasis}(n_1,m_1,q_1)$ and $\mathsf{GenBasis}(n_3,m_3,q_3)$, respectively. The key pair $(\mathsf{sk}_{\mathsf{RA}},\mathsf{pk}_{\mathsf{RA}}) := (\mathbf{T_A},\mathbf{A})$ is that of RA and $(\mathsf{sk}_{\mathsf{RevA}},\mathsf{pk}_{\mathsf{RevA}}) := (\mathbf{R},\mathbf{T_R})$ is for the RevA. The TA also generates its key pair for IBE-GPV as $(\mathsf{sk}_{\mathsf{TA}},\mathsf{pk}_{\mathsf{TA}}) := (\mathbf{T_D},\mathbf{D}) \leftarrow \mathsf{GenBasis}(n_2,m_2,q_2)$. Finally, it samples a vector $\mathbf{u}\in\mathbb{Z}_q^{n_1}$ that is used in the key-issuing phase of the scheme, and defines the attribute universe as $\mathbb{A}=\{\mathbf{a}_i\}_{i=1}^{N}$, for N total attributes. It outputs these under public parameters $\mathsf{pp} := (\mathbf{A},\mathbf{R},\mathbf{D},\mathbf{u},\{\mathbf{A}_i^{(b)}\}_{j=1}^{ld},\{\mathbf{R}_i^{(b)}\}_{j=1}^{d},\mathbb{A})$, which will be an implicit input to all algorithms.

UKGen: To join the scheme, the RA selects an identity id as a binary string of length d. It computes the corresponding public key $\mathsf{upk} := \mathbf{A}_{\mathsf{uid}}$ as $[\mathbf{A}||\mathbf{A}_1^{\mathsf{id}[1]}||...||\mathbf{A}_d^{\mathsf{id}[d]}]$. It computes $\mathbf{T}_{\mathbf{A}_{\mathsf{uid}}} \leftarrow$ $\mathsf{RandBasis}(\mathsf{ExtBasis}(\mathbf{T_A},\mathbf{A}_{\mathsf{uid}}),\mathbf{u},\beta_1)$, and then computes the user signing key as $\mathsf{usk}=\mathbf{z}_0\leftarrow\mathsf{SampleD}(\mathbf{T}_{\mathbf{A}_{\mathsf{uid}}},\mathbf{A}_{\mathsf{uid}},\mathbf{u},\beta_1)$, which satisfies $\mathbf{A}_{\mathsf{uid}}\mathbf{z}_0=\mathbf{u}$. The key pair is issued to the user.

AKGen: For an Authority joining the scheme, it is issued an identity $\mathsf{id}\in\{0,1\}^d$. The public and secret keys for the authorities are issued during attribute delegation as they are dependent on both the attribute and position within the hierarchy.

$\mathsf{AttIssue}$: This algorithm takes as input an attribute $\mathsf{att}=\mathbf{a}$, an AA secret key ask_i, a public key apk_{i+1} for either an IA or a user, and warr containing a matrix $\mathbf{A}_i$ with corresponding trapdoor $\mathbf{T}_{\mathbf{A}_i}$. For the target vector (or syndrome), the attribute $\mathbf{a}$, a k^{th} level AA extends its public key $\mathbf{A}_{\mathsf{id}_1||...||\mathsf{id}_i} :=$ $[\mathbf{A}||\mathbf{A}_1^{\mathsf{id}_1[1]}||...||\mathbf{A}_{kd+1}^{\mathsf{id}_i[1]}||...||\mathbf{A}_{(k+1)d}^{\mathsf{id}_i[d]}]$ to a $k+1$-level entity by computing $\mathbf{A}'\leftarrow[\mathbf{A}_{(k+1)d+1}^{\mathsf{id}_j[1]}||...||\mathbf{A}_{(k+2)d}^{\mathsf{id}_j[d]}]$ and executing $\mathbf{T}_{\mathbf{A}_j,\mathsf{att}}\leftarrow\mathsf{RandBasis}(\mathsf{ExtBasis}(\mathbf{T}_{\mathbf{A}_{\mathsf{id}}},[\mathbf{A}_{\mathsf{id}}||\mathbf{A}']),\mathbf{a},\beta_1)$. If it is issuing to an authority, it sets $\mathsf{skd}_i\leftarrow\mathbf{T}_{\mathbf{A}_j,\mathsf{att}}$, if it is issuing to a user, then it first computes a Bonsai signature on $\mathbf{A}_j$ with respect to $\mathbf{a}$. That is, a short vector $\mathbf{z}\leftarrow\mathsf{SampleD}(\mathbf{T}_{\mathbf{A}_j,\mathsf{att}},\mathbf{A}_j,\mathbf{a},\beta_1)$ and sets $\mathsf{skd}_i=\mathbf{z}$. It appends $(\mathsf{id},\mathbf{A}_j,\mathbf{a},\mathsf{skd}_i)$ the (possibly empty) warrant and returns warr.

Revoke: To revoke an identity, the algorithm Revoke takes as input a user identity, id. Using its secret key, $\mathsf{sk}_{\mathsf{RevA}}=\mathbf{T_R}$, a trapdoor for $\mathbf{R}$, it sets $\mathbf{R}_{\mathsf{id}}=[\mathbf{R}||\mathbf{R}_1^{\mathsf{id}[1]}||...||\mathbf{R}_d^{\mathsf{id}[d]}]$ and then computes the revocation token $\mathbf{y}$ as a short vector that satisfies $\mathbf{R}_{\mathsf{id}}\mathbf{y}=\mathbf{0}$. It appends $\mathbf{y}$ to $\mathsf{RevokeList}$.

Sign: The signing algorithm takes as input a set of attributes $\{\mathbf{a}_i\}_{i=1}^{\delta}$ with associated $\{\mathbf{A}_{\mathsf{id}_i},\mathbf{T}_{\mathbf{A}_{\mathsf{id}_i}}=\mathbf{z}_i\}_{i=1}^{\delta}$, the user's secret key $\mathsf{usk}=\mathbf{z}_0$, a message m and a policy Ψ, and we denote the signer's ID as uid for clarity. Recall that for each attribute, the delegated keys are short vectors that solve $\mathbf{A}_{\mathsf{id}_i}\mathbf{z}_i=\mathbf{a}_i$, and similarly the public value $\mathbf{u}$ proves that the signer has knowledge of usk as it solves $\mathbf{A}_{\mathsf{uid}}\mathbf{z}_0=\mathbf{u}$. It then prepares its revocation tokens by computing an LWE instance as, for each $\mathsf{id}\in\mathsf{warr}$, $\mathbf{C}_{\mathsf{id}}=\mathbf{B}\mathbf{R}_{\mathsf{id}}+\mathbf{E}$ where $\mathbf{B}$ is a binary matrix sampled from $\mathbb{B}^{m_3\times n_3}$ and $\mathbf{E}$ is an error matrix sampled from χ. $\mathbf{B}$ must be a binary secret so that the norm is small and we can use the zero-knowledge protocol described in Section 4.1. It generates keys for a one-time signature as $(\mathsf{osk},\mathsf{ovk})$ and encrypts the identities of the AAs in the delegation path under the Identity-based Encryption Scheme (GPV-IBE) using ovk as a pseudo-identity. To do this, it samples $\mathbf{e}_1,\mathbf{e}_2 \leftarrow \chi$ and $\mathbf{s} \hookleftarrow \mathbb{Z}_q^n$ and computes the following: $\mathbf{f}_i^{(1)} = \mathbf{D}\mathbf{s} + \mathbf{e}_1, \mathbf{f}_i^{(2)} = \mathbf{N}^T\mathbf{s} + \mathbf{e}_2 + \lfloor q/2 \rfloor \mathsf{id}_i$ and sets $\mathbf{f}_i = [\mathbf{f}_i^{(1)} || \mathbf{f}_i^{(2)}]$, for each $i \in [1,\delta]$. It generates the zero-knowledge argument of knowledge π, described in Section 5.2 for the relation $\mathcal{R}_1$. In particular, ZKAoK ensures that for each identity that appears in a delegation path for an attribute also, it appears in a corresponding ciphertext and revocation token. This is done by showing that the vectors $\mathbf{t}_{i,\mathbf{z}}^{(j)}$ and matrices $\mathbf{T}_{\mathsf{id},\mathbf{B}}$ (as described in Section 5.2) belong to the sets $\mathsf{SecretExt}_\beta(d_i^{(1)})$ and $\overline{\mathsf{SecretExt}_\beta}(d_{\mathsf{id}}^{(1)})$, respectively for a delegation path i, identity id and where the vectors d_i are the message encrypted in the ciphertext. Using the Fiat-Shamir heuristic [4], the signer turns the interactive protocol into non-interactive and binds the message to the message, policy, revocation tokens $\mathbf{C} = \{\mathbf{C}_{\mathsf{id}}\}_{\mathsf{id} \in \mathsf{warr}}$, ciphertext $\mathbf{f} := \{\mathbf{f}_i\}_{i=1}^{\delta}$ and proof. It computes the challenge as:

$$\mathsf{CH} = \{Ch\}_{i=1}^{t} = \mathcal{H}_1(\mathsf{m}, \Psi, \mathbf{f}, \mathbf{C}, \mathsf{ovk}, \mathsf{pp}, \{\mathsf{CMT}_i\}_{i=1}^{t})$$

Finally, it computes a one-time signature over the proof π, ciphertext $\mathbf{f}$, message m and policy Ψ as σ_o. Since the choice of the OTS can be generic we leave the function here unspecified. However, for security and instantiation, we shall reuse the Bonsai signature scheme from [12], with use of a chameleon hash function $\mathcal{H}_2 : \mathbb{Z}_q^* \to \{0,1\}^{\tilde{m}}$ (see Section 4.2). It outputs the VLR-HABS signature: $\sigma = (\mathbf{f}, \mathbf{C}, \pi, \sigma_o, \mathsf{ovk})$.

Verify: To verify a candidate signature σ, a verifier obtains the list $\mathsf{RevokeList}$ from the RevA, potentially offline and before the signature is presented. It parses σ as $(\mathbf{f}, \mathbf{C}, \pi, \sigma_o, \mathsf{ovk})$ and checks that π and σ_o pass verification. It then computes $\mathbf{C}_{\mathsf{id}}\mathbf{y}$ and outputs 0 if any $\mathbf{C}_{\mathsf{id}}\mathbf{y} \leqslant n_3\beta_3^2$ for any $\mathbf{y} \in \mathsf{RevokeList}$, $\mathsf{id} \in \mathsf{warr}$.

Trace: On input of a candidate VLR-HABS signature, the tracing algorithm parses σ as $(\mathbf{f}, \mathbf{C}, \pi, \sigma_o, \mathsf{ovk})$. It first verifies σ, then, using its secret key, $\mathbf{T}_\mathbf{D}$ it can create an identity-dependent decryption key $\mathbf{S}_{\mathsf{ovk}}$ for a ciphertext $\mathbf{f}_{\mathsf{id}}$, with which it can extract the user ID and identities of the authorities that appear in the delegation path of any attribute. This algorithm outputs $\mathbf{S}_{\mathsf{ovk}}, \{\mathsf{id}_i\}_{i=1}^{\delta}$.

Judge: This algorithm is then able to verify the correctness of decryption of this IBE-GPV ciphertext. It takes as input the decryption key $\mathbf{S}_{\mathsf{ovk}}$ and checks that it is a valid key for $\mathbf{f}_i$, that is, it checks $\mathbf{D}\mathbf{S} \overset{?}{=} \mathcal{H}_0(\mathsf{ovk})$ and $||\mathbf{S}|| \leqslant \beta_2$. If this passes, it also checks the decryption is correct by evaluating $\mathbf{f} \overset{?}{=} \mathbf{P}^*\mathbf{e} + \mathbf{Q}\mathsf{id}$ and outputs 1 if all checks hold, else it outputs 0. By using a one-time identity in our IBE scheme, we are able to bypass expensive zero-knowledge proofs in this stage and instead only require the Trace algorithm it output a verifiable key for the "identity" ovk.

Detailed description. We provide the complete specification of our protocol only for the Setup and AKGen algorithms, and illustrate UKGen, $\mathsf{AttIssue}$, Revoke, Sign, Verify, Trace, Judge in Figures 5 to 7.

– $\mathsf{Setup}(\lambda)$. It generates the Root Authority and Revocation Authority key-pairs as $(\mathsf{skd}_0, \mathsf{pkd}_0) := (\mathbf{T}_\mathbf{A}, \mathbf{A}) \leftarrow \mathsf{GenBasis}(n_1, m_1, q_1)$, and$(\mathsf{sk}_{\mathrm{RevA}}, \mathsf{pk}_{\mathrm{RevA}}) := (\mathbf{T}_\mathbf{R}, \mathbf{R}) \leftarrow \mathsf{GenBasis}(n_3, m_3, q_3)$. Next it samples random matrices $\{\mathbf{A}_i^b\}_{i=1}^{ld} \hookleftarrow \mathbb{Z}_{q_1}^{n_1 \times m_1}$ and $\{\mathbf{R}_i\}_{i=1}^{d} \hookleftarrow \mathbb{Z}_{q_3}^{n_3 \times m_3}$. During this phase, the TA keys are also computed as $(\mathsf{skd}_{TA}, \mathsf{pkd}_{TA}) = (\mathbf{T}_\mathbf{D}, \mathbf{D}) \leftarrow \mathsf{GenBasis}(n_2, m_2, q_2)$. Define χ_2 be a β_2 bounded distribution $\mathcal{D}_{\mathbb{Z},\alpha_2}$, and similarly let $\chi_3 = \mathcal{D}_{\mathbb{Z},\alpha_3}$ (i.e. bounded by β_3).
– AKGen. Sample and output $\mathsf{id} \hookleftarrow \{0,1\}^d$.

6 Security, Efficiency and Extensions

In this section, we provide analysis of correctness and security followed by efficiency considerations and parameter selection for our VLR-HABS scheme. We start by stating and proving two lemmata that we will use in the analysis of our scheme.

[4] As in [4], we choose to present the FS heuristic for simplicity. We note, however, that one could instantiate our scheme with the Unruh transform of [55] to achieve security in the quantum random oracle model (QROM). This follows from the fact that the transform is generic, and applies to any Sigma protocol that has the standard properties of Honest Verifier Zero-Knowledge and Special Soundness.

```
UKGen (ask_0)
0: Sample $\mathsf{uid} \hookleftarrow \{0,1\}^d$
1: $\mathbf{A}_{\mathsf{uid}} = [\mathbf{A} || \mathbf{A}_1^{\mathsf{id}[1]} || \ldots || \mathbf{A}_d^{\mathsf{id}[d]}]$
2: $\mathbf{T}_{\mathbf{A}^*} \leftarrow \mathsf{ExtBasis}(\mathbf{T}_{\mathbf{A}}, \mathbf{A}_{\mathsf{uid}})$
3: $\mathbf{z}_{\mathsf{uid}} \leftarrow \mathsf{SampleD}(\mathbf{T}'_{\mathbf{A}^*}, \mathbf{A}_{\mathsf{uid}}, \mathbf{u}, \beta_1)$
4: return $(\mathsf{id}, \mathsf{skd}_{\mathsf{id}}, \mathsf{pkd}_{\mathsf{id}}) = (\mathsf{uid}, \mathbf{z}_{\mathsf{uid}}, \mathbf{A}_{\mathsf{uid}})$
```

```
AttIssue ((warr_i),att,[pkd_j|upk])
0: Parse $(\mathsf{skd}_{i,\mathsf{att}}, \mathsf{pkd}_j)$ as $((\mathbf{T}_{\mathbf{A}_i,\mathbf{a}}, \mathbf{A}_i, \mathsf{id}_i), \mathbf{A}_j, \mathsf{id}_j)$
1: $\mathbf{A}' \leftarrow [\mathbf{A}_{(k+1)d+1}^{\mathsf{id}_j[1]} || \ldots || \mathbf{A}_{(k+2)d}^{\mathsf{id}_j[d]}]$
2: $\mathbf{T}'_{\mathbf{A}^*} \leftarrow \mathsf{RandBasis}(\mathsf{ExtBasis}(\mathbf{T}_{\mathbf{A}_{\mathsf{id}}}, [\mathbf{A}_{\mathsf{id}} || \mathbf{A}']), \mathbf{a}, \beta_1)$
3: if $|\mathsf{id}_j| = ld$ then
4:     $\mathbf{z}_j \leftarrow \mathsf{SampleD}(\mathbf{T}'_{\mathbf{A}^*}, [\mathbf{A}_i || \mathbf{A}_j], \mathbf{a}, \beta_1)$
5:     $\mathsf{skd}_{j,\mathsf{att}} \leftarrow \mathbf{z}_j$
6: else ,  $\mathsf{skd}_{j,\mathsf{att}} \leftarrow \mathbf{T}_{\mathbf{A}^*}$
7: return $\mathsf{warr} = \mathsf{warr}_i \cup \{\mathsf{id}_j, \mathbf{A}_j, \mathsf{skd}_{j,\mathsf{att}}, \mathbf{a}\}$
```

Fig. 5. Algorithms UKGen and AttIssue of our VLR-HABS construction.

```
Sign ((usk,warr),pkd_id,m,Ψ)
0: Parse warr as $\{\mathsf{id}_{i,j}, \mathbf{A}_{i,j}, \mathbf{z}_i, \mathbf{a}_i\}_{i \in [1,\delta], j \in [1,l]}$ with $|\Psi| = \delta$
1: $(\mathsf{ovk}, \mathsf{osk}) \leftarrow \mathsf{OTS.KGen}(\lambda)$
2: $\mathbf{N} := \mathcal{H}_0(\mathsf{ovk}), \mathbf{s} \hookleftarrow \mathbb{Z}_q^n, \mathbf{e}_1 \hookleftarrow \chi_2, \mathbf{e}_2 \hookleftarrow \chi_2$
3: foreach $i \in [1,\delta]$
4:     $\mathbf{f}_i^{(1)} = \mathbf{D}^T\mathbf{s} + \mathbf{e}_1, \mathbf{f}_i^{(2)} = \mathbf{N}^T\mathbf{s} + \mathbf{e}_2 + \lfloor q/2 \rfloor \mathsf{id}_i$
5:     $\mathbf{f}_i = [\mathbf{f}_i^{(1)} || \mathbf{f}_i^{(2)}]$
6: foreach $i \in \mathsf{warr}$
7:     $\mathbf{R}_i \leftarrow [\mathbf{R} || \mathbf{R}_1^{\mathsf{id}[1]} || \ldots || \mathbf{R}_d^{\mathsf{id}[d]}]$
8:     $\mathbf{B}_i \hookleftarrow \mathbb{B}^{k_2 \times (d+1)n_2}, \mathbf{E}_i \hookleftarrow \chi_3^m, \mathbf{C}_i \leftarrow \mathbf{B}_i\mathbf{R}_i + \mathbf{E}_i$
9: $\mathbf{f} := \{\mathbf{f}_i\}_{i=1}^{\delta}, \mathbf{C} := \{\mathbf{C}_i\}_{i \in \mathsf{warr}}$
10: $\pi := (\{\mathsf{CMT}_i, \mathsf{RSP}_i, \mathsf{CH}_i\}_{i=1}^t) \leftarrow \mathsf{ZKAoK}(\mathsf{uid}, \mathbf{z}_0, \{\mathsf{id}_i, \mathbf{B}_i, \mathbf{E}_i\}_{i \in \mathsf{warr}},$
        $\{\mathbf{e}_i, \mathbf{z}_i, \mathbf{a}_i\}_{i=1}^{\delta}, (\mathbf{C}, \mathbf{Q}, \mathbf{P}, \mathbf{f}, \mathsf{ovk}, \Psi, \mathsf{pp}), \mathcal{R}_1)$
11: $\sigma_o \leftarrow \mathsf{OTS.Sign}(\mathsf{osk}, \mathcal{H}_2(\mathsf{m}, \Psi, \pi, \mathbf{f}, \mathbf{C}))$
12: return $\sigma \leftarrow (\pi, \sigma_o, \mathsf{ovk}, \mathbf{f}, \mathbf{C})$
```

```
Verify (pkd_0,σ, RevokeList)
0: Parse $\sigma$ as $(\pi, \sigma_o, \mathsf{ovk}, \{\mathbf{f}_i\}_{i=1}^{\delta}, \{\mathbf{C}_i\}_{i \in \mathsf{warr}})$
1: if $\exists \mathbf{y} \in \mathsf{RevokeList}$ and $\exists \mathbf{C}_i : i \in \mathsf{warr}$ s.t. $\mathbf{C}_i\mathbf{y} \leqslant n_3\beta_3^2$, return 0
2: if $\mathsf{ZKAoK.Verify}(\pi, \Psi, \mathsf{m}, \{\mathbf{C}_i\}_{i=1}^{l\delta}) \neq 1$, return 0
3: if $\mathsf{OTS.Verify}(\mathsf{ovk}, \sigma_o, \mathcal{H}_2(\mathsf{m}, \Psi, \pi, \mathbf{f}, \mathbf{C})) \neq 1$, return 0
4: else return 1
```

Fig. 6. Algorithms Sign and Verify of our VLR-HABS construction.

Lemma 1. *Let $\beta = \mathsf{poly}(n)$, $q \geqslant (2n\beta^2+1)^2$ and $m \geqslant 2n$, then for a fixed $\mathbf{y} \in \mathbb{Z}_q^m$ with $||\mathbf{y}||_\infty \leqslant \beta$, and a uniformly random matrix $\mathbf{C} \hookleftarrow \mathbb{Z}_q^{k \times m}$, we have*

$$\Pr[||\mathbf{C}\mathbf{y}||_\infty \leqslant n\beta^2] \leqslant \mathsf{negl}(n)$$

Proof. Using the fact that $\mathbf{C}\mathbf{y}$ is uniform in $\mathbb{Z}_q^n$, we have:

$$\Pr[||\mathbf{C}\mathbf{y}||_\infty \leqslant n\beta^2] \leqslant \frac{(2n\beta^2+1)^m}{q^m} \leqslant \frac{1}{(2n\beta^2+1)^m} \leqslant (2n\beta^2+1)^{-n} \leqslant \mathsf{negl}(n)$$

In our scheme, as $\mathbf{C}$ is an LWE sample, it will only be statistically close to uniform, so one should account for a $\mathsf{negl}(n)$ in the first inequality, which carries through and is absorbed in the final bound.

```
Revoke ($\mathbf{T_R}$,(id,RevokeList))
0: $\mathbf{R}_{\mathsf{id}} \leftarrow [\mathbf{R}||\mathbf{R}_1^{\mathsf{id}[1]}||...||\mathbf{R}_d^{\mathsf{id}[d]}]$
1: $\mathbf{T}_{\mathbf{R}^*} \leftarrow \mathsf{ExtBasis}(\mathbf{T_R},\mathbf{R}_{\mathsf{id}})$
2: $\mathbf{y} \leftarrow \mathsf{SampleD}(\mathbf{T}'_{\mathbf{R}^*},\mathbf{R}^*,\mathbf{0},\beta_3)$
3: return $\mathsf{RevokeList} = \mathsf{RevokeList} \cup \{\mathbf{y}\}$
```

```
Trace ($\sigma$,$\mathbf{T_D}$,RevokeList)
0: Parse $\sigma$ as $(\pi,\Psi,\mathsf{m},\{\mathbf{f}_i\}_{i=1}^{\delta},\{\mathbf{C}_i\}_{i=1}^{l\delta})$
1: if $\mathsf{Verify}(\mathbf{A}_0,(\pi,\sigma_o,\mathsf{ovk},\{\mathbf{f}_i\}_{i=1}^{\delta},\{\mathbf{C}_i\}_{i=1}^{l\delta}),\mathsf{RevokeList}) = 1$ then
2:    $\mathbf{S}_{\mathsf{ovk}} \leftarrow \mathsf{SampleD}(\mathbf{T_D},\mathbf{D},\mathcal{H}_0(\mathsf{ovk}),\beta_2)$
3:    for $i \in [1,...,\delta]$:
4:       Parse $\mathbf{f}_i = [\mathbf{f}_i^{(1)}||\mathbf{f}_i^{(2)}]$
5:       $[\mathsf{id}_1||...\mathsf{id}_{l-1}||\mathsf{uid}] = \lfloor \mathbf{f}_i^{(1)} - \mathbf{S}_{\mathsf{ovk}} \cdot \mathbf{f}_i^{(2)} \rceil$
6:       $\mathsf{warr} = \mathsf{warr} \cup \{\mathsf{id}_1,...,\mathsf{uid}\}$
7:    return $(\mathsf{warr},\mathbf{S}_{\mathsf{ovk}})$
8: else return $\perp$
```

```
Judge ($\sigma$,warr,$\mathbf{S}_{\mathsf{ovk}}$,RevokeList)
0: Parse $\sigma$ as $(\pi,\Psi,\mathsf{m},\{[\mathbf{f}_1^{(i)}||\mathbf{f}_2^{(i)}]^T\}_{i=1}^{\delta},\{\mathbf{C}_i\}_{i=1}^{l\delta})$
1: if $\lfloor \mathbf{f}_2^{(i)} - \mathbf{S}_{\mathsf{ovk}} \cdot \mathbf{f}_1^{(i)} \rceil \neq \mathsf{warr}[i]$ for any $i \in [1,..,\delta]$, return 0
2: elseif $\mathbf{D}\mathbf{S}_{\mathsf{ovk}} \neq \mathcal{H}_0(\mathsf{ovk})$, return 0
3: else return 1
```

Fig. 7. Algorithms Revoke, Trace and Judge of our VLR-HABS construction.

Lemma 2. *Let $\beta = \mathsf{poly}(n)$, then for $(\mathbf{R},\mathbf{B},\mathbf{C},\mathbf{E},\mathbf{y}) \in \mathbb{Z}_q^{m\times k} \times \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^{n\times k} \times \mathbb{Z}_q^{n\times k} \times \mathbb{Z}_q^{k}$ such that $\mathbf{Ry} = \mathbf{0}$ with $\|\mathbf{y}\|_\infty \leqslant \beta$ and $\mathbf{C} = \mathbf{BR} + \mathbf{E}$, where $\mathbf{B},\mathbf{R}$ are uniformly random and $\mathbf{E}$ is drawn from β-bounded distribution χ over $\mathbb{Z}_q^n$, then*

$$\Pr[\|\mathbf{Cy}\|_\infty \leqslant n\beta^2] = 1$$

Proof. Expanding out the computation, we have:

$$\mathbf{Cy} = (\mathbf{BR}+\mathbf{E})\mathbf{y} = \mathbf{BRy} + \mathbf{Ey} = \mathbf{B}\cdot\mathbf{0}^{m\times k} + \mathbf{Ey} = \mathbf{Ey}$$

Noting that $\|\mathbf{E}\|_\infty \leqslant \beta$ and $\|\mathbf{y}\|_\infty \leqslant \beta$, we derive $\|\mathbf{Ey}\|_\infty \leqslant n\beta^2$ from the Cauchy-Schwarz Inequality.

Theorem 1. *Our* VLR-HABS *construction given in Figures 5 and 6 and* **??** *is correct.*

Proof. See Appendix C.

Theorem 2. *Let* COM *be a statistically hiding and computationally binding string commitment scheme. Then our* VLR-HABS *construction given in Figures 5 to 7 offers Path Anonymity, Non-frameability and Path Traceability in the Random Oracle Model if the* $\mathsf{LWE}_{n_2,m_2,q_2,\chi_2}$,$\mathsf{LWE}_{n_3,m_3,q_3,\chi_3}$ *and* $\mathsf{SIS}_{n_5,m_5,q_5,\beta_5}$,$\mathsf{SIS}_{n_1,m_1,q_1,\beta_1}$ *and* $\mathsf{SIS}_{n_4,m_4,q_4,\beta_4}$ *assumptions hold,* $\mathcal{H}_0$ *and* $\mathcal{H}_2$ *are collision resistant and* $\mathcal{H}_1$ *is a random oracle.*

Proof. The proof follows from Lemmas 3 to 5 found in Appendix D.

6.1 Efficiency and Parameters

We instantiate the scheme with the parameter choices given in Table 1 so that it runs in polynomial time and such that the security and correctness properties hold. We used the estimator by Albrecht et al. [2]

to evaluate the estimated security of each LWE and SIS instance. For the values relating to the Bonsai signature, GPV-IBE, OTS and KTX commitment scheme COM, we directly use conditions as given in their original works. For the ZKAoK we use the parameters of the underlying commitment scheme and use a soundness parameter $t = \omega(\lambda)$. The values for n_i are assumed to be fixed and are typically a small polynomials in λ. The VLR mechanism uses values from the trapdoor delegation in [12] restricted to the conditions of Lemmas 1 and 2. We note that if one relaxes the perfect correctness given by Lemma 2, one could obtain smaller values for q_3 at the trade off of introducing a correctness error. According to Lindner and Peikert [33], this error is close to an exponential $\exp(-\pi(\beta_3\alpha_3)^2)$, where α is the Gaussian parameter for the LWE instance of $\mathbf{C}$. We briefly discuss the key sizes and algorithm complexity with respect to the security parameters n_i. Public keys are elements of $\mathbb{Z}_q^{n_1 \times m_1}$ which is quadratic in n_1. The signing operation takes $t \cdot \mathcal{O}(|\mathsf{warr}|(n_1^2 + n_2^2 + n_3^2) + n_4^2)$ steps, where t is the soundness parameter for ZKAoK, and the length of produced signatures is also quadratic in the parameter n_1. Signature verification also runs in this asymptotic bound. The revocation check that completes the verification algorithm is linear in the number of revoked users, which matches other VLR schemes such as [30] where they note this complexity for VLR seems unavoidable. The Trace and Judge algorithms are linear in $|\mathsf{warr}|$.

We briefly note some final generic changes to improve upon efficiency. Firstly, the protocol benefits from the pre-computation of offline/online signatures [52] that are naturally compatible with our one-time signature. Here, the OTS signature is produced ahead of time (potentially batched), using a chameleon hash function for $\mathcal{H}_3$, that would allow the signer to find a corresponding randomness to match the message it must sign when generating the VLR-HABS signature. Secondly, as noted in [49], commitments in ZKAoK can be hashed prior to sending to minimise size of the signature, at the expense of additional hash computations by the signer and verifier. Thirdly, our delegation techniques consist of delegating short basis have order $\mathcal{O}(n^2)$, per issuance. More efficient trapdoor delegations exist, for example the lattice trapdoor by Micciancio and Peikert [42], however it is not clear how one might argue security for VLR-HABS as the structure of the resulting Boyen signature does not lend itself to be embedded over multiple delegations. Finding a more efficient yet compatible trapdoor could be viewed as an interesting open problem. Finally, we note using complexity assumptions and tools for ideal lattices [39] instead of integral lattices reduce most of their associated operations by about a linear factor in the security parameter. The practical suitability of the resulting schemes may still depend on careful selection of parameters, nonetheless the techniques we have used in this paper can be realised from such structures and do provide generic overall improvements.

Table 1. Parameter Selection for VLR-HABS based on its building blocks. We target 128-bit security, and so set the soundness parameter $t = 219$. We set ID bit length to be $d = 16$, which allows the schemes to support 65536 unique entities across a hierarchy of depth $l = 3$. Note here that m_i' are the number of samples in LWE and SIS challenges.

Building Blocks	i	n_i	m_i'	m_i	q_i	β_i	α_i
Bonsai Signature	1	500	618	9840	2^{21}	31440	—
GPV-IBE	2	400	—	16800	2^{16}	—	10^{-4}
VLR	3	1400	1840	29440	2^{62}	—	10^{-3}
OTS	4	500	41	656	2^{21}	31440	—
COM	5	400	—	25600	2^{16}	3200	—

6.2 Revoking Attributes

We now briefly describe how to achieve attribute revocation for our VLR-HABS scheme. We immediately observe that we can apply similar techniques to those used to revoke users. In particular, the signer, upon generating a VLR-HABS signature, also commits to the attributes used as an LWE instance $\mathbf{C} = \mathbf{B}\tilde{\mathbf{A}} + \mathbf{E}$, where $\tilde{\mathbf{A}}$ is the binary decomposition of an attribute concatenated with $\mathbf{R}$, $[\mathbf{R}||\mathbf{a}]$. Revocation is then performed by the authority by computing a short vector such that $\mathbf{C}\mathbf{y} \leqslant n\beta^2$. The argument ZKAoK would have to be modified to show that the attributes are correctly committed to. Minor modifications to the security properties path anonymity, non-frameability and path traceability are required. In particular, in path anonymity we prevent the adversary from revoking an attribute used in the challenge signature, as this would allow it to trivially break path anonymity. This can be achieved by standard bookkeeping techniques and the proofs follow a similar strategy.

This idea could be further extended to revoking an attribute for a specific user. By altering the delegation process to delegate an attribute of the form $\mathsf{id}||\mathsf{att}$ (the bit-string concatenation) and further requiring the ZKAoK to link id to the identity used in $\mathbf{A}_{\mathsf{id}}$ and $\mathbf{R}_{\mathsf{id}}$. This could be achieved using the techniques already used in Section 4.1. We finally note that this would require the policy to be encoded in a specific format that, intuitively, ignores the first d bits of the attribute $\mathsf{id}||\mathsf{att}$.

7 Conclusion

The VLR-HABS scheme proposed in this paper improves upon security and functionality of existing HABS constructions by proposing a lattice-based scheme which supports verifier-local revocation and a wider range of signing policies. Our scheme is based on LWE and SIS assumptions which are believed to offer post-quantum security. It supports inner-product relations which allow for conjunctive, disjunctive and threshold policies as well as polynomial evaluations of attributes. Revocation in our HABS schemes uses a novel VLR mechanism that allows revocation of signers, attributes as well as intermediate authorities. Our scheme also implies the first lattice-based (non-hierarchical) ABS scheme with these properties.

References

1. M. Ajtai. Generating hard instances of lattice problems. STOC '96. ACM, 1996.
2. M. R. Albrecht, R. Player, and S. Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9:169 – 203, 2015.
3. G. Ateniese, D. Song, and G. Tsudik. Quasi-efficient revocation of group signatures. In *Proceedings of the 6th International Conference on Financial Cryptography*, FC'02, 2002.
4. R. E. Bansarkhani and A. E. Kaafarani. Post-quantum attribute-based signatures from lattice assumptions. Cryptology ePrint Archive, Report 2016/823, 2016. `https://eprint.iacr.org/2016/823`.
5. C. Baum, H. Lin, and S. Oechsner. Towards practical lattice-based one-time linkable ring signatures. In *Information and Communications Security*, 2018.
6. M. Bellare and G. Fuchsbauer. Policy-based signatures. In *PKC 2014*, pages 520–537, 2014.
7. D. Boneh and H. Shacham. Group signatures with verifier-local revocation. CCS '04. ACM, 2004.
8. E. Brickell. An effcient protocol for anonymously providing assurance of the container of a private key. 2003.
9. J. Bringer and A. Patey. Backward unlinkability for a vlr group signature scheme with efficient revocation check. In *SECRYPT*, volume 2012, 2011.
10. J. Camenisch, G. Neven, and M. Rückert. Fully anonymous attribute tokens from lattices. In *Security and Cryptography for Networks*, pages 57–75. Springer Berlin Heidelberg, 2012.
11. R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In *EUROCRYPT 2004*, 2004.
12. D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. *J. Cryptol.*, 25(4):601–639, Oct. 2012.
13. P.-L. Cayrel, R. Lindner, M. Rückert, and R. Silva. A lattice-based threshold ring signature scheme. In M. Abdalla and P. S. L. M. Barreto, editors, *LATINCRYPT*, 2010.
14. C.-K. Chu, J. K. Liu, X. Huang, and J. Zhou. Verifier-local revocation group signatures with time-bound keys. In *ASIACCS*. ACM, 2012.
15. C.-C. Drăgan, D. Gardham, and M. Manulis. Hierarchical attribute-based signatures. In *Cryptology and Network Security*, 2018.
16. L. Ducas and D. Micciancio. Improved short lattice signatures in the standard model. In *CRYPTO 2014*, 2014.
17. A. El Kaafarani and E. Ghadafi. Attribute-based signatures with user-controlled linkability without random oracles. In *Cryptog. and Coding*, pages 161–184, 2017.
18. A. El Kaafarani, E. Ghadafi, and D. Khader. Decentralized traceable attribute-based signatures. In *CT-RSA 2014*, pages 327–348, 2014.
19. A. El Kaafarani and S. Katsumata. Attribute-based signatures for unbounded circuits in the rom and efficient instantiations from lattices. In *Public-Key Cryptography*, pages 89–119, 2018.
20. A. Escala, J. Herranz, and P. Morillo. Revocable attribute-based signatures with adaptive security in the standard model. In *AFRICACRYPT 2011*, 2011.
21. D. Gardham and M. Manulis. Hierarchical attribute-based signatures: Short keys and optimal signature length. In *Applied Cryptography and Network Security*, 2019.

22. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *ACM Symposium on Theory of Computing*, STOC '08, page 197–206, 2008.
23. 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.
24. S. D. Gordon, J. Katz, and V. Vaikuntanathan. A group signature scheme from lattice assumptions. In M. Abe, editor, *ASIACRYPT*, 2010.
25. J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *Advances in Cryptology – EUROCRYPT 2008*, 2008.
26. J. Herranz. Attribute-based signatures from RSA. *Theor. Comput. Sci.*, 527:73–82, Mar. 2014.
27. J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, 2008.
28. A. Kawachi, K. Tanaka, and K. Xagawa. Concurrently Secure Identification Schemes Based on the Worst-Case Hardness of Lattice Problems. In *AsiaCrypt2008*.
29. H. Krawczyk and T. Rabin. Chameleon signatures. In *NDSS*, 2000.
30. A. Langlois, S. Ling, K. Nguyen, and H. Wang. Lattice-based group signature scheme with verifier-local revocation. In *Public-Key Cryptography – PKC 2014*, 2014.
31. 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.
32. B. Libert and D. Vergnaud. Group signatures with verifier-local revocation and backward unlinkability in the standard model. In *CANS*, 2009.
33. R. Lindner and C. Peikert. Better key sizes (and attacks) for lwe-based encryption. In *CT-RSA'11*.
34. S. Ling, K. Nguyen, D. Stehlé, and H. Wang. Improved zero-knowledge proofs of knowledge for the isis problem, and applications. In *PKC'13*.
35. S. Ling, K. Nguyen, and H. Wang. Group signatures from lattices: Simpler, tighter, shorter, ring-based. In *PKC*, 2015.
36. S. Ling, K. Nguyen, H. Wang, and Y. Xu. Lattice-based group signatures: Achieving full dynamicity with ease. In *ACNS*, 2017.
37. S. Ling, K. Nguyen, H. Wang, and Y. Xu. Constant-size group signatures from lattices. In M. Abdalla and R. Dahab, editors, *PKC*, 2018.
38. S. Ling, K. Nguyen, H. Wang, and Y. Xu. Accountable tracing signatures from lattices. In M. Matsui, editor, *CT-RSA 2019*, 2019.
39. V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, 2010.
40. H. K. Maji, M. Prabhakaran, and M. Rosulek. Attribute-based signatures. In *CT-RSA'11*.
41. D. Micciancio. On the hardness of learning with errors with binary secrets. *ToC'14*.
42. D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, pages 700–718, 2012.
43. D. Micciancio and C. Peikert. Hardness of sis and lwe with small parameters. In *CRYPTO*, 2013.
44. T. Nakanishi and N. Funabiki. Verifier-local revocation group signature schemes with backward unlinkability from bilinear maps. In *ASIACRYPT 2005*.
45. 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.
46. T. Okamoto and K. Takashima. Decentralized attribute-based signatures. In *PKC'13*, 2013.
47. C. Peikert. A decade of lattice cryptography. *Found. Trends Theor. Comput. Sci.*, 10(4), 2016.
48. D. Pointcheval and S. Vaudenay. On provable security for digital signature algorithms. 11 1996.
49. G. Poupard and J. Stern. Short proofs of knowledge for factoring. In *PKC*, 2000.
50. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. STOC '05. ACM, 2005.
51. Y. Sakai, N. Attrapadung, and G. Hanaoka. Practical attribute-based signature schemes for circuits from bilinear map. *IET Information Security*, pages 184–193(9), 2018.
52. A. Shamir and Y. Tauman. Improved online/offline signature schemes. In *CRYPTO'01*.
53. Q. Su, R. Zhang, R. Xue, and P. Li. Revocable attribute-based signature for blockchain-based healthcare system. *IEEE Access*, 8:127884–127896, 2020.
54. R. Tsabary. An equivalence between attribute-based signatures and homomorphic signatures, and new constructions for both. In *TCC (2)'17*, pages 489–518, 2017.
55. D. Unruh. Non-interactive zero-knowledge proofs in the quantum random oracle model. In *EUROCRYPT*, pages 755–784, 2015.
56. Q. Wang, S. Chen, and A. Ge. A new lattice-based threshold attribute-based signature scheme. In *Information Security Practice and Experience*, 2015.
57. Y. Zhang, X. Liu, Y. Hu, Q. Zhang, and H. Jia. Attribute-based signatures for inner-product predicate from lattices. In *Cyberspace Safety and Security*, 2019.
58. S. Zhou and D. Lin. Shorter verifier-local revocation group signatures from bilinear maps. In *Cryptology and Network Security*, 2006.

Appendices

A Zero-Knowledge Protocol

The protocol is as follows:

- Public parameters: $\mathbf{A}, \mathbf{R}, \{\mathbf{A}_i^b\}_{i=1}^{ld}, \{\mathbf{R}_i^b\}_{i=1}^{d}, \mathbf{G}^*, \mathbf{P}^*, \mathbf{Q}, \{\mathbf{C}_{\mathsf{id}}\}_{\mathsf{id} \in \mathsf{warr}}, \{\mathbf{f}_i\}_{i=1}^{\delta}, \mathbf{p}, \mathbf{u}$.
- The prover's witness are the vectors $\mathsf{uid}, \mathbf{z}_0, \{\mathbf{z}_i, \mathbf{a}_i, \mathbf{e}_i\}_{i=1}^{\delta}, \{\mathsf{id}\}_{\mathsf{id} \in \mathsf{warr}}$ and the matrices $\{\mathbf{B}_{\mathsf{id}}, \mathbf{E}_{\mathsf{id}}\}_{\mathsf{id} \in \mathsf{warr}}$.

The goal is prove the following relation:

$$\mathcal{R}_1 := \begin{cases} \mathbf{A}_{[\mathsf{id}_1||\ldots||\mathsf{id}_l]}(\mathbf{z}_i) = \mathbf{a}_i \mod q_1, i \in [1,\delta] \text{ and } \mathsf{id}_j \in \mathsf{warr}[\mathbf{a}_i] \\ \mathbf{C}_{\mathsf{id}} = \mathbf{B}\mathbf{R}_{\mathsf{id}} + \mathbf{E} \mod q_3 \text{ for } \mathsf{id} \in \mathsf{warr} \\ \mathbf{f}_{\mathsf{id}} = \mathbf{P}^*\mathbf{e} + \mathbf{Q}[\mathsf{id}_1||\ldots||\mathsf{uid}] \mod q_2 \text{ for every } \mathsf{id} \in \mathsf{warr}[\mathbf{a}_i],\ i \in [1,\delta] \\ \langle \mathbf{a}, \mathbf{p} \rangle = 1 \text{ where } \mathbf{a} = [\mathbf{a}_1||\ldots||\mathbf{a}_\delta] \wedge \mathbf{A}_{\mathsf{uid}}(\mathbf{z}_0) = \mathbf{u} \mod q_1 \end{cases}$$

Prior to the interaction, the prover P and the verifier V both extend the matrix $\mathbf{A}' := [\mathbf{A}||\mathbf{A}_1^{(0)}||\mathbf{A}_1^{(1)}||\ldots||\mathbf{A}_{ld}^{(0)}||\mathbf{A}_{ld}^{(1)}]$ to $\mathbf{A}^* \leftarrow \mathsf{MatExt}(\mathbf{A}')$ and similarly $\mathbf{R}' := [\mathbf{R}||\mathbf{R}_1^{(0)}||\mathbf{R}_1^{(1)}||\ldots||\mathbf{R}_d^{(0)}||\mathbf{R}_d^{(1)}]$ to $\mathbf{R}^* \leftarrow \mathsf{MatExt}(\mathbf{R})$. Both parties also compute $\mathbf{G}^*$ which is the gadget matrix defined in Section 4.4, and also $\mathbf{G}'$ which is also a gadget matrix but of size $n_1 \times n_1 \log q_1$. The prover then prepares its witness vectors with the Decomposition-Extension techniques for the values $p_1 = \log_2(\beta_1), p_2 = \log_2(\beta_2)$ and $p_3 = \log_2(\beta_3)$. That is,

$$\begin{aligned} &\mathsf{id}^* \leftarrow \mathsf{VecExt}_2(\mathsf{id}), \forall \mathsf{id} \in \mathsf{warr} \\ &\{\mathbf{z}_{0,1}, \ldots, \mathbf{z}_{0,p_1}\} \leftarrow \mathsf{VecDE}(\mathbf{z}_0, [\mathsf{uid}^*]) \\ &\{\mathbf{z}_{i,1}, \ldots, \mathbf{z}_{i,p_1}\} \leftarrow \mathsf{VecDE}(\mathbf{z}_i, [\mathsf{id}_1^*||\ldots||\mathsf{id}_{l-1}^*||\mathsf{uid}^*]), \forall i \in [1,\delta] \\ &\{\mathbf{e}_{i,1}, \ldots, \mathbf{e}_{i,p_2}\} \leftarrow \mathsf{VecDec}(\mathbf{e}_i), \forall i \in [1,\delta] \\ &\mathbf{B}_{\mathsf{id}} \leftarrow \mathsf{MatDE}(\mathbf{B}_{\mathsf{id}}, \mathsf{id}^*), \forall \mathsf{id} \in \mathsf{warr} \\ &\{\mathbf{E}_{\mathsf{id},1}, \ldots, \mathbf{E}_{\mathsf{id},p_3}\} \leftarrow \mathsf{MatDec}(\mathbf{E}_{\mathsf{id}}), \forall \mathsf{id} \in \mathsf{warr} \end{aligned}$$

The protocol follows the typical format of a Σ protocol, and is executed in 3 moves.
1. Commit. To begin, P uniformly samples randomness r_1, r_2 and r_3 for COM, but we omit these from the equations below for readability. It should be understood that r_i is an implicit input in the commitment c_i. It then samples the random permutations and masking vectors as:

$$\begin{array}{ll} \rho_i \hookleftarrow \{0,1\}^{ld} & \mathbf{r}_{i,\mathsf{id}} \hookleftarrow \{0,1\}^{ld}, \forall i \in [1,\delta] \\ \phi_{i,\mathbf{z}}^{(1)}, \ldots, \phi_{i,\mathbf{z}}^{(p_1)} \hookleftarrow \mathcal{P} & \mathbf{r}_{i,\mathbf{z}}^{(1)}, \ldots, \mathbf{r}_{i,\mathbf{z}}^{(p_1)} \hookleftarrow \mathbb{Z}_q^{3m_1(2ld+1)}, \forall i \in [1,\delta] \\ \phi_{0,\mathbf{z}}^{(1)}, \ldots, \phi_{0,\mathbf{z}}^{(p_1)} \hookleftarrow \hat{\mathcal{P}}, & \mathbf{r}_{0,\mathbf{z}}^{(1)}, \ldots, \mathbf{r}_{0,\mathbf{z}}^{(p_1)} \hookleftarrow \mathbb{Z}_q^{3m_1(2d+1)} \\ \phi_{i,\mathbf{a}} \hookleftarrow \mathcal{S}_{2n_1\lceil \log q_1 \rceil}, & \mathbf{r}_{i,\mathbf{a}} \hookleftarrow \mathbb{Z}_q^{2n_1\lceil \log q \rceil} \text{ for } i \in [1,\delta] \\ \phi_{\mathsf{id},\mathbf{B}} \hookleftarrow \overline{\mathcal{P}} & \mathbf{R}_{\mathsf{id},\mathbf{B}} \hookleftarrow \mathbb{Z}_q^{m_3 \times 2n_3(2d+1)} \ \forall \mathsf{id} \in \mathsf{warr} \\ \phi_{\mathsf{id},\mathbf{E}}^{(1)}, \ldots, \phi_{\mathsf{id},\mathbf{E}}^{(p_3)} \hookleftarrow \mathcal{S}_{3k_3 \times m_3} & \mathbf{R}_{\mathsf{id},\mathbf{E}}^{(1)}, \ldots, \mathbf{R}_{\mathsf{id},\mathbf{E}}^{(p_3)} \hookleftarrow \mathbb{Z}_q^{m_3 \times 3k_3} \text{ for } \mathsf{id} \in \mathsf{warr} \\ \phi_{i,\mathbf{e}}^{(1)}, \ldots, \phi_{i,\mathbf{e}}^{(p_2)} \hookleftarrow \mathcal{S}_{3h}, & \mathbf{r}_{i,\mathbf{e}}^{(1)}, \ldots, \mathbf{r}_{i,\mathbf{e}}^{(p_2)} \hookleftarrow \mathbb{Z}_q^{3h}, \forall i \in [1,\delta] \end{array}$$

Let $\rho_{\mathsf{id}} = \rho_i[kd+1 : 2kd]$, i.e. for a k^{th} level authority for attribute $\mathbf{a}_i$, take the d bits from $d_i^{(1)}$ that correspond to its position in the delegation path. This mapping links an encrypted identity's position in the ciphertext with its revocation token and location in the Bonsai signatures. This leaks no information about the witness and should be assumed to be handled implicitly by the protocol. Let $\mathbf{r}_{\mathbf{a}} := [\mathbf{r}_{1,\mathbf{a}}||..||\mathbf{r}_{\delta,\mathbf{a}}]^T$ and compute:

$$c_1 = \mathsf{COM}\Big(\{\{\phi_{i,\mathbf{z}}^{(j)}\}_{i=1}^{\delta}, \phi_{0,\mathbf{z}}^{(j)}\}_{j=1}^{p_1}, \{\phi_{\mathsf{id},\mathbf{B}}, \{\phi_{\mathsf{id},\mathbf{E}}^{(j)}\}_{j=1}^{p_3}\}_{\mathsf{id}\in\mathsf{warr}}, \{\{\phi_{i,\mathbf{e}}^{(j)}\}_{j=1}^{p_2}, \phi_{i,\mathbf{a}}, \rho_i\}_{i=1}^{\delta}, \rho_0$$

$$\overline{\mathbf{A}}^*\left(\textstyle\sum_{j=1}^{p_1}\beta_1^{(j)}\mathbf{r}_{0,\mathbf{z}}^{(j)}\right), \left\{\mathbf{A}^*\left(\textstyle\sum_{j=1}^{p_1}\beta_1^{(j)}\cdot\mathbf{r}_{i,\mathbf{z}}^{(j)}\right) - \mathbf{G}'\mathbf{r}_{i,\mathbf{a}}; \quad \mathbf{P}^*\left(\textstyle\sum_{j=1}^{p_2}\beta_2^{(j)}\mathbf{r}_{i,\mathbf{e}}^{(j)}\right) + \mathbf{Q}^*\mathbf{r}_{i,\mathbf{d}}\right\}_{i=1}^{\delta},$$

$$\left\{\mathbf{R}_{\mathsf{id},\mathbf{B}}\mathbf{R}^* + \textstyle\sum_{j=1}^{p_3}\beta_3^{(j)}\mathbf{R}_{\mathsf{id},\mathbf{E}}^{(j)}\right\}_{\mathsf{id}\in\mathsf{warr}}, \langle \mathbf{G}^*\cdot\mathbf{r}_a, \mathbf{p}\rangle\Big)$$

$$c_2 = \mathsf{COM}\Big(\{\{T_{\rho_i}\circ\phi_{i,\mathbf{z}}^{(j)}(\mathbf{r}_{i,\mathbf{z}}^{(j)})\}_{i=1}^{\delta}, T_{\rho_0}\circ\phi_{0,\mathbf{z}}^{(j)}(\mathbf{r}_{0,\mathbf{z}}^{(j)})\}_{j=1}^{p_1}, \{\{\phi_{i,\mathbf{e}}^{(j)}(\mathbf{r}_{i,\mathbf{e}}^{(j)})\}_{j=1}^{p_2}, \phi_{i,\mathbf{a}}(\mathbf{r}_{i,\mathbf{a}})\}_{i=1}^{\delta},$$

$$\{T_{\rho_i}(\mathbf{r}_{i,d}), T_{\rho_i}\circ\phi_{\mathsf{id},\mathbf{B}}(\mathbf{R}_{\mathsf{id},\mathbf{B}}), \{\phi_{\mathsf{id},\mathbf{E}}(\mathbf{R}_{\mathsf{id},\mathbf{E}})\}_{j=1}^{p_3}\}_{\mathsf{id}\in\mathsf{warr}}\Big)$$

$$c_3 = \mathsf{COM}\Big(\{\{T_{\rho_i}\circ\phi_{i,\mathbf{z}}^{(j)}(\mathbf{z}_i^{(j)} + \mathbf{r}_{i,\mathbf{z}}^{(j)})\}_{i=1}^{\delta}, T_{\rho_0}\circ\phi_{0,\mathbf{z}}^{(j)}(\mathbf{z}_0^{(j)} + \mathbf{r}_{0,\mathbf{z}}^{(j)})\}_{j=1}^{p_1},$$

$$\{\{\phi_{i,\mathbf{e}}^{(j)}(\mathbf{e}_i^{(j)} + \mathbf{r}_{i,\mathbf{e}}^{(j)})\}_{j=1}^{p_2}, \phi_{i,\mathbf{a}}(\mathbf{a}_i' + \mathbf{r}_{i,\mathbf{a}})\}_{i=1}^{\delta}, T_{\rho_i}(\mathsf{id}_i^* + \mathbf{r}_{i,\mathsf{id}}),$$

$$\mathbf{T}_{\rho_{\mathsf{id}}}\circ\phi_{\mathsf{id},\mathbf{B}}(\mathbf{B}_{\mathsf{id}} + \mathbf{R}_{\mathsf{id},\mathbf{B}}), \{\phi_{\mathsf{id},\mathbf{E}}(\mathbf{E}_{\mathsf{id}}^{(j)} + \mathbf{R}_{\mathsf{id},\mathbf{E}}^{(j)})\}_{j=1}^{p_3}\}_{\mathsf{id}\in\mathsf{warr}}\Big)$$

It sends the commitment $\mathsf{CMT} = (c_1, c_2, c_3)$ to V.

2. Challenge. The verifier sends a challenge $CH \hookleftarrow \{1,2,3\}$ to the prover.

3. Response. P replies as follows.
If $CH = 1$, let $d_i^{(1)} = \mathsf{id}_i \oplus \rho_i$ then:

$$\begin{aligned}
\text{For } i\in[1,\delta], j\in[1,p_1]&: \mathbf{t}_{i,\mathbf{z}}^{(j)} = T_{\rho_i}\circ\phi_{i,\mathbf{z}}^{(j)}(\mathbf{z}_i^{(j)}), \mathbf{v}_{i,\mathbf{z}}^{(j)} = T_{\rho_i}\circ\phi_{i,\mathbf{z}}^{(j)}(\mathbf{r}_{i,\mathbf{z}}^{(j)}).\\
\text{For } j\in[1,p_1]&: \mathbf{t}_{0,\mathbf{z}}^{(j)} = T_{\rho_0}\circ\phi_{0,\mathbf{z}}^{(j)}(\mathbf{z}_0^{(j)}), \mathbf{v}_{0,\mathbf{z}}^{(j)} = T_{\rho_0}\circ\phi_{0,\mathbf{z}}^{(j)}(\mathbf{r}_{0,\mathbf{z}}^{(j)}).\\
\text{For } i\in[1,\delta]&: \mathbf{t}_{i,\mathbf{a}} = \phi_{i,\mathbf{a}}(\mathbf{a}_i), \mathbf{v}_{i,\mathbf{a}} = \phi_{i,\mathbf{a}}^{(j)}(\mathbf{r}_{i,\mathbf{a}}^{(j)}).\\
\text{For } i\in[1,\delta], j\in[1,p_2]&: \mathbf{t}_{i,\mathbf{e}}^{(j)} = \phi_{i,\mathbf{e}}^{(j)}(\mathbf{e}_i^{(j)}), \mathbf{v}_{i,\mathbf{e}}^{(j)} = \phi_{i,\mathbf{e}}^{(j)}(\mathbf{r}_{i,\mathbf{e}}^{(j)}).\\
\text{For } i\in[1,\delta]&: \mathbf{t}_{i,\mathsf{id}} = T_{\rho_i}(\mathsf{id}_i^*), \mathbf{v}_{i,\mathsf{id}} = T_{\rho_i}(\mathbf{r}_{i,\mathsf{id}})\\
\text{For } \mathsf{id}\in\mathsf{warr}&: \mathbf{T}_{\mathsf{id},\mathbf{B}} = \ T_{\rho_i}\circ\phi_{\mathsf{id},\mathbf{B}}^{(j)}(\mathbf{B}_{\mathsf{id}}), \mathbf{V}_{\mathsf{id},\mathbf{B}} = T_{\rho_i}\circ\phi_{\mathsf{id},\mathbf{B}}(\mathbf{R}_{\mathsf{id},\mathbf{B}})\\
\text{For } \mathsf{id}\in\mathsf{warr}, j\in[1,p_3]&: \mathbf{T}_{\mathsf{id},\mathbf{E}}^{(j)} = \phi_{\mathsf{id},\mathbf{E}}^{(j)}(\mathbf{E}_{\mathsf{id}}^{(j)}), \mathbf{V}_{\mathsf{id},\mathbf{E}}^{(j)} = \phi_{\mathsf{id},\mathbf{E}}^{(j)}(\mathbf{R}_{\mathsf{id},\mathbf{E}}^{(j)}).
\end{aligned}$$

It sends the response

$$\begin{aligned}
\mathsf{RSP} := (\{\mathbf{t}_{0,\mathbf{z}}^{(j)}, \mathbf{v}_{0,\mathbf{z}}^{(j)}\}_{j=1}^{p_1}, \{\{\mathbf{t}_{i,\mathbf{z}}^{(j)}, \mathbf{v}_{i,\mathbf{z}}^{(j)}\}_{j=1}^{p_1}, \{\mathbf{t}_{i,\mathbf{e}}^{(j)}, \mathbf{v}_{i,\mathbf{e}}^{(j)}\}_{j=1}^{p_2}, \mathbf{t}_{i,\mathbf{a}}, \mathbf{v}_{i,\mathbf{a}}, \mathbf{t}_{i,\mathsf{id}}, \mathbf{v}_{i,\mathsf{id}}\}_{i=1}^{\delta},\\
\{\{\mathbf{T}_{\mathsf{id},\mathbf{E}}^{(j)}, \mathbf{V}_{\mathsf{id},\mathbf{E}}^{(j)}\}_{j=1}^{p_3}, \mathbf{T}_{\mathsf{id},\mathbf{B}}, \mathbf{V}_{\mathsf{id},\mathbf{B}}\}_{\mathsf{id}\in\mathsf{warr}})
\end{aligned}$$

If $CH = 2$, let $d_i^{(2)} = \rho_i$ then:

$$\begin{aligned}
\text{For } i\in[1,\delta], j\in[1,p_1]&: \psi_{i,\mathbf{z}}^{(j)} = \phi_{i,\mathbf{z}}^{(j)}, \mathbf{w}_{i,\mathbf{z}}^{(j)} = \mathbf{z}_i^{(j)} + \mathbf{r}_{i,\mathbf{z}}^{(j)}\\
\text{For } j\in[1,p_1]&: \psi_{0,\mathbf{z}}^{(j)} = \phi_{0,\mathbf{z}}^{(j)}, \mathbf{w}_{0,\mathbf{z}}^{(j)} = \mathbf{z}_0^{(j)} + \mathbf{r}_{0,\mathbf{z}}^{(j)}\\
\text{For } i\in[1,\delta]&: \psi_{i,\mathbf{a}} = \phi_{i,\mathbf{a}}, \mathbf{w}_{i,\mathbf{a}} = \mathbf{a}_i' + \mathbf{r}_{i,\mathbf{a}}\\
\text{For } i\in[1,\delta], j\in[1,p_2]&: \psi_{i,\mathbf{e}}^{(j)} = \phi_{i,\mathbf{e}}^{(j)}, \mathbf{w}_{i,\mathbf{e}}^{(j)} = \mathbf{e}_i^{(j)} + \mathbf{r}_{i,\mathbf{e}}^{(j)}\\
\text{For } i\in[1,\delta]&: \mathbf{w}_{i,\mathsf{id}} = \mathsf{id}_i^* + \mathbf{r}_{i,\mathsf{id}},\\
\text{For } \mathsf{id}\in\mathsf{warr}&: \psi_{\mathsf{id},\mathbf{B}} = \phi_{\mathsf{id},\mathbf{B}}, \mathbf{W}_{\mathsf{id},\mathbf{B}} = \mathbf{B}_{\mathsf{id}} + \mathbf{R}_{\mathsf{id},\mathbf{B}}\\
\text{For } \mathsf{id}\in\mathsf{warr}, j\in[1,p_3]&: \psi_{\mathsf{id},\mathbf{E}}^{(j)} = \phi_{\mathsf{id},\mathbf{E}}^{(j)}, \mathbf{W}_{\mathsf{id},\mathbf{E}}^{(j)} = \mathbf{E}_{\mathsf{id}}^{(j)} + \mathbf{R}_{\mathsf{id},\mathbf{E}}^{(j)}
\end{aligned}$$

It sends the response

$$\begin{aligned}\mathsf{RSP} := (\{\psi^{(j)}_{0,\mathbf{z}},\mathbf{w}^{(j)}_{0,\mathbf{z}}\}_{j=1}^{p_1},\{\{\psi^{(j)}_{i,\mathbf{z}},\mathbf{w}^{(j)}_{i,\mathbf{z}}\}_{j=1}^{p_1},\{\psi^{(j)}_{i,\mathbf{e}},\mathbf{w}^{(j)}_{i,\mathbf{e}}\}_{j=1}^{p_2}\psi_{i,\mathbf{a}},\mathbf{w}_{i,\mathbf{a}},\mathbf{w}_{i,\mathsf{id}}\}_{i=1}^{\delta},\\ \{\{\psi^{(j)}_{\mathsf{id},\mathbf{E}},\mathbf{W}^{(j)}_{\mathsf{id},\mathbf{E}}\}_{j=1}^{p_3},\psi_{\mathsf{id},\mathbf{B}},\mathbf{W}_{\mathsf{id},\mathbf{B}}\}_{\mathsf{id}\in\mathsf{warr}})\end{aligned}$$

If $CH=3$, let $d_i^{(3)}=\rho_i$ then:

$$\begin{aligned}\text{For } i\in[1,\delta],j\in[1,p_1]&: \theta^{(j)}_{i,\mathbf{z}}=\phi^{(j)}_{i,\mathbf{z}},\mathbf{y}^{(j)}_{i,\mathbf{z}}=\mathbf{r}^{(j)}_{i,\mathbf{z}}\\ \text{For } j\in[1,p_1]&: \theta^{(j)}_{0,\mathbf{z}}=\psi^{(j)}_{0,\mathbf{z}},\mathbf{y}^{(j)}_{0,\mathbf{z}}=\mathbf{r}^{(j)}_{0,\mathbf{z}}\\ \text{For } i\in[1,\delta]&: \theta_{i,\mathbf{a}}=\phi_{i,\mathbf{a}},\mathbf{y}_{i,\mathbf{a}}=\mathbf{r}_{i,\mathbf{a}}\\ \text{For } i\in[1,\delta],j\in[1,p_2]&: \theta^{(j)}_{i,\mathbf{e}}=\phi^{(j)}_{i,\mathbf{e}},\mathbf{y}^{(j)}_{i,\mathbf{e}}=\mathbf{r}^{(j)}_{i,\mathbf{e}}\\ \text{For } i\in[1,\delta]&: \mathbf{y}_{i,\mathsf{id}}=\mathbf{r}_{i,\mathsf{id}},\\ \text{For } \mathsf{id}\in\mathsf{warr}&: \theta_{\mathsf{id},\mathbf{B}}=\phi_{\mathsf{id},\mathbf{B}},\mathbf{Y}_{\mathsf{id},\mathbf{B}}=\mathbf{R}_{\mathsf{id},\mathbf{B}}\\ \text{For } \mathsf{id}\in\mathsf{warr},j\in[1,p_3]&: \theta^{(j)}_{\mathsf{id},\mathbf{E}}=\phi^{(j)}_{\mathsf{id},\mathbf{E}},\mathbf{Y}^{(j)}_{\mathsf{id},\mathbf{E}}=\mathbf{R}^{(j)}_{\mathsf{id},\mathbf{E}}\end{aligned}$$

It sends the response

$$\begin{aligned}\mathsf{RSP} := (\{\theta^{(j)}_{0,\mathbf{z}},\mathbf{y}^{(j)}_{0,\mathbf{z}}\}_{j=1}^{p_1},\{\{\theta^{(j)}_{i,\mathbf{z}},\mathbf{y}^{(j)}_{i,\mathbf{z}}\}_{j=1}^{p_1},\{\theta^{(j)}_{i,\mathbf{e}},\mathbf{y}^{(j)}_{i,\mathbf{e}}\}_{j=1}^{p_2}\theta_{i,\mathbf{a}},\mathbf{y}_{i,\mathbf{a}},\mathbf{y}_{i,\mathsf{id}}\}_{i=1}^{\delta},\\ \{\{\theta^{(j)}_{\mathsf{id},\mathbf{E}},\mathbf{Y}^{(j)}_{\mathsf{id},\mathbf{E}}\}_{j=1}^{p_3},\theta_{\mathsf{id},\mathbf{B}},\mathbf{Y}_{\mathsf{id},\mathbf{B}}\}_{\mathsf{id}\in\mathsf{warr}})\end{aligned}$$

Verification. Depending on the challenge, the verifier computes the following.
If $CH=1$, then:
Parse RSP as $(\{\mathbf{t}^{(j)}_{0,\mathbf{z}},\mathbf{v}^{(j)}_{0,\mathbf{z}}\}_{j=1}^{p_1},\{\{\mathbf{t}^{(j)}_{i,\mathbf{z}},\mathbf{v}^{(j)}_{i,\mathbf{z}}\}_{j=1}^{p_1},\{\mathbf{t}^{(j)}_{i,\mathbf{e}},\mathbf{v}^{(j)}_{i,\mathbf{e}}\}_{j=1}^{p_2},\mathbf{t}_{i,\mathbf{a}},\mathbf{v}_{i,\mathbf{a}},\mathbf{t}_{i,\mathsf{id}},\mathbf{v}_{i,\mathsf{id}}\}_{i=1}^{\delta},$ $\{\{\mathbf{T}^{(j)}_{\mathsf{id},\mathbf{E}},\mathbf{V}^{(j)}_{\mathsf{id},\mathbf{E}}\}_{j=1}^{p_3},\mathbf{T}_{\mathsf{id},\mathbf{B}},\mathbf{V}_{\mathsf{id},\mathbf{B}}\}_{\mathsf{id}\in\mathsf{warr}})$.
Let $d_{\mathsf{id}}^{(1)}=d_i^{(1)}[kd+1:2kd]$, i.e. for a k^{th} level authority for attribute $\mathbf{a}_i$, take the d bits from $d_i^{(1)}$ that correspond to its position in the delegation path. This mapping is assumed to be sent by the prover. Then, check that:

$$\begin{aligned}\text{For all } j\in[1,...,p_1]&: \mathbf{t}^{(j)}_{0,\mathbf{z}}\in\mathsf{SecretExt}_{\beta}(d_0^{(1)})\\ \text{For all } j\in[1,...,p_1],i\in[i,\delta]&: \mathbf{t}^{(j)}_{i,\mathbf{z}}\in\mathsf{SecretExt}_{\beta}(d_i^{(1)})\\ \text{For all } \mathsf{id}\in\mathsf{warr}&: \mathbf{T}_{\mathsf{id},\mathbf{B}}\in\overline{\mathsf{SecretExt}}_{\beta}(d_{\mathsf{id}}^{(1)})\\ \text{For all } i\in[]1,\delta]&: \mathbf{t}_{i,\mathsf{id}}=d_i^{(1)}\\ \text{For all } j\in[1,...,p_2],i\in[i,\delta]&: \mathbf{t}^{(j)}_{i,\mathbf{e}}\in\mathsf{B}_{3h}\\ \text{For all } j\in[1,...,p_3],\mathsf{id}\in\mathsf{warr}&: \mathbf{T}^{(j)}_{\mathsf{id},\mathbf{E}}\in\mathsf{B}_{m_3\times 3k_3}\\ \text{For all } i\in[i,\delta]&: \mathbf{t}_{i,\mathbf{a}}\in\mathsf{B}_{2n_1\log q_1}\end{aligned}$$

and the following holds:

$$\begin{cases} c_2=\mathsf{COM}(\{\{\mathbf{v}^{(j)}_{i,\mathbf{z}},\}_{i=1}^{\delta}\mathbf{v}^{(j)}_{0,\mathbf{z}}\}_{j=1}^{p_1},\{\{\mathbf{v}^{(j)}_{i,\mathbf{e}}\}_{j=1}^{p_2}\mathbf{v}_{i,\mathbf{a}}\}_{i=1}^{\delta},\{\mathbf{v}_{\mathsf{id}},\mathbf{V}_{\mathsf{id},\mathbf{B}},\{\mathbf{V}_{\mathsf{id},\mathbf{E}}\}_{j=1}^{p_3}\}_{\mathsf{id}\in\mathsf{warr}})\\ c_3=\mathsf{COM}(\{\{\mathbf{t}^{(j)}_{i,\mathbf{z}}+\mathbf{v}^{(j)}_{i,\mathbf{z}}\}_{i=1}^{\delta},\mathbf{t}^{(j)}_{0,\mathbf{z}}+\mathbf{v}^{(j)}_{0,\mathbf{z}}\}_{j=1}^{p_1},\{\{\mathbf{t}^{(j)}_{i,\mathbf{e}}+\mathbf{v}^{(j)}_{i,\mathbf{e}}\}_{j=1}^{p_2},\mathbf{t}_{i,\mathbf{a}}+\mathbf{v}_{i,\mathbf{a}}\}_{i=1}^{\delta},\\ \qquad \{\mathbf{t}_{i,\mathsf{id}}+\mathbf{v}_{i,\mathsf{id}},\mathbf{T}_{\mathsf{id},\mathbf{B}}+\mathbf{V}_{\mathsf{id},\mathbf{B}},\{\mathbf{T}_{\mathsf{id},\mathbf{E}}+\mathbf{V}_{\mathsf{id},\mathbf{E}}\}_{j=1}^{p_3}\}_{\mathsf{id}\in\mathsf{warr}})\end{cases}$$

If $CH=2$, then:
Parse RSP as $(\{\psi^{(j)}_{0,\mathbf{z}},\mathbf{w}^{(j)}_{0,\mathbf{z}}\}_{j=1}^{p_1},\{\{\psi^{(j)}_{i,\mathbf{z}},\mathbf{w}^{(j)}_{i,\mathbf{z}}\}_{j=1}^{p_1},\{\psi^{(j)}_{i,\mathbf{e}},\mathbf{w}^{(j)}_{i,\mathbf{e}}\}_{j=1}^{p_2}\psi_{i,\mathbf{a}},\mathbf{w}_{i,\mathbf{a}},\psi_{\mathsf{id}_i},\mathbf{w}_{i,\mathsf{id}}\}_{i=1}^{\delta},$ $\{\{\psi^{(j)}_{\mathsf{id},\mathbf{E}},\mathbf{W}^{(j)}_{\mathsf{id},\mathbf{E}}\}_{j=1}^{p_3},\psi_{\mathsf{id},\mathbf{B}},\mathbf{W}_{\mathsf{id},\mathbf{B}}\}_{\mathsf{id}\in\mathsf{warr}})$, and build $\mathbf{w}_a := [\mathbf{w}_{1,\mathbf{a}}||..||\mathbf{w}_{\delta,\mathbf{a}}]^T$ and $\mathbf{w}_{i,\mathbf{d}} := [\mathbf{w}_{\mathsf{id}_{i,1}}||..||\mathbf{w}_{\mathsf{id}_{i,l}}]^T$.
Let $d_{\mathsf{id}}^{(2)}=d_i^{(2)}[kd+1:2kd]$, i.e. for a k^{th} level authority for attribute $\mathbf{a}_i$, take the d bits from $d_i^{(2)}$ that correspond to its position in the delegation path. This mapping is assumed to be sent by the prover.

Check that

$$\begin{cases} c_1 = \mathsf{COM}\Bigg(\{\{\psi_{i,\mathbf{z}}^{(i)}\}_{i=1}^{\delta}, \psi_{0,\mathbf{z}}^{(j)}\}_{j=1}^{p_1}, \{\psi_{\mathsf{id}}, \psi_{\mathsf{id},\mathbf{B}}, \{\psi_{\mathsf{id},\mathbf{E}}^{(j)}\}_{j=1}^{p_3}\}_{\mathsf{id}\in\mathsf{warr}}, \{\{\psi_{i,\mathbf{e}}^{(i)}\}_{j=1}^{p_2}, \psi_{i,\mathbf{a}}, d_i^{(2)}\}_{i=1}^{\delta}, d_0^{(2)} \\ \overline{\mathbf{A}}^* \left(\sum_{j=1}^{p_1} \beta_1^{(j)} \mathbf{w}_{0,\mathbf{z}}^{(j)} \right) - \mathbf{u}, \left\{ \mathbf{A}^* \left(\sum_{j=1}^{p_1} \beta_1^{(j)} \cdot \mathbf{w}_{i,\mathbf{z}}^{(j)} \right) - \overline{\mathbf{G}}^* \mathbf{w}_{i,\mathbf{a}}; \mathbf{P}^* \left(\sum_{j=1}^{p_2} \beta_2^{(j)} \mathbf{w}_{i,\mathbf{e}}^{(j)} \right) + \mathbf{Q}^* \mathbf{w}_{i,\mathbf{d}} - \mathbf{f}_i \right\}_{i=1}^{\delta}, \\ \qquad \left\{ \mathbf{W}_{\mathsf{id},\mathbf{B}} \mathbf{R}^* + \sum_{j=1}^{p_3} \beta_3^{(j)} \mathbf{W}_{\mathsf{id},\mathbf{E}}^{(j)} - \mathbf{C}_{\mathsf{id}} \right\}_{\mathsf{id}\in\mathsf{warr}}, \langle \mathbf{G}^* \cdot \mathbf{w}_a, \mathbf{p} \rangle - 1 \Bigg) \\ c_3 = \mathsf{COM}(\{\{T_{d_i^{(2)}} \circ \psi_{i,\mathbf{z}}^{(j)}(\mathbf{w}_{i,\mathbf{z}}^{(j)})\}_{i=1}^{\delta}, T_{d_0^{(2)}} \circ \psi_{0,\mathbf{z}}^{(j)}(\mathbf{w}_{0,\mathbf{z}}^{(j)})\}_{j=1}^{p_1}, \{\{\{\psi_{i,\mathbf{e}}^{(j)}(\mathbf{w}_{i,\mathbf{e}}^{(j)})\}_{j=1}^{p_2}, \\ \qquad \psi_{i,\mathbf{a}}(\mathbf{w}_{i,\mathbf{a}})\}_{i=1}^{\delta}, \{T_{d_i^{(2)}}(\mathbf{w}_{i,\mathsf{id}}), T_{d_{\mathsf{id}}^{(2)}} \circ \psi_{\mathsf{id},\mathbf{B}}(\mathbf{W}_{\mathsf{id},\mathbf{B}}), \{\psi_{\mathsf{id},\mathbf{E}}(\mathbf{W}_{\mathsf{id},\mathbf{E}}^{(j)})\}_{j=1}^{p_3}\}_{\mathsf{id}\in\mathsf{warr}}) \end{cases}$$

If $CH = 3$, then:
Parse RSP as $(\{\theta_{0,\mathbf{z}}^{(j)}, \mathbf{y}_{0,\mathbf{z}}^{(j)}\}_{j=1}^{p_1}, \{\{\theta_{i,\mathbf{z}}^{(j)}, \mathbf{y}_{i,\mathbf{z}}^{(j)}\}_{j=1}^{p_1}, \{\theta_{i,\mathbf{e}}^{(j)}, \mathbf{y}_{i,\mathbf{e}}^{(j)}\}_{j=1}^{p_2} \theta_{i,\mathbf{a}}, \mathbf{y}_{i,\mathbf{a}},, \theta_{\mathsf{id}_i}, \mathbf{y}_{i,\mathsf{id}}\}_{i=1}^{\delta},$
$\{\{\theta_{\mathsf{id},\mathbf{E}}^{(j)}, \mathbf{Y}_{\mathsf{id},\mathbf{E}}^{(j)}\}_{j=1}^{p_3}, \theta_{\mathsf{id},\mathbf{B}}, \mathbf{Y}_{\mathsf{id},\mathbf{B}}\}_{\mathsf{id}\in\mathsf{warr}})$. Build the vectors $\mathbf{y}_a := [\mathbf{y}_{1,\mathbf{a}} || .. || \mathbf{y}_{\delta,\mathbf{a}}]^T$ and $\mathbf{y}_{i,\mathbf{d}} := [\mathbf{y}_{\mathsf{id}_{i,1}} || .. || \mathbf{y}_{\mathsf{id}_{i,l}}]^T$ according to the mapping. Check that

$$\begin{cases} c_1 = \mathsf{COM}\Bigg(\{\{\theta_{i,\mathbf{z}}^{(i)},\}_{i=1}^{\delta}, \theta_{0,\mathbf{z}}^{(j)}\}_{j=1}^{p_1}, \{\theta_{\mathsf{id}}, \theta_{\mathsf{id},\mathbf{B}}, \{\theta_{\mathsf{id},\mathbf{E}}^{(j)}\}_{j=1}^{p_3}\}_{\mathsf{id}\in\mathsf{warr}}, \{\{\theta_{i,\mathbf{e}}^{(i)}\}_{j=1}^{p_2}, \theta_{i,\mathbf{a}}\}_{i=1}^{\delta}, \\ \qquad \overline{\mathbf{A}}^* \left(\sum_{j=1}^{p_1} \beta_1^{(j)} \mathbf{y}_{0,\mathbf{z}}^{(j)} \right), \left\{ \mathbf{A}^* \left(\sum_{j=1}^{p_1} \beta_1^{(j)} \cdot \mathbf{y}_{i,\mathbf{z}}^{(j)} \right) - \mathbf{G}' \mathbf{y}_{i,\mathbf{a}}; \mathbf{P}^* \left(\sum_{j=1}^{p_2} \beta_2^{(j)} \mathbf{y}_{i,\mathbf{e}}^{(j)} \right) + \mathbf{Q}^* \mathbf{y}_{i,\mathbf{d}} \right\}_{i=1}^{\delta}, \\ \qquad \left\{ \mathbf{Y}_{\mathsf{id},\mathbf{B}} \mathbf{R}^* + \sum_{j=1}^{p_3} \beta_3^{(j)} \mathbf{Y}_{\mathsf{id},\mathbf{E}}^{(j)} - \mathbf{C}_{\mathsf{id}} \right\}_{\mathsf{id}\in\mathsf{warr}}, \langle \mathbf{G}^* \cdot \mathbf{y}_a, \mathbf{p} \rangle \Bigg) \\ c_2 = \mathsf{COM}(\{\{T_{d_{\mathsf{id}}^{(2)}} \circ \theta_{i,\mathbf{z}}^{(j)}(\mathbf{y}_{i,\mathbf{z}}^{(j)})\}_{i=1}^{\delta}, \mathbf{T}_{d_{\mathsf{id}}^{(2)}} \circ \theta_{0,\mathbf{z}}^{(j)}(\mathbf{y}_{0,\mathbf{z}}^{(j)})\}_{j=1}^{p_1}, \{\{\{\psi_{i,\mathbf{e}}^{(j)}(\mathbf{y}_{i,\mathbf{e}}^{(j)})\}_{j=1}^{p_2}, \psi_{i,\mathbf{a}}(\mathbf{y}_{i,\mathbf{a}})\}_{i=1}^{\delta}, \\ \qquad \{T_{d_i^{(2)}}(\mathbf{y}_{i,\mathsf{id}}), T_{d_{\mathsf{id}}^{(2)}} \circ \theta_{\mathsf{id},\mathbf{B}}(\mathbf{Y}_{\mathsf{id},\mathbf{B}}), \{\psi_{\mathsf{id},\mathbf{E}}(\mathbf{Y}_{\mathsf{id},\mathbf{E}}^{(j)})\}_{j=1}^{p_3}\}_{\mathsf{id}\in\mathsf{warr}}) \end{cases}$$

The verifier outputs 1 if and only if all the conditions hold, else output 0.

B Proof of Theorem 1

We first recall the theorem.

Theorem 1. *Let* COM *be a statistically hiding and computationally binding string commitment scheme. Then the protocol given in Section 5.2 is a zero-knowledge argument of knowledge with perfect completeness with soundness error* $2/3$. *Explicitly, that is:*

- *There exists a polynomial-time simulator that outputs an accepting transcript that is statistically close to a transcript produced by an honest prover with a valid witness.*
- *There exists a polynomial-time extractor, such that, on input of a commitment* CMT *and 3 responses* $(\mathsf{RSP}_1, \mathsf{RSP}_2, \mathsf{RSP}_3)$ *corresponding to each challenge* $\{1,2,3\}$, *outputs a valid witness for the relation* $\mathcal{R}_1$.

Proof. The proof consists of 4 parts and utilises standard techniques for Stern-like protocols.

Soundness. As with typical Stern-like protocols, our protocol has soundness of $2/3$. This can be seen by analysing a dishonest prover attempting to produce a valid script. Before the protocol is run, the dishonest prover will guess which challenge from $\{1,2,3\}$ the verifier will not ask for. If the adversary guesses correctly, it is able to simulate the protocol without knowledge of a valid witness and wrongly convince the verifier to accept the transcript. If the adversary guesses incorrectly, the verifier is able to identify the cheating behaviour. Therefore, the protocol has soundness of $2/3$. By repeating the protocol $t = \omega(\log n_1)$ times in parallel, this error can be made negligibly small.

Perfect Completeness. We note that, using the Decomposition-Extension technique, an honest prover following the protocol with a valid witness will always be able to produce a script that is accepted by the verifier. Therefore, the protocol has perfect completeness.

Zero-Knowledge. We will show that if COM is a statistically hiding commitment scheme, then ZKAoK has the Zero-Knowledge property. In particular, we construct a polynomial time simulator Sim that is able to produce a transcript that is statistically close to a real protocol run with an possibly malicious verifier. This simulator will fail with probability negligibly close to 2/3. The public parameters are $\mathbf{A},\mathbf{R},\{\mathbf{A}_i^b\}_{i=1}^{ld}\{\mathbf{R}_i^b\}_{i=1}^{d},\mathbf{G}^*,\mathbf{P}^*,\mathbf{Q},\{\mathbf{C}_{\mathsf{id}}\}_{\mathsf{id}\in\mathsf{warr}},\{\mathbf{f}_i\}_{i=1}^{\delta},\mathbf{p},\mathbf{u}$. Sim starts by selecting a challenge $\overline{CH}\in\{1,2,3\}$ that it guesses the verifier will not choose.

If $\overline{CH}=1$, then: In this case, Sim computes the following vectors and matrices via standard linear algebra:

1. $\mathbf{a}_i\in\mathbb{Z}_q^{n_1}$ s.t. $\mathbf{a}:=[\mathbf{a}_1||...||\mathbf{a}_\delta]$ satisfies $\langle\mathbf{G}^*\cdot\mathsf{bin}(\mathbf{a}),\mathbf{p}\rangle=1 \mod q$
2. $\mathbf{z}_i^{(1)},...,\mathbf{z}_i^{(p_1)}\in\mathsf{B}_{3m_1(2ld+1)}$ s.t. $\mathbf{A}^*\left(\sum_{j=1}^{p_1}\beta_1^{(j)}\cdot\mathbf{z}_i^{(j)}\right)=\mathbf{G}'\cdot\mathbf{a}_i \mod q_1$ for $i\in[1,\delta]$
3. $\mathbf{e}_i^{(1)},...,\mathbf{e}_i^{(p_2)}\in\mathsf{B}_{3hl(2d+2)}$, and $\mathsf{id}_i\in\{0,1\}^{ld}$
$$\text{s.t. } \mathbf{P}^*\left(\sum_{j=1}^{p_2}\beta_2^{(j)}\cdot\mathbf{e}_i^{(j)}\right)+\mathbf{Q}\mathsf{id}_i=\mathbf{f}_{\mathsf{id}_i} \mod q_2 \text{ for } i\in[1,\delta]$$
4. For each $\mathsf{id}\in\mathsf{warr},\mathbf{B}_{\mathsf{id}}\in\mathsf{B}_{m_3\times(2d+1)n_3},\mathbf{E}_{\mathsf{id}}^{(1)},...,\mathbf{E}_{\mathsf{id}}^{(p_3)}\in\mathsf{B}_{m_3\times k_3}$ s.t. $\mathbf{C}_{\mathsf{id}}=\mathbf{B}_{\mathsf{id}}\mathbf{R}^*+\sum_{j=1}^{p_3}\beta_3^{(j)}\mathbf{E}_{\mathsf{id}}$
5. $\mathbf{z}_0^{(1)},...,\mathbf{z}_0^{(p_1)}\in\mathsf{B}_{3m_1(2ld+1)}$ s.t. $\overline{\mathbf{A}}^*\left(\sum_{j=1}^{p_1}\beta_1^{(j)}\cdot\mathbf{z}_0^{(j)}\right)=\mathbf{u} \mod q_1$

It then samples the random permutations and masking vector as in the honest protocol, namely:

$$\begin{array}{ll}
\rho_i\hookleftarrow\{0,1\}^{ld} & \mathbf{r}_{i,\mathsf{id}}\hookleftarrow\mathbb{Z}_q^{ld},\forall i\in[1,\delta] \\
\phi_{i,\mathbf{z}}^{(1)},...,\phi_{i,\mathbf{z}}^{(p_1)}\hookleftarrow\mathcal{P} & \mathbf{r}_{i,\mathbf{z}}^{(1)},...,\mathbf{r}_{i,\mathbf{z}}^{(p_1)}\hookleftarrow\mathbb{Z}_q^{3m_1(2ld+1)},\forall i\in[1,\delta] \\
\phi_{0,\mathbf{z}}^{(1)},...,\phi_{0,\mathbf{z}}^{(p_1)}\hookleftarrow\hat{\mathcal{P}}, & \mathbf{r}_{0,\mathbf{z}}^{(1)},...,\mathbf{r}_{0,\mathbf{z}}^{(p_1)}\hookleftarrow\mathbb{Z}_q^{3m_1(2d+1)} \\
\phi_{i,\mathbf{a}}\hookleftarrow\mathcal{S}_{2n_1\lceil\log q_1\rceil}, & \mathbf{r}_{i,\mathbf{a}}\hookleftarrow\mathbb{Z}_q^{2n_1\lceil\log q\rceil} \text{ for } i\in[1,\delta] \\
\phi_{\mathsf{id},\mathbf{B}}\hookleftarrow\overline{\mathcal{P}} & \mathbf{R}_{\mathsf{id},\mathbf{B}}\hookleftarrow\mathbb{Z}_q^{m_3\times 2n_3(2d+1)}\ \forall\mathsf{id}\in\mathsf{warr} \\
\phi_{\mathsf{id},\mathbf{E}}^{(1)},..,\phi_{\mathsf{id},\mathbf{E}}^{(p_3)}\hookleftarrow\mathcal{S}_{3k_3\times m_3} & \mathbf{R}_{\mathsf{id},\mathbf{E}}^{(1)},...,\mathbf{R}_{\mathsf{id},\mathbf{E}}^{(p_3)}\hookleftarrow\mathbb{Z}_q^{m_3\times 3k_3} \text{ for } \mathsf{id}\in\mathsf{warr} \\
\phi_{i,\mathbf{e}}^{(1)},...,\phi_{i,\mathbf{e}}^{(p_2)}\hookleftarrow\mathcal{S}_{3h}, & \mathbf{r}_{i,\mathbf{e}}^{(1)},...,\mathbf{r}_{i,\mathbf{e}}^{(p_2)}\hookleftarrow\mathbb{Z}_q^{3h},\forall i\in[1,\delta]
\end{array}$$

The simulator Sim then computes the following commitments $\mathsf{CMT}=(c_1',c_2',c_3')$ and sends them to the verifier. Let $\rho_{\mathsf{id}}=\rho_i[kd+1:2kd]$.

Then, let $\mathbf{r}_\mathbf{a}:=[\mathbf{r}_{1,\mathbf{a}}||..||\mathbf{r}_{\delta,\mathbf{a}}]^T$ and computes the following commitments:

$$c_1=\mathsf{COM}\Bigg(\{\{\phi_{i,\mathbf{z}}^{(j)}\}_{i=1}^{\delta},\phi_{0,\mathbf{z}}^{(j)}\}_{j=1}^{p_1},\{\phi_{\mathsf{id},\mathbf{B}},\{\phi_{\mathsf{id},\mathbf{E}}^{(j)}\}_{j=1}^{p_3}\}_{\mathsf{id}\in\mathsf{warr}},\{\{\phi_{i,\mathbf{e}}^{(j)}\}_{j=1}^{p_2},\phi_{i,\mathbf{a}},\rho_i\}_{i=1}^{\delta},\rho_0$$

$$\overline{\mathbf{A}}^*\left(\textstyle\sum_{j=1}^{p_1}\beta_1^{(j)}\mathbf{r}_{0,\mathbf{z}}^{(j)}\right),\left\{\mathbf{A}^*\left(\textstyle\sum_{j=1}^{p_1}\beta_1^{(j)}\cdot\mathbf{r}_{i,\mathbf{z}}^{(j)}\right)-\mathbf{G}'\mathbf{r}_{i,\mathbf{a}};\quad \mathbf{P}^*\left(\textstyle\sum_{j=1}^{p_2}\beta_2^{(j)}\mathbf{r}_{i,\mathbf{e}}^{(j)}\right)+\mathbf{Q}^*\mathbf{r}_{i,\mathbf{d}}\right\}_{i=1}^{\delta},$$

$$\left\{\mathbf{R}_{\mathsf{id},\mathbf{B}}\mathbf{R}^*+\textstyle\sum_{j=1}^{p_3}\beta_3^{(j)}\mathbf{R}_{\mathsf{id},\mathbf{E}}^{(j)}\right\}_{\mathsf{id}\in\mathsf{warr}},\langle\mathbf{G}^*\cdot\mathbf{r}_a,\mathbf{p}\rangle\Bigg)$$

$$c_2 = \mathsf{COM}\Bigg(\{\{T_{\rho_i}\circ\phi^{(j)}_{i,\mathbf{z}}(\mathbf{r}^{(j)}_{i,\mathbf{z}})\}^{\delta}_{i=1}, T_{\rho_0}\circ\phi^{(j)}_{0,\mathbf{z}}(\mathbf{r}^{(j)}_{0,\mathbf{z}})\}^{p_1}_{j=1}, \{\{\phi^{(j)}_{i,\mathbf{e}}(\mathbf{r}^{(j)}_{i,\mathbf{e}})\}^{p_2}_{j=1}, \phi_{i,\mathbf{a}}(\mathbf{r}_{i,\mathbf{a}})\}^{\delta}_{i=1},$$

$$\{T_{\rho_i}(\mathbf{r}_{i,d}), T_{\rho_i}\circ\phi_{\mathsf{id},\mathbf{B}}(\mathbf{R}_{\mathsf{id},\mathbf{B}}), \{\phi_{\mathsf{id},\mathbf{E}}(\mathbf{R}_{\mathsf{id},\mathbf{E}})\}^{p_3}_{j=1}\}_{\mathsf{id}\in\mathsf{warr}}\Bigg)$$

$$c_3 = \mathsf{COM}\Bigg(\{\{T_{\rho_i}\circ\phi^{(j)}_{i,\mathbf{z}}(\mathbf{z}^{(j)}_i + \mathbf{r}^{(j)}_{i,\mathbf{z}})\}^{\delta}_{i=1}, T_{\rho_0}\circ\phi^{(j)}_{0,\mathbf{z}}(\mathbf{z}^{(j)}_0 + \mathbf{r}^{(j)}_{0,\mathbf{z}})\}^{p_1}_{j=1},$$

$$\{\{\phi^{(j)}_{i,\mathbf{e}}(\mathbf{e}^{(j)}_i + \mathbf{r}^{(j)}_{i,\mathbf{e}})\}^{p_2}_{j=1}, \phi_{i,\mathbf{a}}(\mathbf{a}'_i + \mathbf{r}_{i,\mathbf{a}})\}^{\delta}_{i=1}, T_{\rho_i}(\mathsf{id}^*_i + \mathbf{r}_{i,\mathsf{id}}),$$

$$\mathbf{T}_{\rho_{\mathsf{id}}}\circ\phi_{\mathsf{id},\mathbf{B}}(\mathbf{B}_{\mathsf{id}} + \mathbf{R}_{\mathsf{id},\mathbf{B}}), \{\phi_{\mathsf{id},\mathbf{E}}(\mathbf{E}^{(j)}_{\mathsf{id}} + \mathbf{R}^{(j)}_{\mathsf{id},\mathbf{E}})\}^{p_3}_{j=1}\}_{\mathsf{id}\in\mathsf{warr}}\Bigg)$$

Upon receiving the challenge from the verifier, it proceeds as follows:
If $CH = 1$, output $\perp$ and abort.
If $CH = 2$, send response

$$\mathsf{RSP} := (\{\{\phi^{(j)}_{i,\mathbf{z}}, \mathbf{z}^{(j)}_i + \mathbf{r}^{(j)}_{i,\mathbf{z}}\}^{\delta}_{i=1}, \phi^{(j)}_{0,\mathbf{z}}, \mathbf{z}^{(j)}_0 + \mathbf{r}^{(j)}_{0,\mathbf{z}}\}^{p_1}_{j=1}, \{\{\phi^{(j)}_{i,\mathbf{e}}, \mathbf{e}^{(j)}_i + \mathbf{r}^{(j)}_{i,\mathbf{e}}\}^{p_2}_{j=1},$$
$$\phi_{i,\mathbf{a}}, \mathbf{a}_i + \mathbf{r}_{i,\mathbf{a}}\}^{\delta}_{i=1}, \{\{\phi^{(j)}_{\mathsf{id},\mathbf{E}}, \mathbf{E}^{(j)}_{\mathsf{id}} + \mathbf{R}^{(j)}_{\mathsf{id},\mathbf{E}}\}^{p}_{j=1}, \mathsf{id}^*_i + \mathbf{r}_{i,\mathsf{id}}, \phi_{\mathsf{id},\mathbf{B}}, \mathbf{B}_{\mathsf{id}} + \mathbf{R}_{\mathsf{id},\mathbf{B}}\}_{\mathsf{id}\in\mathsf{warr}})$$

If $CH = 3$, send response

$$\mathsf{RSP} := (\{\{\phi^{(j)}_{i,\mathbf{z}}, \mathbf{r}^{(j)}_{i,\mathbf{z}}\}^{\delta}_{i=1}, \phi^{(j)}_{0,\mathbf{z}}, \mathbf{r}^{(j)}_{0,\mathbf{z}}\}^{p_1}_{j=1}, \{\{\phi^{(j)}_{i,\mathbf{e}}, \mathbf{r}^{(j)}_{i,\mathbf{e}}\}^{p_2}_{j=1}, \phi_{i,\mathbf{a}}, \mathbf{r}_{i,\mathbf{a}}\}^{\delta}_{i=1},$$
$$\{\{\phi^{(j)}_{\mathsf{id},\mathbf{E}}, \mathbf{R}^{(j)}_{\mathsf{id},\mathbf{E}}\}^{p_3}_{j=1}, \mathbf{r}_{\mathsf{id}}, \phi_{\mathsf{id},\mathbf{B}}, \mathbf{R}_{\mathsf{id},\mathbf{B}}\}_{\mathsf{id}\in\mathsf{warr}})$$

If $\overline{CH} = 2$, *then:* The simulator samples random bit strings, permutations and masking vectors as follows:

$$\begin{array}{ll}
\mathsf{uid}, \mathsf{id}_{1,1}, \ldots, \mathsf{id}_{l-1,\delta} \hookleftarrow \{0,1\}^d & \mathbf{z}^{(j)}_1, \ldots, \mathbf{z}^{(j)}_\delta \hookleftarrow \mathsf{SecretExt}_\beta(\mathsf{id}_{1,i}||\ldots||\mathsf{id}_{l-1,i}||\mathsf{uid}) \\
\mathbf{B}_{\mathsf{id}} \leftarrow \overline{\mathsf{SecretExt}_\beta}(\mathsf{id}) & \mathbf{E}_{\mathsf{id}} \hookleftarrow \mathbb{Z}_q^{m_3\times k_3} \ \text{ for } \mathsf{id}\in\mathsf{warr} \\
\mathbf{e}^{(1)}_i, \ldots, \mathbf{e}^{(p)}_i \hookleftarrow \mathsf{B}_{3h}, & \mathbf{a}_i \hookleftarrow \mathbb{Z}_q^{n_1} \text{ for } i\in[1,\delta] \\
\rho_i \hookleftarrow \{0,1\}^{ld} & \mathbf{r}_{\mathsf{id}_i} \hookleftarrow \mathbb{Z}_q^{ld}, \forall i\in[1,\delta] \\
\phi^{(1)}_{i,\mathbf{z}}, \ldots, \phi^{(p_1)}_{i,\mathbf{z}} \hookleftarrow \mathcal{P} & \mathbf{r}^{(1)}_{i,\mathbf{z}}, \ldots, \mathbf{r}^{(p_1)}_{i,\mathbf{z}} \hookleftarrow \mathbb{Z}_q^{3m_1(2ld+1)}, \forall i\in[1,\delta] \\
\phi^{(1)}_{0,\mathbf{z}}, \ldots, \phi^{(p_1)}_{0,\mathbf{z}} \hookleftarrow \overline{\mathcal{P}}, & \mathbf{r}^{(1)}_{0,\mathbf{z}}, \ldots, \mathbf{r}^{(p_1)}_{0,\mathbf{z}} \hookleftarrow \mathbb{Z}_q^{3m_1(2d+1)} \\
\phi_{i,\mathbf{a}} \hookleftarrow \mathcal{S}_{2n_1\lceil\log q_1\rceil}, & \mathbf{r}_{i,\mathbf{a}} \hookleftarrow \mathbb{Z}_q^{2n_1\lceil\log q\rceil} \ \text{ for } i\in[1,\delta] \\
\phi_{\mathsf{id},\mathbf{B}} \hookleftarrow \mathcal{P}' & \mathbf{R}_{\mathsf{id},\mathbf{B}} \hookleftarrow \mathbb{Z}_q^{m_3\times 2n_3(2d+1)} \ \forall \mathsf{id}\in\mathsf{warr} \\
\phi^{(1)}_{\mathsf{id},\mathbf{E}}, \ldots, \phi^{(p_3)}_{\mathsf{id},\mathbf{E}} \hookleftarrow \mathcal{S}_{3k_3\times m_3} & \mathbf{R}^{(1)}_{\mathsf{id},\mathbf{E}}, \ldots, \mathbf{R}^{(p_3)}_{\mathsf{id},\mathbf{E}} \hookleftarrow \mathbb{Z}_q^{m_3\times 3k_3} \ \text{ for } \mathsf{id}\in\mathsf{warr} \\
\phi^{(1)}_{i,\mathbf{e}}, \ldots, \phi^{(p_2)}_{i,\mathbf{e}} \hookleftarrow \mathcal{S}_{3h}, & \mathbf{r}^{(1)}_{i,\mathbf{e}}, \ldots, \mathbf{r}^{(p_2)}_{i,\mathbf{e}} \hookleftarrow \mathbb{Z}_q^{3h}, \forall i\in[1,\delta]
\end{array}$$

Sim proceeds by computing the commitments as in the case that $\overline{CH} = 1$. Upon receiving the commitment, the verifier issues a challenge to Sim. If $CH = 1$, build $\mathsf{id}^*_i = [\mathsf{id}^*_1||\ldots|\mathsf{id}^*_{l-1}||\mathsf{uid}]$ and send response

$$\mathsf{RSP} := (\{\{T_{\rho_i}\circ\phi^{(j)}_{i,\mathbf{z}}(\mathbf{z}^{(j)}_i), T_{\rho_i}\circ\phi^{(j)}_{i,\mathbf{z}}(\mathbf{r}^{(j)}_{i,\mathbf{z}})^{\delta}_{i=1}, T_{\rho_0}\circ\phi^{(j)}_{0,\mathbf{z}}(\mathbf{z}^{(j)}_i), T_{\rho_0}\circ\phi^{(j)}_{0,\mathbf{z}}(\mathbf{r}^{(j)}_{0,\mathbf{z}})\}^{p_1}_{j=1},$$
$$\{\{\phi^{(j)}_{i,\mathbf{e}}(\mathbf{e}^{(j)}_i), \phi^{(j)}_{i,\mathbf{e}}(\mathbf{r}^{(j)}_{i,\mathbf{e}}),\}^{p_2}_{j=1}, \phi_{i,\mathbf{a}}(\mathbf{a}_i), \phi^{(j)}_{i,\mathbf{a}}(\mathbf{r}^{(j)}_{i,\mathbf{a}})\}^{\delta}_{i=1}, \{\{\phi^{(j)}_{\mathsf{id},\mathbf{E}}(\mathbf{E}^{(j)}_{\mathsf{id}}), \phi^{(j)}_{\mathsf{id},\mathbf{E}}(\mathbf{R}^{(j)}_{\mathsf{id},\mathbf{E}})\}^{p_3}_{j=1},$$
$$T_{\rho_i}(\mathsf{id}_i), T_{\rho_i}(\mathbf{r}_{\mathsf{id}_i}), T_{\rho_{\mathsf{id}}}\circ\phi_{\mathsf{id},\mathbf{B}}(\mathbf{B}_{\mathsf{id}}), T_{\rho_{\mathsf{id}}}\circ\phi_{\mathsf{id},\mathbf{B}}(\mathbf{R}_{\mathsf{id},\mathbf{B}})\}_{\mathsf{id}\in\mathsf{warr}})$$

If $CH = 2$, output $\perp$ and abort.
If $CH = 3$, send response as given in $(\overline{CH} = 1, CH = 3)$.

If $\overline{CH} = 3$, then: The simulator samples the permutations and vectors as in $\overline{CH} = 2$. It sends the commitment as in $\overline{CH} = 1$, with the exception of c_1 which it computes as follows. First, build $\mathbf{a} := [\mathbf{a}_1||..||\mathbf{a}_\delta]^T$, $\mathbf{r}_\mathbf{a} := [\mathbf{r}_{1,\mathbf{a}}||..||\mathbf{r}_{\delta,\mathbf{a}}]^T$, Then compute

$$\begin{aligned} c_1 = \mathsf{COM}\Bigg(& \{\{\phi^{(j)}_{i,\mathbf{z}},\}^{\delta}_{i=1},\phi^{(j)}_{0,\mathbf{z}}\}^{p_1}_{j=1},\{\phi_{\mathsf{id}},\phi_{\mathsf{id},\mathbf{B}},\{\phi^{(j)}_{\mathsf{id},\mathbf{E}}\}^{p_3}_{j=1}\}_{\mathsf{id}\in\mathsf{warr}},\{\{\phi^{(j)}_{i,\mathbf{e}}\}^{p_2}_{j=1},\phi_{i,\mathbf{a}},\rho_i\}^{\delta}_{i=1},\rho_0, \\ & \overline{\mathbf{A}}^*\cdot\left(\sum_{j=1}^{p_1}\beta_1^{(j)}(\mathbf{z}_0^{(j)}+\mathbf{r}^{(j)}_{0,\mathbf{z}})\right),\Bigg\{\mathbf{A}^*\left(\sum_{j=1}^{p_1}\beta_1^{(j)}\cdot(\mathbf{z}_i^{(j)}+\mathbf{r}^{(j)}_{i,\mathbf{z}})\right) \\ & -\mathbf{G}'(\mathbf{a}_i+\mathbf{r}_{i,\mathbf{a}}); \quad \mathbf{P}^*\left(\sum_{j=1}^{p_2}\beta_2^{(j)}(\mathbf{e}_i^{(j)}+\mathbf{r}^{(j)}_{i,\mathbf{e}})\right)+\mathbf{Q}^*(\mathsf{id}_i+\mathbf{r}_{\mathsf{id}_i})\Bigg\}^{\delta}_{i=1}, \\ & \Bigg\{(\mathbf{B}_{\mathsf{id}}+\mathbf{R}_{\mathsf{id},\mathbf{B}})\mathbf{R}^*+\sum_{j=1}^{p_3}\beta_3^{(j)}(\mathbf{E}^{(j)}_{\mathsf{id}}+\mathbf{R}^{(j)}_{\mathsf{id},\mathbf{E}})\Bigg\}_{\mathsf{id}\in\mathsf{warr}},\langle\mathbf{G}^*\cdot(\mathbf{a}+\mathbf{r}_a),\mathbf{p}\rangle\Bigg) \end{aligned}$$

After receving the challenge, Sim responds as follows:
If $CH = 1$, send response as given in $\overline{CH} = 2, CH = 1$.
If $CH = 2$, send response as given in $\overline{CH} = 1, CH = 2$.
If $CH = 3$, output $\perp$ and abort.

The commitment scheme COM is statistically hiding, therefore the distribution of the commitments are statistically close to those in a real running of the protocol. We note the protocol only aborts when $\overline{CH} = CH$ and thus probability the simulation fails is $1/3$. When the protocol does not fail, we see that we have an accepting transcript obtained by Sim and V that is statistically close to that of an honest prover. Thus, we have constructed a simulator that successfully emulates honest P that succeeds with probability $2/3$, which proves our protocol has the zero-knowledge property.

Argument of Knowledge. We will show that, if COM is a computationally binding commitment scheme, then the protocol is an argument of knowledge. To show this, it suffices to show that it has the special soundness property, that is, given a commitment CMT and 3 valid responses RSP_1,RSP_2 and RSP_3 (to each of the 3 challenges respectively), then there exists an extractor that can output a witness in polynomial time. Since all 3 responses satisfy the verification equations, we have that:

$$\begin{aligned} & \forall j\in[1,...,p_1], i\in[1,\delta] : \mathbf{t}^{(j)}_{i,\mathbf{z}}\in\mathsf{SecretExt}_\beta(\mathsf{id}_i) \ \wedge \ \mathbf{t}^{(j)}_{0,\mathbf{z}}\in\mathsf{SecretExt}_\beta(\mathsf{uid}) \\ & \forall j\in[1,...,p_2], i\in[1,\delta] : \mathbf{t}^{(j)}_{i,\mathbf{e}}\in\mathsf{B}_{3h} \\ & \forall j\in[1,...,p_3], \forall\mathsf{id}\in\mathsf{warr} : \mathbf{T}^{(j)}_{\mathsf{id},\mathbf{E}}\in\mathsf{B}_{m_3\times k_3} \\ & \quad \forall i\in[1,\delta] : \mathbf{t}_{i,\mathsf{id}} = d_i^{(1)}, \mathbf{T}_{\mathsf{id},\mathbf{B}}\in\overline{\mathsf{SecretExt}_\beta}(\mathsf{id}) \\ & c_1 = \mathsf{COM}\Bigg(\{\{\psi^{(j)}_{i,\mathbf{z}},\}^{\delta}_{i=1},\psi^{(j)}_{0,\mathbf{z}}\}^{p_1}_{j=1},\{\psi_{\mathsf{id},\mathbf{B}},\{\psi^{(j)}_{\mathsf{id},\mathbf{E}}\}^{p_3}_{j=1}\}_{\mathsf{id}\in\mathsf{warr}}, \\ & \qquad \{\{\psi^{(i)}_{i,\mathbf{e}}\}^{p_2}_{j=1},\psi_{i,\mathbf{a}},d_i^{(2)}\}^{\delta}_{i=1},d_0^{(2)},\overline{\mathbf{A}}^*\left(\sum_{j=1}^{p_1}\beta_1^{(j)}\mathbf{w}^{(j)}_{0,\mathbf{z}}\right)-\mathbf{u}, \\ & \qquad \Bigg\{\mathbf{A}^*\left(\sum_{j=1}^{p_1}\beta_1^{(j)}\cdot\mathbf{w}^{(j)}_{i,\mathbf{z}}\right)-\overline{\mathbf{G}}^*\mathbf{w}_{i,\mathbf{a}};\mathbf{P}^*\left(\sum_{j=1}^{p_2}\beta_2^{(j)}\mathbf{w}^{(j)}_{i,\mathbf{e}}\right)+\mathbf{Q}^*\mathbf{w}_{i,\mathbf{d}}-\mathbf{f}_i\Bigg\}^{\delta}_{i=1}, \\ & \qquad \Bigg\{\mathbf{W}_{\mathsf{id},\mathbf{B}}\mathbf{R}^*+\sum_{j=1}^{p_3}\beta_3^{(j)}\mathbf{W}^{(j)}_{\mathsf{id},\mathbf{E}}-\mathbf{C}_{\mathsf{id}}\Bigg\}_{\mathsf{id}\in\mathsf{warr}},\langle\mathbf{G}^*\mathbf{w}_a,\mathbf{p}\rangle-1\Bigg) \end{aligned}$$

$$
\begin{aligned}
&=\mathsf{COM}\Bigg(\{\{\theta_{i,\mathbf{z}}^{(j)}\}_{i=1}^{\delta},\theta_{0,\mathbf{z}}^{(j)}\}_{j=1}^{p_1},\{\theta_{\mathsf{id},\mathbf{B}},\{\theta_{\mathsf{id},\mathbf{E}}^{(j)}\}_{j=1}^{p_3}\}_{\mathsf{id}\in\mathsf{warr}},\{\{\theta_{i,\mathbf{z}}^{(j)},\theta_{i,\mathbf{e}}^{(j)}\}_{j=1}^{p_2},\theta_{i,\mathbf{a}},d_i^{(3)}\}_{i=1}^{\delta},d_0^{(3)},\\
&\qquad\overline{\mathbf{A}}^*\left(\sum_{j=1}^{p_1}\beta_1^{(j)}\mathbf{y}_{0,\mathbf{z}}^{(j)}\right),\left\{\mathbf{A}^*\left(\sum_{j=1}^{p_1}\beta_1^{(j)}\mathbf{y}_{i,\mathbf{z}}^{(j)}\right)-\mathbf{G}'\mathbf{y}_{i,\mathbf{a}};\mathbf{P}^*\left(\sum_{j=1}^{p_2}\beta_2^{(j)}\mathbf{y}_{i,\mathbf{e}}^{(j)}\right)+\mathbf{Q}^*\mathbf{y}_{i,\mathbf{d}}\right\}_{i=1}^{\delta},\\
&\qquad\left\{\mathbf{Y}_{\mathsf{id},\mathbf{B}}\mathbf{R}^*+\sum_{j=1}^{p_3}\beta_3^{(j)}\mathbf{Y}_{\mathsf{id},\mathbf{E}}^{(j)}\right\}_{\mathsf{id}\in\mathsf{warr}},\langle\mathbf{G}^*\mathbf{y}_a,\mathbf{p}\rangle\Bigg)\\
c_2&=\mathsf{COM}(\{\{\mathbf{v}_{i,\mathbf{z}}^{(j)}\}_{i=1}^{\delta},\mathbf{v}_{0,\mathbf{z}}^{(j)}\}_{j=1}^{p_1},\{\{\mathbf{v}_{i,\mathbf{e}}^{(j)}\}_{j=1}^{p_2},\mathbf{v}_{i,\mathbf{a}}\}_{i=1}^{\delta},\{\mathbf{v}_{\mathsf{id}},\mathbf{V}_{\mathsf{id},\mathbf{B}},\{\mathbf{V}_{\mathsf{id},\mathbf{E}}\}_{j=1}^{p_3}\}_{\mathsf{id}\in\mathsf{warr}})\\
&=\mathsf{COM}(\{\{T_{d_i^{(3)}}\circ\theta_{i,\mathbf{z}}^{(j)}(\mathbf{y}_{i,\mathbf{z}}^{(j)})\}_{i=1}^{\delta},T_{d_0^{(3)}}\circ\theta_{0,\mathbf{z}}^{(j)}(\mathbf{y}_{0,\mathbf{z}}^{(j)})\}_{j=1}^{p_1},\{\{\{\theta_{i,\mathbf{e}}^{(j)}(\mathbf{y}_{i,\mathbf{e}}^{(j)})\}_{j=1}^{p_2},\theta_{i,\mathbf{a}}(\mathbf{y}_{i,\mathbf{a}})\}_{i=1}^{\delta},\\
&\qquad\{T_{d_{\mathsf{id}}^{(3)}}(\mathbf{y}_{i,\mathsf{id}}),T_{d_{\mathsf{id}}^{(3)}}\circ\theta_{\mathsf{id},\mathbf{B}}(\mathbf{Y}_{\mathsf{id},\mathbf{B}}),\{\theta_{\mathsf{id},\mathbf{E}}(\mathbf{Y}_{\mathsf{id},\mathbf{E}}^{(j)})\}_{j=1}^{p_3}\}_{\mathsf{id}\in\mathsf{warr}})\\
c_3&=\mathsf{COM}(\{\{\mathbf{t}_{i,\mathbf{z}}^{(j)}+\mathbf{v}_{i,\mathbf{z}}^{(j)}\}_{i=1}^{\delta},\mathbf{t}_{0,\mathbf{z}}^{(j)}+\mathbf{v}_{0,\mathbf{z}}^{(j)}\}_{j=1}^{p_1},\{\{\mathbf{t}_{i,\mathbf{e}}^{(j)}+\mathbf{v}_{i,\mathbf{e}}^{(j)}\}_{j=1}^{p_2},\mathbf{t}_{i,\mathbf{a}}+\mathbf{v}_{i,\mathbf{a}}\}_{i=1}^{\delta},\\
&\qquad\{\mathbf{t}_{\mathsf{id}}+\mathbf{v}_{\mathsf{id}},\mathbf{T}_{\mathsf{id},\mathbf{B}}+\mathbf{V}_{\mathsf{id},\mathbf{B}},\{\mathbf{T}_{\mathsf{id},\mathbf{E}}+\mathbf{V}_{\mathsf{id},\mathbf{E}}\}_{j=1}^{p_3}\}_{\mathsf{id}\in\mathsf{warr}})\\
&=\mathsf{COM}(\{\{T_{d_i^{(2)}}\circ\psi_{i,\mathbf{z}}^{(j)}(\mathbf{w}_{i,\mathbf{z}}^{(j)})\}_{i=1}^{\delta},T_{d_i^{(2)}}\circ\psi_{0,\mathbf{z}}^{(j)}(\mathbf{w}_{0,\mathbf{z}}^{(j)})\}_{j=1}^{p_1},\{\{\psi_{i,\mathbf{e}}^{(j)}(\mathbf{w}_{i,\mathbf{e}}^{(j)})\}_{j=1}^{p_2},\\
&\qquad\psi_{i,\mathbf{a}}(\mathbf{w}_{i,\mathbf{a}})\}_{i=1}^{\delta},\{T_{d_i^{(2)}}(\mathbf{w}_{i,\mathsf{id}}),T_{d_i^{(2)}}\circ\psi_{\mathsf{id},\mathbf{B}}^{(j)}(\mathbf{W}_{\mathsf{id},\mathbf{B}}),\{\psi_{\mathsf{id},\mathbf{E}}(\mathbf{W}_{\mathsf{id},\mathbf{E}}^{(j)})\}_{j=1}^{p_3}\}_{\mathsf{id}\in\mathsf{warr}})
\end{aligned}
$$

The computational binding property of COM implies that:

$$
\begin{cases}
\forall j\in[1,\dots,p_1],\\
\psi_{0,\mathbf{z}}^{(j)}=\theta_{0,\mathbf{z}}^{(j)},T_{d_0^{(2)}}\circ\psi_{0,\mathbf{z}}(\mathbf{w}_{0,\mathbf{z}}^{(j)})=\mathbf{t}_{0,\mathbf{z}}^{(j)}+\mathbf{v}_{0,\mathbf{z}}^{(j)},T_{d_0^{(2)}}\circ\psi_{0,\mathbf{z}}(\mathbf{y}_{0,\mathbf{z}}^{(j)})=\mathbf{v}_{0,\mathbf{z}}^{(j)}\\
\forall j\in[1,\dots,p_1],i\in[1,\delta]:\\
\psi_{i,\mathbf{z}}^{(j)}=\theta_{i,\mathbf{z}}^{(j)},T_{d_i^{(2)}}\circ\psi_{i,\mathbf{z}}(\mathbf{w}_{i,\mathbf{z}}^{(j)})=\mathbf{t}_{i,\mathbf{z}}^{(j)}+\mathbf{v}_{i,\mathbf{z}}^{(j)},T_{d_i^{(2)}}\circ\psi_{i,\mathbf{z}}(\mathbf{y}_{i,\mathbf{z}}^{(j)})=\mathbf{v}_{i,\mathbf{z}}^{(j)}\\
\forall j\in[1,\dots,p_2],i\in[1,\delta]:\\
\psi_{i,\mathbf{e}}^{(j)}=\theta_{i,\mathbf{e}}^{(j)},\psi_{i,\mathbf{e}}^{(j)}(\mathbf{w}_{i,\mathbf{e}}^{(j)})=\mathbf{t}_{i,\mathbf{e}}^{(j)}+\mathbf{v}_{i,\mathbf{e}}^{(j)},\theta_{i,\mathbf{e}}^{(j)}(\mathbf{y}_{i,\mathbf{e}}^{(j)})=\mathbf{v}_{i,\mathbf{e}}^{(j)}\\
\forall i\in[1,\delta]:\\
\psi_{i,\mathbf{a}}=\theta_{i,\mathbf{a}},\psi_{i,\mathbf{a}}(\mathbf{w}_{i,\mathbf{a}})=\mathbf{t}_{i,\mathbf{a}}+\mathbf{v}_{i,\mathbf{a}},\theta_{i,\mathbf{a}}(\mathbf{y}_{i,\mathbf{a}})=\mathbf{v}_{i,\mathbf{a}}\\
\forall\mathsf{id}\in\mathsf{warr}:\\
\mathbf{w}_{i,\mathsf{id}}=\mathbf{t}_{i,\mathsf{id}}+\mathbf{v}_{i,\mathsf{id}},\mathbf{y}_{i,\mathsf{id}}=\mathbf{v}_{i,\mathsf{id}}\\
\forall\mathsf{id}\in\mathsf{warr},j\in[1,p_3]:\\
\psi_{\mathsf{id},\mathbf{E}}^{(j)}=\theta_{\mathsf{id},\mathbf{E}}^{(j)},\psi_{\mathsf{id},\mathbf{E}}^{(j)}(\mathbf{W}_{\mathsf{id},\mathbf{E}}^{(j)})=\mathbf{T}_{\mathsf{id},\mathbf{E}}^{(j)}+\mathbf{V}_{\mathsf{id},\mathbf{E}}^{(j)},\theta_{\mathsf{id},\mathbf{E}}^{(j)}(\mathbf{Y}_{i,\mathsf{id}}^{(j)})=\mathbf{V}_{\mathsf{id},\mathbf{E}}^{(j)}\\
\psi_{\mathsf{id},\mathbf{B}}=\theta_{\mathsf{id},\mathbf{B}},T_{d_i^{(2)}}\circ\psi_{\mathsf{id},\mathbf{B}}(\mathbf{W}_{\mathsf{id},\mathbf{B}})=\mathbf{T}_{\mathsf{id},\mathbf{B}}+\mathbf{V}_{\mathsf{id},\mathbf{B}},T_{d_i^{(2)}}\circ\psi_{\mathsf{id},\mathbf{B}}(\mathbf{Y}_{\mathsf{id},\mathbf{B}})=\mathbf{V}_{\mathsf{id},\mathbf{B}}\\
\overline{\mathbf{A}}^*\left(\sum_{j=1}^{p_1}\beta_1^{(j)}(\mathbf{w}_{0,\mathbf{z}}^{(j)}-\mathbf{y}_{0,\mathbf{z}}^{(j)})\right)=\mathbf{u}\quad\bmod q_1\\
\forall i\in[1,\delta]:\mathbf{A}^*\left(\sum_{j=1}^{p_1}\beta_1^{(j)}(\mathbf{w}_{i,\mathbf{z}}^{(j)}-\mathbf{y}_{i,\mathbf{z}}^{(j)})\right)=\mathbf{G}'\cdot(\mathbf{w}_{a_i}-\mathbf{y}_{a_i})\quad\bmod q_1\\
\forall i\in[1,\delta]:\mathbf{P}^*\left(\sum_{j=1}^{p_2}\beta_2^{(j)}(\mathbf{w}_{i,\mathbf{e}}^{(j)}-\mathbf{y}_{i,\mathbf{e}}^{(j)})\right)+\mathbf{Q}^*(\mathbf{w}_{\mathsf{id}_i}-\mathbf{y}_{\mathsf{id}_i})=\mathbf{f}_{\mathsf{id}_i}\quad\bmod q_2\\
\langle\mathbf{G}^*\cdot(\mathbf{w}_a-\mathbf{y}_a),\mathbf{p}\rangle=1\quad\text{mod }q_1,\text{ for }\mathbf{y}_a=[\mathbf{y}_{a_1}\|\dots\|\mathbf{y}_{a_\delta}],\mathbf{w}_a=[\mathbf{w}_{a_1}\|\dots\|\mathbf{w}_{a_\delta}]\\
\forall\mathsf{id}\in\mathsf{warr}:(\mathbf{W}_{\mathsf{id},\mathbf{B}}-\mathbf{Y}_{\mathsf{id},\mathbf{B}})\mathbf{R}^*+\sum_{j=1}^{p_3}\beta_3^{(j)}(\mathbf{W}_{\mathsf{id},\mathbf{E}}-\mathbf{Y}_{\mathsf{id},\mathbf{E}})=\mathbf{C}_{\mathsf{id}}\quad\bmod q_3
\end{cases}
$$

We now observe that:

$$
\begin{aligned}
T_{d_i^{(2)}}(\mathbf{w}_{i,\mathsf{id}}-\mathbf{y}_{i,\mathsf{id}})&=\mathbf{t}_{i,\mathsf{id}}=d_i^{(1)}\\
T_{d_{\mathsf{uid}}^{(1)}}\circ\phi_{0,\mathbf{z}}^{(j)}(\mathbf{w}_{0,\mathbf{z}}^{(j)}-\mathbf{y}_{0,\mathbf{z}}^{(j)})&=\mathbf{t}_{0,\mathbf{z}}^{(j)}\in\mathsf{SecretExt}_\beta(d_{\mathsf{uid}}^{(1)})\\
T_{d_i^{(2)}}\circ\phi_{i,\mathbf{z}}^{(j)}(\mathbf{w}_{i,\mathbf{z}}^{(j)}-\mathbf{y}_{i,\mathbf{z}}^{(j)})&=\mathbf{t}_{i,\mathbf{z}}^{(j)}\in\mathsf{SecretExt}_\beta(d_i^{(1)})\\
T_{d_{\mathsf{id}}^{(2)}}\circ\phi_{\mathsf{id},\mathbf{B}}(\mathbf{w}_{\mathsf{id},\mathbf{B}}-\mathbf{Y}_{\mathsf{id},\mathbf{B}})&=\mathbf{T}_{\mathsf{id},\mathbf{B}}\in\overline{\mathsf{SecretExt}_\beta}(d_{\mathsf{id}}^{(1)})
\end{aligned}
$$

Then, it follows that:

$$\begin{aligned}\mathbf{w}_{\mathsf{id}}-\mathbf{y}_{\mathsf{id}}=d_{\mathsf{id}}^{(2)}\oplus d_{\mathsf{id}}^{(1)}\\ \phi_{0,\mathbf{z}}^{(j)}(\mathbf{w}_{0,\mathbf{z}}^{(j)}-\mathbf{y}_{0,\mathbf{z}}^{(j)})\in\mathsf{SecretExt}_{\beta}(d_{\mathsf{uid}}^{(1)}\oplus d_{\mathsf{uid}}^{(1)})\\ \phi_{i,\mathbf{z}}^{(j)}(\mathbf{w}_{i,\mathbf{z}}^{(j)}-\mathbf{y}_{i,\mathbf{z}}^{(j)})\in\mathsf{SecretExt}_{\beta}(d_{i}^{(1)}\oplus d_{i}^{(2)})\\ \phi_{\mathsf{id},\mathbf{B}}(\mathbf{W}_{\mathsf{id},\mathbf{B}}-\mathbf{Y}_{\mathsf{id},\mathbf{B}})\in\overline{\mathsf{SecretExt}_{\beta}}(d_{\mathsf{id}}^{(1)}\oplus d_{\mathsf{id}}^{(2)})\end{aligned}$$

Let $\mathsf{id}^*:=\mathbf{w}_{i,\mathsf{id}}-\mathbf{y}_{i,\mathsf{id}},\mathbf{z}_0^{(j)*}:=\mathbf{w}_{0,\mathbf{z}}^{(j)}-\mathbf{y}_{0,\mathbf{z}}^{(j)},\mathbf{z}_i^{(j)*}:=\mathbf{w}_{i,\mathbf{z}}^{(j)}-\mathbf{y}_{i,\mathbf{z}}^{(j)},\mathbf{B}_{\mathsf{id}}^*:=\mathbf{W}_{\mathsf{id},\mathbf{B}}-\mathbf{Y}_{\mathsf{id},\mathbf{B}}$ and $\mathsf{id}:=d_{\mathsf{id}}^{(2)}\oplus d_{\mathsf{id}}^{(1)},\mathsf{id}_i:=d_i^{(2)}\oplus d_i^{(1)}$ and, since the permutations ϕ preserve structure, we see that:

$$\begin{aligned}\mathsf{id}_i^*=d_{\mathsf{id}}^{(2)}\oplus d_{\mathsf{id}}^{(1)}\\ \mathbf{z}_0^{(j)*}\in\mathsf{SecretExt}_{\beta}(\mathsf{id}_i^*)\\ \mathbf{z}_i^{(j)*}\in\mathsf{SecretExt}_{\beta}(\mathsf{id}_i^*)\\ \mathbf{B}_{\mathsf{id}}^*\in\overline{\mathsf{SecretExt}_{\beta}}(\mathsf{id}^*)\end{aligned}$$

Moreover;

$$\begin{cases}\overline{\mathbf{A}}^*\left(\sum_{j=1}^{p_1}\beta_1^{(j)}\cdot\mathbf{z}_0^{(j)}\right)=\mathbf{u}\mod q_1\\ \mathbf{A}^*\left(\sum_{j=1}^{p_1}\beta_1^{(j)}\cdot\mathbf{z}_i^{(j)}\right)=\mathbf{G}'\mathbf{a}_i\mod q_1\text{ for }i\in[1,\delta]\\ \mathbf{P}^*\left(\sum_{j=1}^{p_2}\beta_2^{(j)}\cdot\mathbf{e}_i^{(j)}\right)+\mathbf{Q}^*\mathsf{id}_i=\mathbf{f}_i\mod q_2\text{ for }i\in[1,\delta]\\ \langle\mathbf{G}^*\cdot\mathbf{a}^*,\mathbf{p}\rangle=1\mod q_1\\ \mathbf{B}_{\mathsf{id}}\mathbf{R}^*+\sum_{j=1}^{p_3}\beta_3^{(j)}\mathbf{E}_{\mathsf{id}}^{(j)}=\mathbf{C}_{\mathsf{id}}\mod q_3,\qquad\forall\mathsf{id}\in\mathsf{warr}\end{cases}$$

For $\mathbf{z}_0^*=\sum_{j=1}^{p_1}\beta_1^{(j)}\mathbf{z}_0^{(j)*}\in\mathbb{Z}_{q_1}^{3m_1(2d+1)}$, then we have that $\|\mathbf{z}_0^*\|_\infty\leqslant\sum_{j=1}^{p_1}\beta_1^{(j)}\left\|\mathbf{z}_0^{(j)*}\right\|_\infty\leqslant\beta_1$ and $\mathbf{A}^*\mathbf{z}_0^*=\mathbf{u}$ $\mod q_1$. Since, $\mathbf{z}_0^{(j)}\in\mathsf{SecretExt}_\beta(\mathsf{uid})$ then so is $\mathbf{z}_0^*$. Now let $\mathbf{z}_0$ be obtained from $\mathbf{z}_0^*$ by removing the last $2m_1$ rows of each of the $2d+1$ blocks of size $3m_1$ that comprise $\mathbf{z}_0^*$. Then, we note that $\|\mathbf{z}_0\|_\infty\leqslant\beta_1$ and $[\mathbf{A}||\mathbf{A}_1^{(0)}||\mathbf{A}_1^{(1)}||...||\mathbf{A}_d^{(0)}||\mathbf{A}_d^{(1)}]\mathbf{z}_0=\mathbf{u}\mod q_1$.

For $\mathbf{z}_i^*=\sum_{j=1}^{p_1}\beta_1^{(j)}\mathbf{z}_i^{(j)*}\in\mathbb{Z}_{q_1}^{3m_1(2ld+1)}$, then we have that $\|\mathbf{z}_i^*\|_\infty\leqslant\sum_{j=1}^{p_1}\beta_1^{(j)}\left\|\mathbf{z}_i^{(j)*}\right\|_\infty\leqslant\beta_1$ and $\mathbf{A}^*\mathbf{z}_i^*=\mathbf{a}_i$ $\mod q_1$. Since, $\mathbf{z}_i^{(j)}\in\mathsf{SecretExt}_\beta(\mathsf{id}_i)$ then so is $\mathbf{z}_i^*$. Now let $\mathbf{z}_i$ be obtained from $\mathbf{z}_i^*$ by removing the last $2m_1$ rows of each of the $2ld+1$ blocks of size $3m_1$ that comprise $\mathbf{z}_l^*$. Then, we note that $\|\mathbf{z}_l\|_\infty\leqslant\beta_1$ and $[\mathbf{A}||\mathbf{A}_1^{(0)}||\mathbf{A}_1^{(1)}||...||\mathbf{A}_{ld}^{(0)}||\mathbf{A}_{ld}^{(1)}]\mathbf{z}_0=\mathbf{a}_i\mod q_1$.

Similarly, we now let $\mathbf{e}_i^*:=\sum_{j=i}^{p_2}\beta_2^{(j)}\mathbf{e}_i^{(j)}\in\mathbb{Z}_q^{3h(2d+1)}$, and we use the fact that $\mathbf{e}_i^{(j)}\in\mathsf{B}_{3lh(2d+1)}$ to bound $\left\|\mathbf{e}_i^{(j)}\right\|_\infty\leqslant1$, which implies $\|\mathbf{e}_i^*\|_\infty\leqslant\sum_{j=i}^{p_2}\beta_2^{(j)}\left\|\mathbf{e}_i^{(j)}\right\|_\infty\leqslant\beta_2\cdot1=\beta_2$. Substituting in $\mathbf{e}_i$, we also have:

$$\mathbf{P}^*\mathbf{e}_i^*+\mathbf{Q}^*\mathsf{id}_i^*=\mathbf{f}_{\mathsf{id}_i}\mod q\text{ for }i\in[1,\delta]$$

Now obtain $\mathbf{e}_i$ by dropping the last $2h$ coordinates from $\mathbf{e}_i^*$, and the last d coordinates from id^*, then we conclude that $\|\mathbf{e}_i\|_\infty\leqslant\beta_2$ and $\mathbf{P}\mathbf{e}_i+\mathbf{Q}\mathsf{id}_i=\mathbf{f}_{\mathsf{id}_i}\mod q,\forall i\in[1,\delta]$, as required.

Next, since $\mathbf{B}_{\mathsf{id}}^*\in\overline{\mathsf{SecretExt}_\beta}(\mathsf{id})$, obtain $\mathbf{B}_{\mathsf{id}}$ from $\mathbf{B}^*$ by dropping the last n_3 columns in each of the $2d+1$ blocks. Let $\mathbf{E}_{\mathsf{id}}^*=\sum_{j=1}^{p_3}\beta_3^{(j)}\mathbf{E}_{\mathsf{id}}^{(j)}$. Note that $\|\mathbf{E}_{\mathsf{id}}^*\|_\infty\leqslant\sum_{j=1}^{p_3}\beta_3^{(j)}\left\|\mathbf{E}_{\mathsf{id}}^{(j)}\right\|_\infty\leqslant\beta_3\cdot1=\beta_3$. Finally, let $\mathbf{E}_{\mathsf{id}}$ be obtained from $\mathbf{E}_{\mathsf{id}}^*$ by removing the final $2m$ columns. Thus, for each $\mathsf{id}\in\mathsf{warr}$, we have constructed an $(\mathbf{B}_{\mathsf{id}},\mathbf{E}_{\mathsf{id}})$ such that $\mathbf{B}_{\mathsf{id}}$ is a binary matrix, $\|\mathbf{E}_{\mathsf{id}}\|_\infty\leqslant\beta_3$ and $\mathbf{C}_{\mathsf{id}}=\mathbf{B}_{\mathsf{id}}[\mathbf{R}||\mathbf{R}_1^{(0)}||\mathbf{R}_1^{(1)}||...||\mathbf{R}_d^{(0)}||\mathbf{R}_d^{(1)}]+\mathbf{E}_{\mathsf{id}}$ $\mod q_3$.

We obtain id from id^* by removing the last d elements of the vector. Finally, obtain $\mathbf{a}_i$ from $\mathbf{a}_i^* := \mathbf{w}_{a_i} - \mathbf{y}_{a_i}$ by removing the final $n_1\lceil \log q_1 \rceil$ elements and construct $\mathbf{a}' := [\mathbf{a}_1||...||\mathbf{a}_\delta]$. Since $\langle \mathbf{G}^* \cdot \mathbf{a}', \mathbf{p} \rangle = 1 \mod q_1$, then $\mathbf{a} := \mathbf{G}^* \mathbf{a}'$ satisfies $\langle \mathbf{a}, \mathbf{p} \rangle = 1$.

Thus, upon receiving 3 correct responses, an extractor is able to output an accepting witness $(\{\mathbf{z}_i, \mathbf{e}_i, \mathbf{a}_i\}_{i=1}^{\delta}, \{\mathbf{B}_{\mathsf{id}}, \mathbf{E}_{\mathsf{id}}, \mathsf{id}\}_{\mathsf{id} \in \mathsf{warr}})$ in polynomial time for the relation $\mathcal{R}_5$, hence we have shown the protocol is an argument of knowledge. □

C Proof of Correctness

Theorem 3. Our VLR-HABS construction given in Figures 5 to 7 is correct.

Proof. With overwhelming probability and honest signer is able to obtain a valid witness for ZKAoK. Then, due to the perfect completeness property of this protocol, conditions (1) and (2) are true (see defn. of Correctness). Next, we note similarly that the correctness of the encryption scheme ensures that the tracing authority is able to compute a valid decryption key for IBE-GPV. Again, perfect completeness of this protocol ensures that conditions (4) also holds.

Then, Lemma 1 shows that it is probabilistically negligible that there $\exists \mathbf{C}$ s.t. $\mathbf{C}\mathbf{y} \leqslant n\beta^2$ for any $\mathbf{y} \in \mathsf{RevokeList}$ and any $\mathbf{C}$ sampled at random from a uniform distribution. Since the revocation tokens are statistically close to uniform, a mild adaptation to the proof allows us to conclude that a revocation token generated by an honest user (i.e. not in RevokeList) does not yield a small vector that would fail verification when post-multiplied by a $\mathbf{y} \in \mathsf{RevokeList}$, provided $|\mathsf{RevokeList}| \leqslant \mathsf{poly}(n)$. □

D Security Proofs of VLR-HABS

D.1 Proof of Path Anonymity

Lemma 3. *Our* VLR-HABS *construction satisfies Path Anonymity if the* $\mathsf{SIS}_{n_4,m_4,q_4,\beta_4}$*,* $\mathsf{SIS}_{n',m',q',1}$*,* $\mathsf{LWE}_{n_2,m_2,q_2,\chi_2}$ *and* $\mathsf{LWE}_{n_3,m_3,q_3,\chi_3}$ *assumptions hold and* $\mathcal{H}_0$ *is collision resistant.*

Proof. $\mathcal{G}_0$: Let this be defined by the experiment defined in Figure 2. The success probabilities of the adversaries are equal.

$$\Pr[\mathcal{G}_0 = 1] = \Pr[\mathsf{Exp}^{\mathsf{pa}\text{-}b}_{\mathsf{VLR\text{-}HABS},\mathcal{A}} = 1]$$

$\mathcal{G}_1$: Define this game as $\mathcal{G}_0$, with the only difference that we move the check "$\mathcal{A}_2$ did not query $\mathcal{O}_{\mathrm{Tr}}(\mathsf{sk}_{\mathsf{TA}}, (\mathsf{m}, \Psi, \sigma_b))$" to the trace oracle. In particular, it aborts if $\sigma = \sigma_b$, where σ_b is the challenge signature and σ is a signature the adversary submits to the oracle. The success probability of the adversay is unaffected by this change, thus we have:

$$\Pr[\mathcal{G}_1 = 1] = \Pr[\mathcal{G}_0 = 1]$$

$\mathcal{G}_2$: This game is identical to $\mathcal{G}_1$ with the exception that we replace ZKAoK with its simulator constructed in the proof of Theorem 1, and programming the random oracle $\mathcal{H}_1$ accordingly. That is, any call to the signing oracle receives simulated proofs as part of the signature. The adversary cannot distinguish $\mathcal{G}_2$ from $\mathcal{G}_1$ unless it can break the zero-knowledge property of ZKAoK as proven in Theorem 1. To see this, the challenger for the path anonymity experiment $\mathcal{B}$, sets up the game according to $\mathcal{G}_1$. It challenges an adversary $\mathcal{A}$ to determine the challenge bit b from the VLR-HABS experiment, and must use this to construct a polynomial-time attack against the zero-knowledge property of ZKAoK. After it invokes the experiment, $\mathcal{B}$ waits for $\mathcal{A}$ to return on line 2. By programming the random oracle $\mathcal{H}_1$, it can successfully execute the ZKAoK simulator. The simulated proof is independent of the witness, is therefore independent of b. If the difference is success probability between $\mathcal{G}_1$ and $\mathcal{G}_2$ is non-negligible, then $\mathcal{B}$ can build a distinguisher against the zero-knowledge simulator for ZKAoK. However, this property is implied by the statistically hiding property of the underlying commitment scheme, which in turn is implied by the $\mathsf{SIS}_{n',m',q',\beta'}$ assumption. Thus, we can bound the winning probabilities as:

$$|\Pr[\mathcal{G}_2=1]-\Pr[\mathcal{G}_1=1]|\leqslant \mathsf{Adv}_{\mathsf{SIS}_{n',m',q',\beta'}}$$

$\mathcal{G}_3$: We now further restrict the check made by the trace oracle. Precisely, the trace oracle aborts if $(\mathbf{f},\mathbf{C},\mathsf{ovk})=(\mathbf{f}_b,\mathbf{C}_b,\mathsf{ovk}_b)$. The adversary can distinguish between the two games if and only if it is able to produce a forgery of the one-time signatures with respect to ovk. To create a forgery against the chosen-message attack game, $\mathcal{B}$ invokes its own game and challenges $\mathcal{A}$ against path anonymity according to $\mathcal{G}_3$. It is able to answer all oracle queries. $\mathcal{B}$ waits for $\mathcal{A}$ to invoke the challenge phase of the experiment, when instead of creating the VLR-HABS signature itself, it does all the steps in algorithm Sign except it calls its EUF-CMA game for a challenge key pair, uses these values for $(\mathsf{osk},\mathsf{ovk})$, regardless of b. It submits the message $\mathsf{m}_o:=(\mathsf{osk},\mathcal{H}_2(\mathsf{m},\Psi,\pi,\{\mathbf{f}_i\}_{i=1}^{\delta},\{\mathbf{C}_i\}_{i=1}^{l\delta}))$ to its single-use signing oracle, and uses the output to complete the VLR-HABS signature which it forwards to $\mathcal{A}$. It then waits for $\mathcal{A}$ to submit a forgery to the tracing oracle. It is only able to win if it did not query the challenge signature. Thus, for *some* component of the message, $\mathsf{m}_o\neq\tilde{\mathsf{m}}_o$. Hence $\mathcal{B}$ submits the one-time signature from the forgery, $\tilde{\sigma_o}$ along with message $\tilde{\mathsf{m}}_o$ as a forgery to the one-time unforgeability and wins its game if $\mathcal{A}$ won $\mathcal{G}_3$. The Bonsai signature we use for OTS in this construction of VLR-HABS is EUF-CMA under the $\mathsf{SIS}_{n_4,m_4,q_4,\beta_4}$ assumption, thus we have:

$$|\Pr[\mathcal{G}_3=1]-\Pr[\mathcal{G}_2=1]|\leqslant \mathsf{Adv}_{\mathsf{SIS}_{n_4,m_4,q_4,\beta_4}}$$

$\mathcal{G}_4$: We restrict once more and check that $(\mathbf{f},\mathbf{C})\neq(\mathbf{f}_b,\mathbf{C}_b)$, that is, we have removed the equality check on the one-time verification key ovk. However, the trace oracle aborts if ovk from the IBE ciphertext $\mathbf{C}_b$ does not match that which was submitted to the oracle anyway so the success probabilities are preserved.

$$\Pr[\mathcal{G}_4=1]=\Pr[\mathcal{G}_3=1]$$

$\mathcal{G}_5$: In this hop, we use the pseudorandomness of LWE to replace the revocation tokens $\{\mathbf{C}_i\}_{i=1}^{l\delta}$ with samples from a uniform distribution via a series of hybrid games $\mathcal{G}_5^{(i)}$, where $\mathcal{G}_5^{(i)}:=\mathcal{G}_5^{(i-1)}$ except that in the i^{th} revocation token is replaced with a uniformly sampled random matrix $\mathbf{C}_i$. We construct an adversary against decisional binary LWE by initiating a challenger $\mathcal{B}$ for $\mathcal{G}_5^{(i)}$ who plays the role of adversary against the LWE challenge. In game i^*, $\mathcal{B}$ creates the challenge VLR-HABS signature according to the changes made up to $\mathcal{G}_4$, except that instead of creating revocation tokens $\mathbf{C}_i$ in the range $i\in[1,i^*-1]$, it uniformly samples a random matrix $\mathbf{C}\hookleftarrow\mathbb{Z}_q^{n_3\times m_3}$. For $i\in[i^*,l\delta]$ it creates the revocation tokens honestly as described in the Sign algorithm. For the special case that $i=i^*$, it uses its LWE challenge matrix. Note that if $\mathbf{C}_{i^*}$ is a random sample, then $\mathcal{G}_5^{(i^*)}=\mathcal{G}_5^{(i)}$ and if $\mathbf{C}_{i^*}$ is a proper LWE sample then we have that $\mathcal{G}_5^{(i^*)}=\mathcal{G}_5^{(i-1)}$. Thus, any difference in the success probabilities for an adversary against $\mathcal{G}_5^{(i-1)}$ and $\mathcal{G}_5^{(i)}$ can be used by $\mathcal{B}$ to build a distinguisher against the LWE property for binary secrets (recall that $\mathbf{B}\in\mathbb{B}^{k_3\times n_3}$). This has been shown is at least as hard as standard LWE problem for appropriate parameters (see [41]). In particular, we stress that $\mathcal{A}$ cannot compute a trapdoor for $\mathbf{R}$ as doing so would allow an adversary to build an LWE inverter as described in [2] and thus win game $\mathcal{G}_5^{(i)}$. Finally, we observe that $\mathcal{G}_5^{(0)}=\mathcal{G}_4$, and conclude that the probability of distinguishing each successive hybrid game is bound by an adversary against $\mathsf{LWE}_{n_3,m_3,q_3,\chi_3}$. Thus;

$$|\Pr[\mathcal{G}_5^{(i)}=1]-\Pr[\mathcal{G}_5^{(i-1)}=1]|\leqslant \mathsf{Adv}_{\mathsf{LWE}_{n_3,m_3,q_3,\chi_3}} \quad \text{and} \quad \Pr[\mathcal{G}_5^{(0)}=1]=\Pr[\mathcal{G}_4=1].$$

Now for sake of conciseness, denote $\mathcal{G}_6:=\mathcal{G}_5^{(l\delta)}$ and assume $\mathcal{H}_0$ is collision resistant. We note that we have reduced the indistinguishability of the challenge signatures for VLR-HABS to indistinguishability of the IBE ciphertexts $\mathbf{f}^{(0)}:=\{\mathbf{f}_i^{(0)}\}_{i=1}^{\delta}$ and $\mathbf{f}_1:=\{\mathbf{f}_i^{(1)}\}_{i=1}^{\delta}$ with respect to the tag $\mathcal{H}_0(\mathsf{ovk})$. We here on out assume $\mathcal{H}_0(\mathsf{ovk})$ is collision resistant, and thus cannot submit $\mathsf{ovk}' \neq \mathsf{ovk}$ such that $\mathcal{H}_0(\mathsf{ovk}') = \mathcal{H}_0(\mathsf{ovk})$. This is exactly the IND-CCA2 property that the GPV-IBE scheme has after we utilise the CHK transform. To build this adversary, we again consider a series of hybrid games $\mathcal{G}_6^i$ for $i \in [1,\delta]$. Set $\mathcal{G}_6^{(0)} = \mathcal{G}_6$ and $\mathcal{G}_6^{(i)} = \mathcal{G}_6^{(i-1)}$ with the exception that on the i^{th} ciphertext in the challenge signature, it encrypts the message $\mathsf{m}_0 = \mathsf{id}_i^0$ irrespective of the challenge bit. We argue that if $\mathcal{A}$ can distinguish between $\mathcal{G}_6^{(i)}$ and $\mathcal{G}_6^{(i-1)}$ the it breaks the IND-CCA2 property of GPV-IBE. To see this, $\mathcal{B}$ challenges $\mathcal{A}$ to game $\mathcal{G}_6^{(i)}$ and invokes its own IND-CCA2 game. During the setup, $\mathcal{B}$ sets $(\mathsf{sk}_{\mathsf{TA}}, \mathsf{pk}_{\mathsf{TA}}) := (\perp, \mathsf{pk}_{enc})$. That is, the key pair for the Tracing Authority is replaced with the challenge key pk_{enc} from the IND-CCA2 game. Any Tracing query from $\mathcal{A}$ is forwarded to the IBE decryption oracle by $\mathcal{B}$, who also returns the result to $\mathcal{A}$. Then, it waits for $\mathcal{A}$ to invoke the challenge phase of the game. Upon doing so, on the i^{th} ciphertext, $\mathcal{B}$ submits two messages to the challenge oracle of the indistinguishability game as $\mathsf{m}_{enc}^{(0)} := \mathsf{id}_i^0$ and $\mathsf{m}_{enc}^{(1)} := \mathsf{id}_i^1$, that is the delegation paths of the i^{th} attribute in Ψ. It receives back a challenge ciphertext $\mathbf{f}^*$. It simulates the proof system ZKAoK and computes σ_o to complete the VLR-HABS signature, which it gives to $\mathcal{A}$. Since the proof is simulated, and the message and policy are the same regardless of the challenge bit b, then any advantage that $\mathcal{A}$ has in winning this game allows $\mathcal{B}$ to win its IND-CCA2 game with at least equal advantage. The GPV-IBE scheme is IND-CCA2 secure under the $\mathsf{LWE}_{n_2,m_2,q_2,\chi_2}$ assumption. We conclude by noting that $\mathcal{G}_6^{(\delta)}$ is independent of the challenge bit, so any $\mathcal{A}$ has advantage negligibly close to 1/2.

$$|\Pr[\mathcal{G}_6^{(i)} = 1] - \Pr[\mathcal{G}_6^{(i-1)} = 1]| \leqslant \mathsf{Adv}_{\mathsf{LWE}_{n_2,m_2,q_2,\chi_2}} \quad \text{and} \quad \Pr[\mathcal{G}_6^{(\delta)-b} = 1] = \frac{1}{2} + \varepsilon_b$$

Since the LWE problem is assumed to be hard, the probability of the adversary winning the experiment is negligibly close to 1/2. From the sequence of games $\mathcal{G}_0$ to $\mathcal{G}_6^{(\delta)}$, we conclude that the advantage of an adversary is bounded by ε, which is negligible in the security parameter λ.

$$\begin{aligned}\mathsf{Adv}^{\mathsf{pa}}_{\mathsf{VLR\text{-}HABS},\mathcal{A}}(\lambda) &= |\Pr[\mathsf{Exp}^{\mathsf{pa\text{-}1}}_{\mathsf{VLR\text{-}HABS},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{pa\text{-}0}}_{\mathsf{VLR\text{-}HABS},\mathcal{A}}(\lambda) = 1] \\ &= \left(\frac{1}{2} + \varepsilon_1\right) - \left(\frac{1}{2} + \varepsilon_2\right) \leqslant \varepsilon\end{aligned}$$

□

D.2 Proof of Non-frameability

Lemma 4. *Our* VLR-HABS *construction satisfies Non-frameability if* $\mathcal{H}_1 : \{0,1\}^* \to \{1,2,3\}^t$ *is a random oracle,* $\mathcal{H}_2 : \mathbb{Z}_q^* \to \{0,1\}^{\tilde{m}}$ *is collision-resistant, and the* $\mathsf{SIS}_{n_1,m_1',q_1,\beta_1}$ *and* $\mathsf{SIS}_{n_4,m_4,q_4,\beta_4}$ *assumptions hold.*

Proof. We consider 4 winning conditions of the experiment:

- $\mathcal{E}_1$: The adversary forges the signature (given on line 4 of Figure 3).
- $\mathcal{E}_2$: The adversary forges a delegation (given on lines 5-8 of Figure 3)
- $\mathcal{E}_3$: The adversary produces a signature using attributes that do not satisfy Ψ (given on line 9 of Figure 3)
- $\mathcal{E}_4$: The adversary produces a fake revocation token (given on line 10 of Figure 3)

Thus we have:

$$\Pr[\mathsf{Exp}^{\mathsf{nf}}_{\mathsf{VLR\text{-}HABS},\mathcal{A}} = 1] \leqslant \Pr[\mathcal{E}_1 = 1] + \Pr[\mathcal{E}_2 = 1] + \Pr[\mathcal{E}_3 = 1] + \Pr[\mathcal{E}_4 = 1]$$

Winning Condition 1: We start with the first experiment $\mathcal{E}_1$ that we aim to show has a negligible probability of success. Following the direction for the previous constructions in this thesis we intuitively want to argue over the values $(\mathsf{upk}', \mathsf{warr}', \mathsf{m}', \Psi'), (\sigma_o', \mathbf{C}', \mathbf{f}', \pi', \mathsf{ovk}')$ that correspond to the input and output of the $\mathcal{O}_{\text{Sig}}$ oracle. We show that they are not sufficient for the adversary $\mathcal{A}$ to create valid proofs and signatures $(\sigma_o,\mathbf{C},\mathbf{f},\pi,\mathsf{ovk})$ for the values $(\mathsf{upk},\mathsf{warr},\mathsf{m},\Psi)$ different from $(\mathsf{upk}',\mathsf{warr}',\mathsf{m}',\Psi')$. More precisely, we take each element of the tuple $(\mathsf{upk}_j,\mathsf{warr}, \mathsf{m},\Psi)$ and reason about their relation with their prime counterpart from $(\mathsf{upk}',\mathsf{warr}',\mathsf{m}',\Psi')$.

The first step considers if "there has been any $\mathcal{O}_{\text{Sig}}$ request that contains upk_j", which sets the direction for the $\mathcal{A}$ of the proof. Next, argue that that the values $\mathsf{warr}',\mathsf{m}',\Psi'$ and ovk' have to coincide with $\mathsf{warr},\mathsf{m},\Psi,\mathsf{ovk}$ for $\mathcal{A}$ to actually produce valid proofs and signatures that pass the verification conditions in $\mathcal{E}_1$, which contradicts the requirement that the forgery is not an output of the signing oracle. For simplicity, we consider the probability of any adversary to guess which oracle constructs the keys for a particular user is $1/q_r$, given q_r user registration oracle calls.

$\mathcal{G}_0$: The game $\mathcal{G}_0$ is defined exactly as $\mathcal{E}_1$ except on line "$\mathcal{A}$ did not query $\mathcal{O}_{\text{Sig}}((\mathsf{upk}_j\ \mathsf{warr}),\mathsf{m},\Psi)$" that is replaced with a membership check $(\mathsf{upk}_j,\mathsf{warr},\mathsf{m},\Psi)\notin\mathsf{SigList}$ for a list $\mathsf{SigList}$ initialised empty at the beginning of the experiment, and gets updated with the inputs of the $\mathcal{O}_{\text{Sig}}$ oracle. Additionally, we introduce the list $\mathsf{SigIOList}$ that stores the input and output of the $\mathcal{O}_{\text{Sig}}$ oracle. We have that $\mathcal{E}_1$ and $\mathcal{G}_0$ have the same probability.

$$\Pr[\mathcal{G}_0=1]=\Pr[\mathcal{E}_1=1]$$

The games $\mathcal{G}_1$ and $\mathcal{G}_2$ are obtained from $\mathcal{G}_0$ by conditioning $(\mathsf{pkd}_j,\star,\star,\star)\notin\mathsf{SigList}$ and $(\mathsf{pkd}_j,\star,\star,\star)\in\mathsf{SigList}$ respectively.

$$\Pr[\mathcal{G}_0=1]=\Pr[\mathcal{G}_0=1\wedge(\mathsf{upk}_j,\star,\star,\star)\notin\mathsf{SigList}]\wedge\Pr[\mathcal{G}_0=1\wedge(\mathsf{upk}_j,\star,\star,\star)\in\mathsf{SigList}]$$

$\mathcal{G}_1$: Let $\mathcal{G}_1$ be defined as $\mathcal{G}_0$ with the restriction that $(\mathsf{upk}_j,\star,\star,\star)\notin\mathsf{SigList}$. The success probability of the adversary in $\mathcal{G}_1$ is bound by the probability it is able to forge a signature on behalf of upk_j. This is bound by the advantage of $\mathsf{SIS}_{n_1,m_1,q_1,\beta_1}$ adversary $\mathcal{B}$ against $\mathbf{A}^*f_{\mathsf{uid}}(\mathbf{z})=\mathbf{A}_{\mathsf{uid}}\mathbf{z}=\mathbf{u}$ that appears in the relation $\mathcal{R}_1$, where $(\mathbf{A}^*,\mathbf{u})$ is the SIS challenge, and $\mathbf{u}$ is the syndrome reserved for the user signature. We claim that the Argument of Knowledge property of ZKAoK ensures that $\mathcal{A}$ must have knowledge of uid.

We now construct such an adversary $\mathcal{B}$ against the SIS assumption, closely following the proof of static unforgebaility of the Bonsai signature [12]. It begins by receiving an SIS challenge of the form $(\mathbf{A}^*=[\mathbf{A}||\mathbf{U}_1^{(0)}||\mathbf{U}_1^{(1)}||...||\mathbf{U}_{ld}^{(0)}||\mathbf{U}_{ld}^{(1)}||\mathbf{u}||\mathbf{a}_1||...||\mathbf{a}_N],\mathbf{0})\in\mathbb{Z}_{q_1}^{n_1\times m_2(2ld+1)+N+1}\times\mathbb{Z}_{q_1}^{n_1}$. Since we are using the multi-syndrome variant of the Bonsai signatures, we require $N=|\mathbb{A}|$ syndromes, and one more to account for $\mathbf{u}$. it also receives the attribute syndromes $\mathbf{a}_i,i\in[1,|\mathbb{A}|]$ for the same parameters. Precisely, during the setup of the game (line 1 of Figure 2) the challenger $\mathcal{B}$ creates a list of q_r-1 top-level identities of length d as $\mathsf{id}_i\hookleftarrow\{0,1\}^d$, where q_r is the maximum number of registration queries made by the adversary $\mathcal{A}$.

First, it creates a set, $\mathcal{P}$, that contains all of the binary strings of length $p\in\{0,1\}^d$ such that p is not a prefix for any of the sampled IDs $\{\mathsf{id}_i\}_{i=1}^{q_r}$. One can think of this as a maximal tree that does not contain a precomputed identity. It has size at most q_rd and can be computed efficiently (see [12] for a description of an algorithm). $\mathcal{B}$ selects some challenge prefix $\mathsf{id}^*\in\mathcal{P}$ of length d. Next, $\mathcal{B}$ creates the matrices $\mathbf{A},\{\mathbf{A}_i^b\}_{i=1}^d$ that comprise the verification key for the VLR-HABS signature as follows:

- For each $i\in[1,d]$, let $\mathbf{A}_i^{\mathsf{id}^*[i]}=\mathbf{U}_i^{(0)}$ For $i\in[d+1,ld]$ and $b\in\{0,1\}$, let $\mathbf{A}_i^{(b)}=\mathbf{U}_i^{(b)}$.
- For each $i\in[1,d]$, compute $\mathbf{A}_i^{1-\mathsf{id}^*[i]}\leftarrow\mathsf{GenBasis}(n_1,m_1,q_1)$ with corresponding short basis $\mathbf{S}_i$.

To begin, $\mathcal{B}$ invokes $\mathcal{G}_1$ against the non-frameability adversary $\mathcal{A}$. It publishes the public parameters as described by the experiment, except for $\mathbf{A},\{\mathbf{A}_i^b\}_{i=1}^d$ which is computed as described. The challenger $\mathcal{B}$ must be able to answer oracle queries. Since we are concerned with a forgery on $\mathbf{A}_{\mathsf{id}}\mathbf{z}_0=\mathbf{u}$, let's start here. $\mathcal{A}$ may make corrupt queries of the form (i,id). If $i\neq i^*$, then $\mathcal{B}$ is able to construction $\mathbf{A}_{\mathsf{id}}=[\mathbf{A}||\mathbf{A}_1^{\mathsf{id}[1]}||..||\mathbf{A}_d^{\mathsf{id}[d]}]$ such that for some $i\in[1,d]$, $\mathsf{id}[i]\neq\mathsf{id}^*[i]$, by construction of P. For this i, $\mathcal{B}$ knows the corresponding short basis $\mathbf{S}_i$. In particular, this means $\mathcal{B}$ can run $\mathsf{ExtBasis}(\mathbf{S}_i,\mathbf{A}_{\mathsf{id}})$ to obtain $\mathbf{S}_{\mathsf{id}}$, a short basis for the matrix $\mathbf{A}_{\mathsf{id}}$. Thus it can use $\mathsf{SampleD}(\mathbf{S}_{\mathsf{id}},\mathbf{A}_{\mathsf{id}},\mathbf{u},\beta_1)$ to obtain a secret key $\mathbf{z}_0$ in response to the corrupt query. Actually, delegation is handled identically but for the syndrome $\mathbf{a}_i$. If issuing to an authority, it instead executes $\mathsf{RandBasis}$ which we note is also possible on input $\mathbf{S}_{\mathsf{id}}$. This means $\mathcal{B}$ can answer any AttIssue, signing and corruption oracles for any identity other than the challenge id^*. If $\mathcal{A}$ queries the challenge then the game aborts, but in this case it had already lost the experiment and thus the winning probability is preserved.

It is now possible to construct a solution for the SIS challenge as follows. If $\mathcal{A}$ produced a forgery $\mathbf{z}_0^*$ for $\mathbf{A}_{\mathsf{id}^*}$ w.r.t $\mathbf{u}^*$, then $\mathbf{A}_{\mathsf{id}^*}$ is the concatenation of ld blocks $\mathbf{U}_i^b$ and one of $\mathbf{a}_i$ or $\mathbf{a}$. Therefore, by inserting zero vectors in $\mathbf{z}_0^*$, $\mathcal{B}$ can generate a non-zero $\mathbf{z}$ such that $\mathbf{A}\mathbf{z}=\mathbf{u} \mod q_1$, which solves the SIS challenge. What remains is to extract $\mathbf{z}_0^*$ from the VLR-HABS signature.

To extract the SIS challenge from the forgery, we make use of a Forking Lemma of [48] to receive 3 accepting transcripts $(\mathsf{RSP}_1,\mathsf{RSP}_2,\mathsf{RSP}_3)$ of ZKAoK. It rewinds the adversary and plays the same random tape to build an extractor as detailed in Section 5.2 in the proof of Theorem 1 to extract a witness to the statement. One can argue that $\mathcal{A}$ must have queried $\mathcal{H}_1$ on input $(\mathsf{m},\Psi,\mathbf{f},\mathbf{C},\mathsf{pp},\{\mathsf{CMT}_i\}_{i=1}^t)$ otherwise, the probability that $(Ch_1,...,Ch_t)=\mathcal{H}_1(\mathsf{m},\Psi,\mathbf{f},\mathbf{C},\mathsf{pp},\{\mathsf{CMT}_i\}_{i=1}^t)$ is at most 3^{-t} . Therefore, with probability at least $\varepsilon'-3^{-t}$, there exists certain $\kappa^*\leqslant q_o$ such that the κ^* oracle query uses $(\mathsf{m},\Psi,\mathbf{f},\mathbf{C},\mathsf{pp},\{\mathsf{CMT}_i\}_{i=1}^t)$. $\mathcal{B}$ picks κ^* as the forking point. It replays $\mathcal{A}$ with the same random tape and input as in the original run. In each rerun, $\mathcal{A}$ is given the same $r_1,...,r_{\kappa^*-1}$ but from r_{κ^*} is is given random values $r_{\kappa^*},...,r_{q_o}\hookleftarrow\{1,2,3\}^t$. Precisely, the Improved Forking Lemma by Pointcheval and Vaudenay says that, with probability larger than $\frac{1}{2}$, the adversary $\mathcal{B}$ can obtain a 3-fork involving the tuple after less than $32\cdot q_o/(\varepsilon'-3^{-t})$ executions of $\mathcal{A}$. Now, let the answers of $\mathcal{B}$ with respect to the 3-fork branches be

$$r_{\kappa^*}^{(1)}=(Ch_1^{(1)},...,Ch_t^{(1)}),\quad r_{\kappa^*}^{(2)}=(Ch_1^{(2)},...,Ch_t^{(2)}),\quad r_{\kappa^*}^{(3)}=(Ch_1^{(3)},...,Ch_t^{(3)})$$

Then, the probability we obtain a valid challenge set, that is

$$\Pr[\exists j\in\{1,...,q_o\}:\{Ch_j^{(1)},Ch_j^{(2)},Ch_j^{(3)}\}=\{1,2,3\}]=1-(7/9)^{-t}.$$

Conditioned on the existence of such an index j, it parses the 3 forgeries corresponding to the fork $(\mathsf{RSP}_1^{(j)},\mathsf{RSP}_2^{(j)},\mathsf{RSP}_3^{(j)})$ branches to obtain. They turn out to be 3 valid responses with respect to 3 different challenges for the same commitment $CMT^{(j)}$. Since COM is assumed to be computationally-binding, we can use the knowledge extractor of the underlying argument system to extract a witness for the relation $\mathcal{R}_5$, and thus extract the SIS solution created by $\mathcal{A}$, using the strategy detailed in [42].

$$\Pr[\mathcal{G}_1=1]=\Pr[\mathcal{G}_0=1\wedge(\mathsf{pk}_j,\star,\star,\star)\notin\mathsf{SigList}]\leqslant\frac{\mathsf{Adv}_{\mathsf{SIS}_{\mathsf{n}_1,\mathsf{m}_1,q_1,\beta_1}}}{2q_r(1-(7/9)^{-t})}$$

$\mathcal{G}_2$: This game uses the exact steps performed by game $\mathcal{G}_1$, but in the setting where $\mathcal{A}$ requested at least one signature that contains user upk_j. There exists an adversary query $((\mathsf{upk}_j,\mathsf{warr}',\mathsf{m}',\Psi'),(\sigma_o',\mathbf{C}',\pi',\mathsf{ovk}'))\in\mathsf{SigIOList}$ with $(\mathsf{warr},\mathsf{m},\Psi)\neq(\mathsf{warr}',\mathsf{m}',\Psi')$. Hence,

$$\begin{aligned}\Pr[\mathcal{G}_2=1]&=\Pr[\mathcal{G}_0=1\wedge(\mathsf{upk}_j,\star,\star,\star)\in\mathsf{SigList}]\\&=\Pr[\mathcal{G}_0=1\wedge(\mathsf{upk}_j,\mathsf{warr}',\mathsf{m}',\Psi'),(\sigma_o',\mathbf{C}',\mathbf{f}',\pi',\mathsf{ovk}'))\in\mathsf{SigIOList}].\end{aligned}$$

Using the method applied on $\mathcal{G}_1$, we reason on the relation between the OTS public keys ovk and ovk'. We split game $\mathcal{G}_2$ based on $\mathsf{ovk}=\mathsf{ovk}'$ and $\mathsf{ovk}\neq\mathsf{ovk}'$.

$$\Pr[\mathcal{G}_2=1]=\Pr[\mathcal{G}_2=1\wedge\mathsf{ovk}=\mathsf{ovk}']+\Pr[\mathcal{G}_2=1\wedge\mathsf{ovk}\neq\mathsf{ovk}'].$$

$\mathcal{G}_3$: We define $\mathcal{G}_3$ as the game $\mathcal{G}_2$ where $\mathsf{ovk}\neq\mathsf{ovk}'$. In this case, the adversary $\mathcal{A}$ is once again able to provide a forgery for upk_j without knowledge of usk_j. The reduction argument is almost identical to the method of computing the bound for $\mathcal{G}_1$, except that now $\mathcal{A}$ asks signature queries for uid. By programming the random oracle $\mathcal{H}_1$, the game is able to simulate the proof ZKAoK without knowledge of the witness for uid, namely $(\mathsf{upk}_j,\mathsf{usk}_j)$, that passes verification as described in the proof of Theorem 3. Since $\mathsf{ovk}\neq\mathsf{ovk}'$, the call to the random oracle will result in a different challenge and thus $\mathcal{A}$ cannot reply π and must generate a new proof. Since any query to the signing oracle on behalf of uid included a simulated proof, it cannot extract a valid witness and thus must produce a fresh forgery for the Boyen signature on behalf of uid. We bind the capabilities of the adversary in this game by constructing an adversary $\mathcal{B}$ against an $\mathsf{SIS}_{n_1,m_1,q_1,\beta_1}$ instance. Again there is a polynomial loss in the reduction as the game is required to guess for which user $\mathcal{A}$ will attempt to create a forgery, and further loss as the extractor (by using the Improved Forking Lemma) only succeeds with probability $1/2(1-(7/9)^{-t})$. Thus we have:

$$\Pr[\mathcal{G}_4=1]=\Pr[\mathcal{G}_3=1\wedge \mathsf{ovk}\neq\mathsf{ovk}']\leqslant\frac{\mathsf{Adv}_{\mathsf{SIS}_{n_1,m_1,q_1,\beta_1}}}{2q_r(1-(7/9)^{-t})}$$

$\mathcal{G}_4$: The game $\mathcal{G}_4$ uses the steps of $\mathcal{G}_2$ with the additional restriction $\mathsf{ovk}'=\mathsf{ovk}$. With the $\mathsf{upk}_j=\mathsf{upk}'$ restriction from $\mathcal{G}_2$ we further transform this game in the view of $\mathcal{G}_1$.

$$\begin{aligned}\Pr[\mathcal{G}_4=1] &=\Pr[\mathcal{G}_2=1\wedge \mathsf{ovk}'=\mathsf{ovk}]\\ &=\Pr[\mathcal{G}_1=1\wedge(\mathsf{upk}_j,\mathsf{warr}',\mathsf{m}',\Psi'),(\sigma_o',\mathbf{C}',\mathbf{f}',\pi',\mathsf{ovk}))\in\mathsf{SigIOList}].\end{aligned}$$

Currently, we have deduced that $\mathcal{A}$ has made a $\mathcal{O}_{\mathrm{Sig}}$ query for $(\mathsf{upk}_j,\mathsf{warr}',\mathsf{m}',\Psi')$ different from $(\mathsf{upk}_j,\mathsf{warr},\mathsf{m},\Psi)$, but with the same OTS signature public key ovk. We split the probability in $\mathcal{G}_5$ based on the equality test between (m,Ψ) and (m',Ψ').

$$\Pr[\mathcal{G}_4=1]=\Pr[\mathcal{G}_4=1\wedge(\mathsf{m},\Psi)=(\mathsf{m}',\Psi')]+\Pr[\mathcal{G}_4=1\wedge(\mathsf{m},\Psi)\neq(\mathsf{m}',\Psi')].$$

$\mathcal{G}_5$: We define game $\mathcal{G}_5$ as the game $\mathcal{G}_4$ where $(\mathsf{m},\Psi)\neq(\mathsf{m}',\Psi')$. 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 osk. In this argument we will assume that $\mathcal{H}_2$ offers collision resistance.

The capabilities of adversary $\mathcal{A}$ in this case, are bounded by the advantage of the unforgeability adversary $\mathcal{B}_{\mathsf{ots}}$ for the OTS signature scheme that uses osk as the secret key. There is a slight loss of accuracy as $\mathcal{B}_{\mathsf{ots}}$ needs to identify which is the $\mathcal{O}_{\mathrm{Sig}}$ query that uses (m',Ψ') among all $\mathcal{O}_{\mathrm{Sig}}$ queries for upk_j. As always, it is able to do this with probability $1/q_s$ if $\mathcal{A}$ makes q_s sign queries for the same upk_j value. The factor $1/q_r$ is given by the guess $\mathcal{B}_{\mathsf{ots}}$ makes on which is the user registration oracle with upk. To build the adversary $\mathcal{B}_{\mathsf{ots}}$ against the the EUF-sCMA property of Bonsai signatures, it follows the exact strategy used to show static unforgeability (EUF-sCMA) in [12], however, instead it does not receive a list of messages before generating the challenge verification key for $\mathcal{A}$. Instead, $\mathcal{B}$ selects a random message policy pair (m^*,Ψ^*) (we stress these can be entirely random and do not have to be meaningful choices) and computes the chameleon hash $h\leftarrow\mathcal{H}_2(\mathsf{m}^*,\Psi^*,\mathbf{f}^*,\pi^*,\mathbf{C}^*;r^*)$. Using the notation of $\mathcal{G}_1$, the set $\mathcal{P}$ for which $\mathcal{B}_{\mathsf{ots}}$ can simulate signatures will contain precisely h. When $\mathcal{A}$ finally makes the oracle query that $\mathcal{B}_{\mathsf{ots}}$ guesses $\mathcal{A}$ will use to compute the forgery, it receives the pair $(\tilde{\mathsf{m}},\tilde{\Psi})$ and using the trapdoor for the chameleon hash function it can computes $\tilde{r}$ such that $\mathcal{H}_2(\tilde{\mathsf{m}},\tilde{\Psi},\mathbf{f},\pi,\mathbf{C};\tilde{r})=h=\mathcal{H}_2(\mathsf{m}^*,\Psi^*,\mathbf{f}^*,\pi^*,\mathbf{C}^*;r^*)$. We briefly note that $\mathbf{f},\pi,\mathbf{C}$ are honestly computed according to the experiment. By design, $\mathcal{B}_{\mathsf{ots}}$ can compute a simulated Bonsai signature on message h and thus it is able to answer $\mathcal{A}$'s signing query. It then waits for $\mathcal{A}$ to output a forgery. If the forged signature verifies, then $\mathcal{A}$ was able to create a forged Bonsai signature, which means it was able to break the the $\mathsf{SIS}_{n_4,m_4,q_4,\beta_4}$ assumption according to the proof of static unforgeability in [12]. To win its game, $\mathcal{B}$ uses the Improved Forking Lemma and the extractor from the proof of Theorem 1 in the way described in $\mathcal{G}_1$ to extract the $\mathsf{SIS}_{n_4,m_4,q_4,\beta_4}$ solution. Thus we bind the success probabilities as:

$$\Pr[\mathcal{G}_5=1]=\Pr[\mathcal{G}_4=1\wedge(\mathsf{m},\Psi)\neq(\mathsf{m}',\Psi')]\leqslant\frac{1}{2q_sq_r(1-(7/9)^{-t})}\times\mathsf{Adv}_{\mathsf{SIS}_{n_4,m_4,q_4,\beta_4}}$$

$\mathcal{G}_6$: We define game $\mathcal{G}_6$ as the game $\mathcal{G}_4$ where $(\mathsf{m},\Psi)=(\mathsf{m}',\Psi')$. Because of the $(\mathsf{warr},\mathsf{m},\Psi)\neq(\mathsf{warr}',\mathsf{m}',\Psi')$ restriction, this leads to $\mathsf{warr}'\neq\mathsf{warr}$. If we include the condition added by game G_5 with respect to game $\mathcal{G}_1$, we have

$$\begin{aligned}\Pr[\mathcal{G}_6=1] &= \Pr[\mathcal{G}_4=1\wedge(\mathsf{m},\Psi)=(\mathsf{m}',\Psi')]\\ &= \Pr[\mathcal{G}_1=1\wedge(\mathsf{upk}_j,\mathsf{warr}',\mathsf{m},\Psi),(\sigma_o',\mathbf{C}',\mathbf{f}',\pi',\mathsf{ovk}))\in\mathsf{SigIOList}]\end{aligned}$$

We proceed by splitting the probability of $\mathcal{G}_6$ based on the equality of $\mathbf{f}$ and $\mathbf{f}'$.

$$\Pr[\mathcal{G}_6=1]=\Pr[\mathcal{G}_6=1\wedge\mathbf{f}=\mathbf{f}']+\Pr[\mathcal{G}_6=1\wedge\mathbf{f}\neq\mathbf{f}']$$

$\mathcal{G}_7$: Let $\mathcal{G}_7$ be the game defined by $\mathcal{G}_6$ with $\mathbf{f}\neq\mathbf{f}'$. In such a case, the adversary $\mathcal{A}$ is able to create a forgery without knowledge of osk, that passed the verification in the body of experiment $\mathcal{G}_6$.

The probability of success for adversary $\mathcal{A}$ in this game, is once again bounded by the advantage of the OTS forger $\mathcal{B}'_{\mathsf{ots}}$ which behaves exactly as $\mathcal{B}_{\mathsf{ots}}$ from game $\mathcal{G}_5$. The difference in this case is given by the output of the adversary. Here, $\mathcal{A}$ provides an OTS signature that satisfies $\mathbf{f}\neq\mathbf{f}'$, while in $\mathcal{G}_5$ we made the requirement that the forged one-time signature is for $(\mathsf{m},\Psi)\neq(\mathsf{m}',\Psi')$. We set up the reduction under the assumption that $\mathcal{H}_2$ is collision-resistant. As before, $\mathcal{B}_{\mathsf{ots}}$ must guess which uid and which signature query $\mathcal{A}$ will atempt to forge and follows a similar strategy to $\mathcal{G}_6$. During the setup phase of the experiment, in addition to the rest of the setup, it computes a chameleon hash value $h\leftarrow\mathcal{H}_2(\mathsf{m}^*,\Psi^*,\mathbf{f}^*,\pi^*,\mathbf{C}^*;r^*)$ which comprises the set $\mathcal{P}$ of all "messages" for which $\mathcal{B}_{\mathsf{ots}}$ can simulate. Since it has access to the trapdoor for the chameleon hash function, it can build $\mathbf{f},\pi,\mathbf{C}$ based on the adversarial inputs $\tilde{\mathsf{m}},\tilde{\Psi}$ and computes a value $\tilde{r}$ such that $h=\mathcal{H}_2(\tilde{\mathsf{m}},\tilde{\Psi},\mathbf{f},\pi,\mathbf{C};\tilde{r})$, thus it can produce a valid VLR-HABS signature for the target uid. It waits until $\mathcal{A}$ submits its forgery. In particular, it was able to produce a forgery for the EUF-CMA property of the Bonsai signature, which is implied by the $\mathsf{SIS}_{n_4,m_4,q_4,\beta_4}$ assumption. As before, $\mathcal{B}_{\mathsf{ots}}$ uses the Improved Forking Lemma and the extractor from Theorem 1 to extract the SIS solution with probability $1/2(1-(7/9)^{-t})$. Hence,

$$\Pr[\mathcal{G}_8=1]=\Pr[\mathcal{G}_6=1\wedge\mathbf{f}\neq\mathbf{f}']\leqslant\frac{1}{q_sq_r(1-(7/9)^{-t})}\mathsf{Adv}_{\mathsf{SIS}_{\mathsf{n}_4,\mathsf{m}_4,\mathsf{q}_4,\beta_4}}$$

$\mathcal{G}_8$: The game $\mathcal{G}_8$ is defined as $\mathcal{G}_6$ where $\mathbf{f}=\mathbf{f}'$. Given $\mathsf{warr}\neq\mathsf{warr}'$, we now show that $\mathbf{f}\neq\mathbf{f}'$. According to the correctness of GPV-IBE, $\mathbf{f}'$ must decrypt with overwhelming probability to one of the two message. In such a case, the adversary $\mathcal{A}$ has managed to produce a ciphertext $\mathbf{f}$ that decrypts to two different messages $m_0=(\mathsf{upk}_j,\mathsf{warr},\mathsf{ovk})$ and $m_1=(\mathsf{upk}_j,\mathsf{warr}',\mathsf{ovk})$. We build $\mathcal{B}_{\mathsf{ibe}}$ that performs the steps in $\mathcal{G}_6$ and waits for $\mathcal{A}$ to provide an output $(((\sigma_o,\mathbf{C},\mathbf{f},\pi,\mathsf{ovk}),\mathsf{m},\Psi),(\mathsf{upk}_j,\mathsf{warr},(\hat{\pi})))$. Then, it uses that output to construct message m_0, and looks through the list $\mathsf{SigIOList}$ for the query the adversary $\mathcal{A}$ has made that produced the same ciphertext $\mathbf{f}$ and builds m_1. $\mathcal{B}_{\mathsf{ibe}}$ outputs the message that does not appears when it does a decryption using the TA's key $\mathsf{sk}_{\mathsf{TA}}$. For simplicity, this adversary also provides the randomness needed to produce the same ciphertext in the body of the correctness experiment. For our choice of GPV-IBE, this happens with all but negligible probability. We have,

$$\Pr[\mathcal{G}_8=1]=\Pr[\mathcal{G}_6=1\wedge\mathbf{f}=\mathbf{f}']\leqslant\tilde{\varepsilon}(\lambda)$$

$\mathcal{G}_9$: Defined as $\mathcal{G}_8$ but with the condition that $\mathbf{C}=\mathbf{C}'$. The parameter m_3 is chosen large enough ($\geqslant n_3\log q_3$) to ensure secret vectors in the $\mathsf{LWE}_{n_3,m_3,q_3,\chi}$ problem are uniquely defined (see [47]). Hence, the adversary $\mathcal{A}$ cannot find $\mathsf{warr}\neq\mathsf{warr}'$, or more precisely, $\mathsf{id}\neq\mathsf{id}'$ s.t. $\mathbf{C}=\mathbf{C}'$. Thus,

$$|\Pr[\mathcal{G}_9=1]|=|\Pr[\mathcal{G}_8=1]|$$

Finally, if $\mathbf{C}\neq\mathbf{C}'$ then once more the adversary was able to forge an OTS signature, this time over the $\mathbf{C}$ component of the OTS message. Again this argument follows with the implicit assumption that $\mathcal{H}_2$ enjoys collision-resistance. The extraction of the SIS challenge follows the same process as before; $\mathcal{B}$ must guess which uid and which signature query $\mathcal{A}$ will attempt to forge and follows a similar strategy to $\mathcal{G}_5$ and $\mathcal{G}_7$. During the setup phase of the experiment, in addition to the rest of the setup, it computes a chameleon hash value $h \leftarrow \mathcal{H}_2(\mathsf{m}^*,\Psi^*,\mathbf{f}^*,\pi^*,\mathbf{C}^*;r^*)$ which comprises the set $\mathcal{P}$ of all "messages" for which $\mathcal{B}$ can simulate. Since $\mathcal{B}$ has access to the trapdoor for the chameleon hash function, it can build $\mathbf{f},\pi,\mathbf{C}$ based on the adversarial inputs $\tilde{\mathsf{m}},\tilde{\Psi}$ and computes a value $\tilde{r}$ such that $h=\mathcal{H}_2(\tilde{\mathsf{m}},\tilde{\Psi},\mathbf{f},\pi,\mathbf{C};\tilde{r})$, thus it can produce a valid VLR-HABS signature for the target uid. It waits until $\mathcal{A}$ submits its forgery. In particular, it was able to produce a forgery for the EUF-sCMA property of the Boyen signature, which is implied by the $\mathsf{SIS}_{n_4,m_4,q_4,\beta_4}$ assumption. As before, $\mathcal{B}$ uses the Improved Forking Lemma and the extractor from Theorem 1 to extract the SIS solution with probability $(1-(7/9)^{-t})$. The probability of success is bounded an adversary against the $\mathsf{SIS}_{n_4,m_4,q_4,\beta_4}$ assumption. Hence

$$\Pr[\mathcal{G}_9=1] \leqslant \mathsf{Adv}_{\mathsf{SIS}_{\mathsf{n}_4,\mathsf{m}_4,\mathsf{q}_4,\beta_4}}$$

From the sequence of games starting $\mathcal{G}_0,...,\mathcal{G}_9$, it follows that the probability of $\mathcal{E}_1$ is bounded by the SIS hardness assumption, and the soundness property of ZKAoK which holds with probability 3^{-t}. Thus the advantage of the adversary, ε_1, is negligible in the security parameter.

Winning Condition 2: The experiment $\mathcal{E}_2$ deals with the case where the adversary $\mathcal{A}$ is able to provide a forged delegation for an honest authority pkd_i and some attribute $\mathsf{att}=\mathbf{a}$. Recall that the experiment prevents the adversary from winning if any authority in the delegation path has been corrupted, which trivially prevents $\mathcal{A}$ extending the corrupt authority $\mathbf{A}_{\mathsf{id}_1||...||\mathsf{id}_l}$ to $\mathbf{A}_{\mathsf{id}_1||...||\mathsf{id}_l||\mathsf{id}_{l+1}}$, for an uncorrupted authority id_{l+1}. The challenger $\mathcal{B}$ embeds the SIS challenge a delegation path, $\mathbf{A}_{\mathsf{id}^*}$, this time of length ld. We argue that if the adversary is able to forge a delegation, that it was able to solve an SIS instance. Again, we must carefully prepare the experiment $\mathcal{E}_2$ so that $\mathcal{B}$ is able to extract the solution. It is identical to the preparation for $\mathcal{E}_1:\mathcal{G}_1$ with the exception that the forgery is created with respect to any syndrome $\mathbf{a}_i$. The argument presented previously is actually general enough to deploy directly, however, the set P now contains all *delegation paths* that $\mathcal{B}$ guesses $\mathcal{A}$ will query. The set P is still computable since the size of the space is only polynomially larger. We note that $\mathcal{B}$ cannot answer corruption queries for which $\mathsf{id}=\mathsf{id}^*[1:kd]$ i.e. a prefix to the challenge. However, this corresponds to $\mathcal{A}$ corrupting an authority above the guessed forgery target, in which case the game aborts. Thus there is no loss in winning probability. In the event $\mathcal{A}$ queries AttIssue for such an id, then $\mathcal{B}$ outputs the corresponding id, warr and att but does not compute $\mathsf{skd}_{i,\mathsf{att}}$. This is undetectable by $\mathcal{A}$ unless it once again tries to corrupt an authority above id^*. It can, however, delegate further once $\exists i$ s.t. $\mathsf{id}[i]\neq\mathsf{id}^*[i]$, which is required by the game since the adversary can ask for delegation and corruption queries for any other delegation path. We stress the only time it cannot answer oracle queries coincide with the loss conditions of the experiment. With these adaptations, the argument is similar to that presented in $\mathcal{E}_1:\mathcal{G}_1$, and we claim the result.

$$\Pr[\mathcal{E}_2=1] \leqslant \frac{\mathsf{Adv}_{\mathsf{SIS}_{\mathsf{n}_1,\mathsf{m}_1,\mathsf{q}_1,\beta_1}}}{2q_r(1-(7/9)^{-t})}$$

Winning Condition 3: Next we will consider a PPT adversary $\mathcal{A}$ against the winning condition $\mathcal{E}_3$. That is, it is able to produce a signature that verifiers with respect to a policy for which the attributes contained in the warrant do not satisfy.

We directly argue that the advantage of an adversary against $\mathcal{E}_3$ is bounded by an adversary $\mathcal{B}_{\mathsf{zkaok}}$ against the soundness property of ZKAoK. To build $\mathcal{B}_{\mathsf{zkaok}}$, it invokes the soundness experiment and executes $\mathcal{E}_3$ against $\mathcal{A}$. It can answer all oracle queries and simply waits for $\mathcal{A}$ to output σ that wins against $\mathcal{E}_3$. It sets the witness as $w=(\mathsf{uid},\mathbf{z}_0,\{\mathbf{z}_i,\mathbf{a}_i,\mathbf{e}_i\}_{i=1}^{\delta},\{\mathsf{id}\}_{\mathsf{id}\in\mathsf{warr}},\{\mathbf{B}_{\mathsf{id}},\mathbf{E}_{\mathsf{id}}\}_{\mathsf{id}\in\mathsf{warr}})$ from the output of $\mathcal{A}$, the statement as $x=(\mathbf{A},\mathbf{R},\{\mathbf{A}_i,\mathbf{R}_i\}_{i=1}^{d},\mathbf{G}^*,\mathbf{P}^*,\mathbf{Q}^*,\mathbf{C},\mathbf{f},\mathbf{p},\mathbf{u})$ for the relation $\mathcal{R}_1$. To win $\mathcal{E}_3$ then $\Psi(A)\neq 1$, but $\mathsf{ZKAoK.Verify}(x,\pi)=1 \implies \langle\mathbf{a},\mathbf{p}\rangle=1$ then $\mathcal{A}$ was able to create a false proof for $\mathcal{R}_1(w,x)$. Then, $\mathcal{B}_{\mathsf{zkaok}}$ forwards $(w,x,\mathcal{R}_1)$ as a response to its soundness game against ZKAoK and wins if $\mathcal{A}$ also won against $\mathcal{E}_3$. We note that soundness only fails with probability 3^{-t} for some soundness parameter t. Thus, we have:

$$\Pr[\mathcal{E}_3=1]\leqslant 3^{-t}$$

Winning Condition 4: Next we will consider the case the adversary wins by creating a signature that passes verification yet contains a revoked identity.
$\mathcal{G}_0$: Defined as $\mathcal{E}_4$ but with the addition of an additional check "$\forall \mathsf{id}\in \mathsf{warr}, \exists \mathbf{C}_i \in \{\mathbf{C}\}_{i=1}^{l\delta}$ s.t. $\mathbf{C}_i = \mathbf{B}_i\mathbf{R}_{\mathsf{id}}+\mathbf{E}_i$", where warr is obtained by decrypting $\mathbf{C}$ in the VLR-HABS signature using the TA's key. The difference in success probability of the adversary between $\mathcal{E}_4$ and $\mathcal{G}_1$ is bound by the advantage of an adversary against the soundness property of ZKAoK. To build this adversary, $\mathcal{B}$ sets up the game according to Figure 2. It can answer all queries made by $\mathcal{A}$ and waits for it to output a forgery. From the VLR-HABS signature, it forwards on the proof π (for the public statement $x = (\mathbf{A},\mathbf{R},\{\mathbf{A}_i,\mathbf{R}_i\}_{i=1}^{d},\mathbf{G}^*,\mathbf{P}^*,\mathbf{Q}^*,\mathbf{C},\mathbf{f},\mathbf{p},\mathbf{u})$) to its soundness game. It wins if $\mathcal{A}$ was able to produce a forgery against $\mathcal{E}_4$. We note that soundness only fails with probability 3^{-t} for some soundness parameter t.

$$|\Pr[\mathcal{E}_4=1]-\Pr[\mathcal{G}_0=1]|\leqslant 3^{-t}.$$

Now, since for each $\mathsf{id}\in\mathsf{warr}, \exists \mathbf{C}_{\mathsf{id}} = \mathbf{B}_i\mathbf{R}_{\mathsf{id}}+\mathbf{E}_i \in \{C_i\}_{i=1}^{l\delta}$, then if we also have $\mathsf{id}\in\mathsf{RevokeList}$, then Lemma 2 implies that $\mathbf{C}_{\mathsf{id}}\mathbf{y}_{\mathsf{id}}\leqslant n_3\beta_3^2$ with probability 1, that is, a signature is falsely accepted with probability 0. Thus we have:

$$|\Pr[\mathcal{G}_0=1]|=0\leqslant\varepsilon_3$$

It follows that the success probability of an adversary against $\mathcal{E}_4$ is bound by 3^{-t}, where $t=\omega(\log n)$ and thus is negligible.

Recalling that each winning condition for the non-frameability experiment is bound by a negligibly small function in the security parameter, we conclude that our scheme is non-frameable.

$$\Pr[\mathsf{Exp}^{\mathrm{nf}}_{\mathsf{VLR\text{-}HABS},\mathcal{A}}=1]\leqslant\Pr[\mathcal{E}_1=1]+\Pr[\mathcal{E}_2=1]+\Pr[\mathcal{E}_3=1]+\Pr[\mathcal{E}_4=1]\leqslant\varepsilon(\lambda)$$

□

D.3 Proof of Path Traceability

Lemma 5. *Our* VLR-HABS *construction satisfies Path Traceability if* $\mathcal{H}_1:\{0,1\}^*\to\{1,2,3\}^t$ *is a random oracle and the* $\mathsf{SIS}_{n_1,m_1,q_1,\beta_1}$ *assumption holds.*

Proof. We divide the advantage of the path traceability adversary $\mathcal{A}$ for the experiment $\mathsf{Exp}^{\mathrm{tr}}_{\mathsf{VLR\text{-}HABS},\mathcal{A}}$ in Figure 4 by the two winning conditions for the adversary:

1. $\mathcal{E}_1$: Trace fails for a valid VLR-HABS signature
2. $\mathcal{E}_2$: There exists a signature in the warrant, for some attribute introduced for a 'rogue' entity, that is not registered, by an honest authority.

We have,

$$\Pr[\mathsf{Exp}^{\mathrm{tr}}_{\mathsf{VLR\text{-}HABS},\mathcal{A}}(\lambda)]\leqslant\Pr[\mathcal{E}_1=1]+\Pr[\mathcal{E}_2=1].$$

Winning Condition 1: We start with experiment $\mathcal{E}_1$, where we use the soundness of ZKAoK to

show that decryption of an GPV-IBE ciphertext cannot fail.
$\mathcal{G}_0$: The game $\mathcal{G}_0$ is defined exactly as From this, it immediately follows that

$$\Pr[\mathcal{G}_0=1]=\Pr[\mathcal{E}_1=1].$$

$\mathcal{G}_1$: We define game $\mathcal{G}_1$ as $\mathcal{G}_0$, except that we add the line

$$\text{“}\forall i,\exists\mathbf{s}\text{ s.t. }\mathbf{f}_i^{(1)}=\mathbf{P}\mathbf{s}+\mathbf{e}_1,\mathbf{f}_i^{(2)}=\mathbf{N}^T\mathbf{s}+\mathbf{e}_2+\lfloor q/2\rfloor\mathsf{id}_i\text{ where }\mathbf{f}_i=[\mathbf{f}_i^{(1)}||\mathbf{f}_i^{(2)}]\wedge\text{”}$$

between lines 3 and 4. This new check performed by $\mathcal{G}_1$ is one of the conditions encoded in the relation for ZKAoK, and $\mathcal{A}$ is able to notice the difference between $\mathcal{G}_0$ and $\mathcal{G}_1$ if it can produce a valid ZKAoK proof for a false statement (that does not have a witness $\mathbf{s}$). We bind the probability of $\mathcal{A}$ to distinguish this games, by the advantage of the soundness adversary $\mathcal{B}_{\mathsf{zkaok}}$ for ZKAoK. To build the adversary, $\mathcal{B}$ follows the experiment as defined in $\mathcal{G}_1$ and we note it is able to answer all oracle queries. It waits for $\mathcal{A}$ to output a VLR-HABS signature σ. If σ wins $\mathcal{G}_0$ but fails $\mathcal{G}_1$ then it was able to produce a proof over a false witness and therefore the proof π contained in σ breaks soundness of ZKAoK. $\mathcal{B}$ extracts π from σ, builds the statement $(\mathbf{A},\mathbf{R},\{\mathbf{A}_i,\mathbf{R}_i\}_{i=1}^{d},\mathbf{G}^*,\mathbf{P}^*,\mathbf{Q}^*,\mathbf{C},\mathbf{f},\mathbf{p},\mathbf{u})$ for the relation $\mathcal{R}_5$, and submits this as a false proof. We have seen that soundness for ZKAoK fails with probability 3^{-t}, see Theorem 1. Hence,

$$|\Pr[\mathcal{G}_0=1]-\Pr[\mathcal{G}_1=1]|\leqslant 3^{-t}$$

We now argue that an adversary against $\mathcal{G}_1$ is bound by the correctness property of GPV-IBE. The ability to output a ciphertext that decrypts to a different warr is bound by the probability of an decryption to fail after an encryption, even on adversarial valid inputs. We construct adversary $\mathcal{B}_{\mathsf{ibe}}$ for the correctness property of IBE that runs the experiment as described in $\mathcal{G}_1$ and invokes $\mathcal{A}$. It waits for $\mathcal{A}$ to submit a VLR-HABS signature σ that wins against $\mathcal{G}_1$. In this case, $\mathcal{B}$ extracts the ciphertext component $\mathbf{C}$ of σ and forwards this as a response to its correctness game. The GPV-IBE scheme is correct with overwhelming probability [22], thus, for $\tilde{\varepsilon}\in\mathsf{poly}(\lambda)$ we have:

$$|\Pr[\mathcal{G}_1=1]|\leqslant\tilde{\varepsilon}(\lambda)$$

Winning Condition 2: The experiment $\mathcal{E}_2$ deals with the case where the adversary $\mathcal{A}$ is able to provide a forged delegation for an honest authority pkd_i and some attribute $\mathsf{att}=\mathbf{a}$, to an unknown registered identity. Recall that the experiment prevents the adversary from winning if any authority in the path has been corrupted, which trivially prevents $\mathcal{A}$ extending the corrupt authority $\mathbf{A}_{\mathsf{id}_1||\ldots||\mathsf{id}_l}$ to $\mathbf{A}_{\mathsf{id}_1||\ldots||\mathsf{id}_l||\mathsf{id}_{l+1}}$, for an uncorrupted authority id_{l+1}. If the adversary is able to forge a delegation, that it was able to compute a solution to an SIS challenge. Again, we must carefully prepare the experiment $\mathcal{E}_2$ so that $\mathcal{B}$ is able to extract an SIS challnge. It is identical to the preparation for non-frameability $\mathcal{E}_1:\mathcal{G}_1$ with the adaptations detailed in non-frameability $\mathcal{E}_2$. Thus we claim the result:

$$\Pr[\mathcal{E}_2=1]\leqslant\frac{1}{2q_r(1-(7/9)^{-t})}\times\mathsf{Adv}_{\mathsf{SIS}_{n_1,m_1,q_1,\beta_1}}.$$

Recalling that both winning conditions are bound by functions that are negligible in the security parameter, we conclude that our scheme satisfies path traceability.

$$\Pr[\mathsf{Exp}^{\mathrm{tr}}_{\mathsf{VLR\text{-}HABS},\mathcal{A}}=1]\leqslant\Pr[\mathcal{E}_1=1]+\Pr[\mathcal{E}_2=1]\leqslant\varepsilon(\lambda)$$

□