Decentralized Policy-Hiding Attribute-Based Encryption with Receiver Privacy

Yan Michalevsky[1] and Marc Joye[2]

[1] Anjuna Security and Stanford University (USA)
[2] NXP Semiconductors (USA)

Abstract. Attribute-based encryption (ABE) enables limiting access to encrypted data to users with certain attributes. Different aspects of ABE were studied, such as the multi-authority setting (MA-ABE), and policy hiding, meaning the access policy is unknown to unauthorized parties. However, no practical scheme so far provably provides both properties, which are often desirable in real-world applications: supporting decentralization, while hiding the access policy. We present the first practical decentralized ABE scheme with a proof of being policy-hiding. Our construction is based on a decentralized inner-product predicate encryption scheme, introduced in this paper, which hides the encryption policy. It results in an ABE scheme supporting conjunctions, disjunctions and threshold policies, that protects the access policy from parties that are not authorized to decrypt the content. Further, we address the issue of receiver privacy. By using our scheme in combination with vector commitments, we hide the overall set of attributes possessed by the receiver from individual authorities, only revealing the attribute that the authority is controlling. Finally, we propose randomizing-polynomial encodings that immunize the scheme in the presence of corrupt authorities.

1 Introduction

Attribute-based encryption (ABE), first proposed by Sahai and Waters [27], addresses the need to provide fine-grained access control to data according to some policy. In contrast to traditional public-key encryption, data is encrypted not under a public key associated with the identity of the intended recipient, but rather under a set of attributes that can be possessed by one or more entities.

This concept falls into the more general paradigm of *functional encryption* (FE) [6,26]. In functional encryption, a setup algorithm produces a matching pair of public/secret keys (mpk, msk). The master public key mpk enables anyone to encrypt data and the master secret key msk enables its holder to issue functional keys —for example, a secret key sk_f for a certain function f. Given a ciphertext $ct = \mathsf{Encrypt}(mpk, x)$ of some message x, anyone who has sk_f can obtain $f(x)$. An important subclass of functional encryption is *Predicate Encryption* (PE) [7,17], and in particular *Inner-Product Predicate Encryption* (IPPE), where data can be accessed if and only if the ciphertext ct and the key sk satisfy a certain predicate $P(sk, ct)$ (namely, orthogonality).

To illustrate ABE, suppose we want to encrypt data stored on the university server such that it is accessible to anyone who "is a *network administrator*, or a *university student* and is taking the class *Introduction to Cryptography*". In single-authority ABE, an authority associated with the university can be in charge of verifying a user's identity and providing a key that certifies all her attributes. However, it is often impractical to rely on a single authority to verify and certify all possible attributes. Continuing the previous example: different university departments are in charge of protecting their own data independently, while also being supported by the IT department. In this case a policy can be "is *network administrator* or *computer science department staff*." The university IT and the computer science department are, in this case, two independent authorities that verify and issue keys for their own attributes.

Multiple works addressed this issue, starting with the ones by Chase [9], Chase and Chow [10], and Müller *et al.* [20] on multi-authority ABE, and following with a decentralized ABE scheme by Lewko and Waters [18]. This last construction enables to encrypt a ciphertext under a general access structure, while the corresponding secret keys are issued by independent authorities that do not need to communicate with each other, or with any central authority, and only refer to common parameters generated by a one-time trusted setup. Another construction with similar properties is by Okamoto and Takashima [22].[3]

Those schemes, however, do not address the often desirable property of hiding the encryption policy. Attribute-hiding means that the ciphertext policy is protected and remains unknown under inspection of the ciphertext. There is a weaker notion, called *weakly attribute-hiding*, which guarantees that the policy is hidden from anyone but a party that is capable of decrypting the ciphertext; *i.e.*, information about it is leaked only upon successful decryption. It is important to be able to hide the access policy since it can contain sensitive meta-data. One example is messaging or emails addressed to a group of users with certain attributes. In addition to protecting the content we may want to hide the target group. In Appendix D, we show that an adversary can reveal the encryption policy in Lewko-Waters' scheme, even when it is not explicitly given. While it is yet to be studied whether there is a policy inference attack on the scheme in [22] or whether it can be proved to be weakly attribute-hiding for inner-products, the paper neither claims, nor proves this property. In this work we set out to provide this important property in a decentralized setting.

While we previously illustrated the use of decentralized ABE with a simple toy example, it has practical real-world applications. Prior work on multi-authority ABE mentions supporting multiple authorities authorizing access to DRM-protected content [20], where hiding the policy is important to protect meta-data that can reveal potentially sensitive information about the content. To mention another example, ABE has recently been explored in the context of access-control for blockchains [24,25]. Indeed, in a blockchain setting, both

[3] The full version of a paper published in *PKC 2013* is mainly concerned with decentralized attribute-based signatures; however, it proposes a decentralized ABE scheme in Appendix E.

decentralization and policy protection are desirable. Attribute-based encryption or signatures can be an extension of the naïve multi-signature implementations in early blockchain-based crypto-currencies like Bitcoin [21], while policy-hiding can serve to preserve the recipient's privacy.

Our contributions. We propose a decentralized, policy-hiding ABE scheme that supports several very useful classes of access policies. To the best of our knowledge, this is the first practical scheme with a proof of the attribute-hiding property. We instantiate ABE from a decentralized inner-product predicate encryption for which we provide a construction and a proof of security in the random oracle model, under the k-linear assumption. The decryption procedure of the underlying inner-product predicate scheme is very efficient, and requires only a small number of pairing operations (as little as four when $k = 2$), independent of the vector size. Based on it, we devise attribute-hiding multi-authority ABE schemes supporting conjunctions, disjunctions, hidden-vector encryption, and threshold policies. On top of that, we add receiver privacy, by preventing individual authorities from knowing the full set of attributes possessed by the recipient when issuing keys.

Achieving security for a decentralized inner-product PE is not trivial since corrupt authorities can assist an adversary to satisfy the predicate by issuing illegitimate keys for specific vector elements. We prevent it by proposing secure policy encodings. This contribution is of independent interest, as it can be applied to any decentralized PE scheme, and is not particular to the construction used in this paper. It is important to note that while [17] proposes several policy encodings in terms of inner-product predicates, it is a single-authority scheme that does not face the problems that arise in a decentralized setting with some corrupt authorities. Our policy encodings specifically address this challenge, and as such constitute a novel contribution.

Decentralization of a PE scheme, that is naturally single-authority, comes at a certain price, and our scheme has two drawbacks: the first is that we require each authority to publish a Diffie-Hellman public key that is visible to other authorities, and the second is that a change of attributes on the receiver's side requires requesting new keys from all participating authorities and not only from the one that controls the changed attribute. However, those are affordable in an on-line setting where requesting a new key for an unchanged attribute may only require presenting a certificate (or key) that has been previously obtained from the authority.

Paper structure. In Section 2, we briefly present required preliminaries and state our computational assumptions. In Section 3, we formally define decentralized inner-product predicate encryption and the corresponding security game.

Section 3.3 describes an important enhancement to the key request procedure, which relieves the receiver from disclosing all its attributes to each authority. Section 4 explains how to turn the decentralized inner-product predicate encryption into an attribute-hiding ABE scheme. In Section 5, we address the presence of corrupt authorities, colluding with the adversary, by introducing randomizing-polynomial encodings. We formally define the security game in this new setting, and suggest several encodings that immunize the underlying predicate encryption scheme from corrupt authorities. Finally, we state related prior work and outline related open problems.

2 Background and Preliminaries

2.1 Inner-product predicate encryption

In a *predicate encryption* scheme, access to encrypted data is controlled by a certain predicate defined over the attributes included in the ciphertext policy. In particular, in *inner-product predicate encryption*, ciphertexts and secret keys are associated with vectors. In order to decrypt, the secret key has to be associated with a vector that is orthogonal to the vector associated with the ciphertext. The works of Katz *et al.* [17] and Chen *et al.* [11] are examples of such cryptosystems.

2.2 Pairing groups

Let $\mathcal{G}$ be an algorithm that on input a security parameter λ generates three groups $\mathbb{G}_1 = \langle g_1 \rangle$, $\mathbb{G}_2 = \langle g_2 \rangle$ and $\mathbb{G}_T$ of prime order p, admitting a pairing $\hat{e}\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ that has the following properties:

1. Bilinearity: for all $a, b \in \mathbb{Z}$, $\hat{e}({g_1}^a, {g_2}^b) = \hat{e}(g_1, g_2)^{ab}$;
2. Non-degeneracy: $\hat{e}(g_1, g_2) \neq 1$.

We write $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}) \leftarrow \mathcal{G}(1^\lambda)$. Groups $\mathbb{G}_1$ and $\mathbb{G}_2$ are called the source groups while $\mathbb{G}_T$ is called the target group.

2.3 Complexity assumptions

The *Symmetric External Diffie-Hellman (SXDH)* assumption states that the DDH assumption holds in both source groups $\mathbb{G}_1$ and $\mathbb{G}_2$. Formally, we have:

Assumption 1 (SXDH) *Given $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}) \leftarrow \mathcal{G}(1^\lambda)$, there exists no polynomial-time distinguisher that can decide with a non-negligible advantage between the distributions $\mathscr{D}_0 = (g_1, g_2, {g_1}^a, {g_1}^b, {g_1}^{ab})$ and $\mathscr{D}_1 = (g_1, g_2, {g_1}^a, {g_1}^b, {g_1}^r)$ where $a, b, r \xleftarrow{\$} \mathbb{Z}_p$, and symmetrically, exchanging the roles of $\mathbb{G}_1$ and $\mathbb{G}_2$, between the distributions $\mathscr{D}_0 = (g_1, g_2, {g_2}^a, {g_2}^b, {g_2}^{ab})$ and $\mathscr{D}_1 = (g_1, g_2, {g_2}^a, {g_2}^b, {g_2}^r)$.*

The SXDH assumption can be weakened using higher-rank matrices [13]. It is useful to introduce some notation. For $a_1, a_2, \dots, a_k \xleftarrow{\$} {\mathbb{Z}_p}^*$, consider

$$\mathbf{A} = \begin{pmatrix} a_1 & 0 & \dots & 0 \\ 0 & a_2 & & 0 \\ \vdots & & \ddots & \\ 0 & 0 & & a_k \\ 1 & 1 & \dots & 1 \end{pmatrix} \in {\mathbb{Z}_p}^{(k+1)\times k} \quad \text{and} \quad \boldsymbol{a}^\perp = \begin{pmatrix} {a_1}^{-1} \\ {a_2}^{-1} \\ \vdots \\ {a_k}^{-1} \\ -1 \end{pmatrix} \in {\mathbb{Z}_p}^{(k+1)} \enspace .$$

Then, $\mathbf{A}^{\intercal} \boldsymbol{a}^{\perp} = \mathbf{0}$. We let $\mathfrak{D}_k(\mathbb{Z}_p)$ denote the distribution induced by the previous sampling.

Assumption 2 (k-Lin in $\mathbb{G}_1$) *Given $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}) \leftarrow \mathcal{G}(1^\lambda)$, there exists no polynomial-time distinguisher that can decide with a non-negligible advantage between the distributions $\mathscr{D}_0 = (g_1, g_2, {g_1}^{\mathbf{A}}, {g_1}^{\mathbf{A}\boldsymbol{s}})$ and $\mathscr{D}_1 = (g_1, g_2, {g_1}^{\mathbf{A}}, {g_1}^{\boldsymbol{z}})$ where $(\mathbf{A}, \boldsymbol{a}^{\perp}) \overset{\$}{\leftarrow} \mathfrak{D}_k(\mathbb{Z}_p)$, $\boldsymbol{s} \overset{\$}{\leftarrow} {\mathbb{Z}_p}^k$, and $\boldsymbol{z} \overset{\$}{\leftarrow} {\mathbb{Z}_p}^{k+1}$.*

The k-Lin assumption in $\mathbb{G}_2$ is defined similarly. By abuse of language, the k-Lin assumption will refer to the k-Lin assumption in both $\mathbb{G}_1$ and $\mathbb{G}_2$.

3 Decentralized Inner-Product Predicate Encryption

Our goal is supporting a multi-authority setting, where keys for different attributes can be requested from n independent authorities, that do not need to communicate with each other or with a central authority. In inner-product predicate encryption, keys are issued by a central authority, given a vector $\boldsymbol{v}$, to eligible parties. We decentralize the key generation algorithm, such that key-parts are issued separately for different vector elements v_i, by n independent authorities. Without loss of generality, we assume that authority i issues keys for attribute number i. Those key-parts are then combined to form a secret key corresponding to the vector $\boldsymbol{v} = (v_1, \ldots, v_n)$.

For simplicity, we first construct a scheme that is weakly attribute-hiding in the absence of corrupt authorities. It is mostly useful as a stepping-stone, to understand how, in combination with special policy encodings, it becomes secure in the presence of corrupt authorities.

Definition 1. *A decentralized inner-product predicate encryption scheme consists of a tuple of PPT algorithms,* (Setup, AuthSetup, KeyGen, Encrypt, Decrypt)*, such that*

- Setup *takes as input the security parameter λ and outputs the master public parameters pp.*
- AuthSetup *takes as input the public parameters pp and the authority index i, and outputs the authority's secret key SK_i and public key PK_i.*
- KeyGen *takes as input the master public parameters pp, the authority index i, its secret key SK_i, the public parameters $\{PK_j\}_{j \neq i}$ of other authorities, a user's global identifier* GID *and the attribute vector $\boldsymbol{v}$, and outputs a secret key part $sk_{i,\mathsf{GID},\boldsymbol{v}}$.*
- Encrypt *takes as input the master public parameters pp, the public parameters of the authorities $\{PK_i\}$, the ciphertext policy vector $\boldsymbol{x}$ and a message M in the message space, and outputs a ciphertext ct.*
  We express it as $ct \overset{\$}{\leftarrow} \mathsf{Encrypt}_{pp}(\boldsymbol{x}, M)$.
- Decrypt *takes as input the collection of obtained secret keys $\{sk_{i,\mathsf{GID},\boldsymbol{v}}\}_{i=1}^n$ and the ciphertext ct, and outputs either the message M or the special symbol $\perp$. We express it as $M \leftarrow \mathsf{Decrypt}\big(\{sk_{i,\mathsf{GID},\boldsymbol{v}}\}, ct\big)$.*

For correctness we require that for all $pp, \boldsymbol{x}, \boldsymbol{v}, sk_{i,\mathsf{GID},\boldsymbol{v}}$:

$$\mathsf{Decrypt}\big(\{sk_{i,\mathsf{GID},\boldsymbol{v}}\}_{i=1}^{n}, \mathsf{Encrypt}_{pp}(\boldsymbol{x}, M)\big) = \begin{cases} M & \text{if } \langle \boldsymbol{x}, \boldsymbol{v} \rangle = 0 \\ \perp & \text{otherwise} \end{cases}$$

with all but negligible probability.

Definition 2 captures security in the absence of corrupt authorities.

Definition 2. *A decentralized inner-product predicate encryption scheme is* weakly attribute-hiding*, with respect to a set of attributes Σ, if for all PPT adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in winning the following game against a challenger $\mathcal{S}$ is negligible in the security parameter:*

1. *$\mathcal{S}$ runs Setup to generate pp and hands it to $\mathcal{A}$.*
2. *$\mathcal{S}$ runs $\mathsf{AuthSetup}(pp, i)$ for each authority i, and gives $\{PK_i\}$ to $\mathcal{A}$.*
3. *$\mathcal{A}$ may request keys for vectors $\boldsymbol{v}$, indicating possession of attributes in Σ. In response, $\mathcal{S}$ gives $\mathcal{A}$ the corresponding keys $sk_{i,\mathsf{GID},\boldsymbol{v}}$ produced by $\mathsf{KeyGen}_{pp}(i, SK_i, \mathsf{GID}, \boldsymbol{v})$. GID is the global identifier of the requesting user; its role is explained in Section 3.1.*
4. *$\mathcal{A}$ outputs two policy vectors $\boldsymbol{x_0}, \boldsymbol{x_1}$ and two equal-length messages M_0, M_1. $\mathcal{S}$ checks that none of the previously queried attribute vectors $\boldsymbol{v}$ are orthogonal to $\boldsymbol{x_0}$ or $\boldsymbol{x_1}$; i.e., $\langle \boldsymbol{x_0}, \boldsymbol{v} \rangle \neq 0 \wedge \langle \boldsymbol{x_1}, \boldsymbol{v} \rangle \neq 0$ for all previously requested $\boldsymbol{v}$. The challenger chooses a random bit b and gives $\mathcal{A}$ the ciphertext $ct \overset{\$}{\leftarrow} \mathsf{Encrypt}_{pp}(\boldsymbol{x_b}, M_b)$.*
5. *$\mathcal{A}$ may request more keys for vectors $\boldsymbol{v}$ as they are not orthogonal to $\boldsymbol{x_0}, \boldsymbol{x_1}$.*
6. *$\mathcal{A}$ outputs a bit b' and wins if $b' = b$.*

The advantage of $\mathcal{A}$ is defined as $\mathrm{adv}(\mathcal{A}) = \left|\Pr[b = b'] - \frac{1}{2}\right|$.

Note 1. The way attribute possession is encoded in the vector $\boldsymbol{v}$ is explained further, when we discuss instantiations of ABE schemes using predicate encryption.

Definition 3 captures security in the presence of corrupt authorities. Here, the adversary does not know $\boldsymbol{x_0}, \boldsymbol{x_1}$ explicitly, as opposed to Definition 2. It provides the policies in the form of a boolean formula, or a threshold t-out-of-n over a set of admissible attributes, or a matching pattern, etc.

Definition 3. *We define a game between an adversary $\mathcal{A}$ and a challenger $\mathcal{S}$:*

1. *$\mathcal{S}$ picks a random bit $b \in \{0, 1\}$ and outputs the public parameters pp.*
2. *$\mathcal{A}$ outputs the set of corrupt authorities $\mathcal{A}^\star$, and provides $\mathcal{S}$ with their public parameters.*
3. *$\mathcal{S}$ runs $\mathsf{AuthSetup}$ for each one of the non-corrupt authorities, and gives the public parameters to $\mathcal{A}$.*
4. *$\mathcal{A}$ outputs two policies π_0, π_1 and two equal-length messages M_0, M_1. The policies require attributes controlled by non-corrupt authorities, and must agree on the attributes controlled by the corrupt authorities $\mathcal{A}^\star$.*

5. *$\mathcal{S}$ outputs a challenge ciphertext $ct \xleftarrow{\$} \mathsf{Encrypt}_{pp}(\boldsymbol{x_b}, M_b)$, where $\boldsymbol{x_b}$ is the encoding vector of policy π_b.*
6. *$\mathcal{A}$ generates key requests for the different authorities. $\mathcal{S}$ checks that the set of attributes, controlled by the authorities for which a non-zero key has been requested, cannot satisfy either of the two policies π_0, π_1.*
7. *$\mathcal{A}$ outputs a guess b'. If $b' = b$ it wins the game.*

Definition 4 (Security). *The scheme is secure (against static corruption of authorities) if any PPT adversary $\mathcal{A}$ has only negligible advantage in winning the game in Section 3 against a challenger $\mathcal{S}$.*

3.1 Collusion prevention and protection against corrupt authorities

A fundamental requirement from an ABE scheme is to prevent collusion between users. Let u_1 and u_2 be two users, possessing sets of key-parts K_1, K_2. K_1 contains key-parts that enable obtaining a secret key to any $\boldsymbol{v_1} \in V_1$, and K_2 contains key-parts that enable obtaining a secret key to any $\boldsymbol{v_2} \in V_2$. u_1 and u_2 must not be able to mix their key-parts in a way that gives them a secret key to a new vector $\boldsymbol{v}$ such that $\boldsymbol{v} \notin V_1$ and $\boldsymbol{v} \notin V_2$. For example, to enforce the policy "*is a university student, and taking Introduction to Cryptography*," it is not enough to secret share the message, and encrypt it under the public-keys of the two authorities. Otherwise, two users having only one of the attributes each, can collude to decrypt the ciphertext. Therefore, all works on multi-authority ABE, including ours, address collusion prevention as one of the main challenges.

Prior works on multi-authority ABE [9,10,18] assign a global identifier (GID) to each user. It is used to associate every secret key with an identity by incorporating it into the decryption keys issued by the authorities. In our setting, it is not sufficient to restrict combination of keys to the same GID. Depending on the policy encoding, we may have to ensure that keys are issued for a well-formed attribute-vector $\boldsymbol{v}$. For instance, in a threshold scheme, if a corrupt authority issues a key for a value $v_i > 1$, the user may be able to decrypt despite not having sufficient attributes to satisfy the policy. For our basic scheme, we require the user to supply its attribute vector $\boldsymbol{v}$ when requesting a key, and tie the issued keys to the tuple $(\mathsf{GID}, \boldsymbol{v})$. This imposes the already mentioned requirement, on part of the receiver, to update keys when attributes change.

We use hash functions $\mathsf{H}_1(\mathsf{GID}, \boldsymbol{v}), \ldots, \mathsf{H}_{k+1}(\mathsf{GID}, \boldsymbol{v})$, modeled as random oracles, to map $(\mathsf{GID}, \boldsymbol{v})$ to random elements. This ensures that different authorities issue keys that correspond to some common parameter. As we show in Section 3.3, we can replace the attribute vector with a commitment. The binding property, in composition with the random oracle, guarantees that the authorities issue keys for a common attribute vector. This modular combination enables us to extend the scheme with receiver privacy, without changing the core construction or its proof of security.

Note that it does not prevent corrupted authorities from computing a key for a different value than that appearing in $\boldsymbol{v}$. However, in the absence of corrupted authorities, it prevents an adversary from obtaining a key to an invalid attribute vector, as well as collusion between multiple adversaries.

Minimal trust requirement. Given a set of l attributes (and l corresponding controlling authorities), we require one special authority (we refer to it as the $l+1$ authority) to be trusted to issue keys only for $v_{l+1} \neq 0$. Note that the authority does not get to learn the policy, or the paylod. This requirement becomes clear once we explain the way policies are encoded. It also ensures that no keys for $\boldsymbol{v} = \mathbf{0}$ are ever issued.

3.2 Construction

We build on the elegant predicate encodings framework by Chen and Wee [12], and the single-authority ZIPE scheme by Chen *et al.* [11]. They use dual-system groups instantiated with prime-order bilinear groups, based on the k-linear computational hardness assumption. In particular, the choice of $k = 1$ corresponds to the External Diffie-Hellman (XDH) assumption, and choosing $k = 2$ corresponds to the decision-linear (DLIN) assumption. Essentially, we achieve decentralization by substituting the randomness, chosen by the sender in their framework, with a publicly computable hash function, modeled as a random oracle, that can be computed by all parties. We also introduce masking terms that force the receiver to combine the key parts received from various authorities, prior to using them in any way. We use a random oracle $\mathcal{H}\colon \mathbb{G}_2 \times \{0,1\}^\lambda \times {\mathbb{Z}_p}^{l+1} \to {\mathbb{Z}_p}^{k+1}$, to generate masking terms that depend on a combination of an authority, the GID, and the attribute vector $\boldsymbol{v}$ (or a commitment to it). It is a simple way to ensure that the receiver cannot use the key parts obtained from the different authorities prior to combining them as specified in the construction. It implies a requirement for certain minimal coordination between authorities. Each one of them publishes a Diffie-Hellman public key, visible to the others. In this sense, our scheme misses the desirable property of full-decentralization, that doesn't require any coordination between authorities whatsoever beyond referring to common public parameters published on setup.

The scheme is as follows:

- $\mathsf{Setup}(\lambda)$: On input of a security parameter λ, the algorithm outputs $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e})$. Let g_1, g_2 be two generators of $\mathbb{G}_1, \mathbb{G}_2$ respectively. It picks a random matrix $\mathbf{A} \in {\mathbb{Z}_p}^{(k+1)\times k}$ and a random matrix $\mathbf{U} \in {\mathbb{Z}_p}^{(k+1)\times(k+1)}$, and publishes the public parameters

$$pp = \left\{g_1, g_2, {g_1}^{\mathbf{A}}, {g_1}^{\mathbf{U}^\intercal \mathbf{A}}\right\}$$

- $\mathsf{AuthSetup}(pp, i)$: The algorithm samples a random matrix $\mathbf{W_i} \in {\mathbb{Z}_p}^{(k+1)\times(k+1)}$, a vector $\boldsymbol{\alpha_i} \xleftarrow{\$} {\mathbb{Z}_p}^{k+1}$ and a random $\sigma_i \in \mathbb{Z}_p$. The authority stores the secret key $SK_i = \{\mathbf{W_i}, \boldsymbol{\alpha_i}, \sigma_i\}$ and publishes the public key

$$PK_i = \left\{{g_1}^{\mathbf{W_i}^\intercal \mathbf{A}}, \hat{e}(g_1, g_2)^{\boldsymbol{\alpha}_i^\intercal \mathbf{A}}, y_i = {g_2}^{\sigma_i}\right\}$$

- $\mathsf{Encrypt}_{pp}(\{PK_i\}, \boldsymbol{x}, m)$: Let $\boldsymbol{x} = (x_1, \ldots, x_n) \in {\mathbb{Z}_p}^n$. The algorithm chooses a random vector $\boldsymbol{s} \in {\mathbb{Z}_p}^k$ and outputs the ciphertext $\boldsymbol{C}$ consisting of the components

$$C_0 = {g_1}^{\mathbf{A}\boldsymbol{s}} \qquad C_i = {g_1}^{(x_i\mathbf{U}^\intercal + \mathbf{W_i}^\intercal)\mathbf{A}\boldsymbol{s}}$$

$$C' = m \cdot \prod_{i=1}^{n} \hat{e}(g_1, g_2)^{\boldsymbol{\alpha_i}^\intercal \mathbf{A}\boldsymbol{s}} = m \cdot \hat{e}(g_1, g_2)^{\boldsymbol{\alpha}^\intercal \mathbf{A}\boldsymbol{s}}$$

where $\boldsymbol{\alpha} = \sum_{i=1}^{n} \boldsymbol{\alpha_i}$.

- $\mathsf{KeyGen}_{pp}(\{PK_i\}, SK_i, \mathsf{GID}, \boldsymbol{v})$: The authority takes the public keys of all other authorities, and computes a masking value $\boldsymbol{\mu_i} \in \mathbb{Z}_p^{k+1}$

  $$\boldsymbol{\mu_i} = \sum_{j=1}^{i-1} \mathcal{H}({y_j}^{\sigma_i}, \mathsf{GID}, \boldsymbol{v}) - \sum_{j=i+1}^{n} \mathcal{H}({y_j}^{\sigma_i}, \mathsf{GID}, \boldsymbol{v})$$

  It is easy to check that $\sum_{i=1}^{n} \boldsymbol{\mu_i} = \mathbf{0}$.
  We use $\mathsf{H}_1(\mathsf{GID}, \boldsymbol{v}), \ldots, \mathsf{H}_{k+1}(\mathsf{GID}, \boldsymbol{v})$ to generate ${g_2}^{\boldsymbol{h}}$ where $\boldsymbol{h} \in {\mathbb{Z}_p}^{k+1}$. Note that the exponent $\boldsymbol{h}$ is unknown and is defined implicitly by the hash functions. We denote

  $$\mathsf{H}(\mathsf{GID}, \boldsymbol{v}) = \big(\mathsf{H}_1(\mathsf{GID}, \boldsymbol{v}), \ldots, \mathsf{H}_{k+1}(\mathsf{GID}, \boldsymbol{v})\big)^\intercal$$

  The algorithm outputs the key $sk_{i,\mathsf{GID},\boldsymbol{v}}$ which consists of

  $$K_i = {g_2}^{\boldsymbol{\alpha_i} - v_i \mathbf{W_i}\boldsymbol{h} + \boldsymbol{\mu_i}}$$

- $\mathsf{Decrypt}_{pp}(\{sk_{i,\mathsf{GID},\boldsymbol{v}}\}, \boldsymbol{C}, \boldsymbol{v})$: Compute

  $$\hat{e}\big(C_0, \textstyle\prod_{i=1}^{n} K_i\big) \cdot \hat{e}\big(\textstyle\prod_{i=1}^{n} C_i^{v_i}, \mathsf{H}(\mathsf{GID}, \boldsymbol{v})\big) = \hat{e}(g_1, g_2)^{\boldsymbol{\alpha}^\intercal \mathbf{A}\boldsymbol{s}}$$

  and recover the message by computing

  $$C' / \, \hat{e}(g_1, g_2)^{\boldsymbol{\alpha}^\intercal \mathbf{A}\boldsymbol{s}} = m$$

Correctness. Let $\boldsymbol{C} = \big(C_0, \{C_i\}_{i=1}^{n}, C'\big)$ and $\{K_i := sk_{i,\mathsf{GID},\boldsymbol{v}}\}_{i=1}^{n}$ be as described above. Then

$$\begin{aligned}
&\hat{e}\big(C_0, \textstyle\prod_{i=1}^{n} K_i\big) \cdot \hat{e}\big(\textstyle\prod_{i=1}^{n} C_i^{v_i}, \mathsf{H}(\mathsf{GID}, \boldsymbol{v})\big) \\
&= \hat{e}({g_1}^{\mathbf{A}\boldsymbol{s}}, {g_2}^{\sum_{i=1}^{n} \boldsymbol{\alpha_i} - v_i \mathbf{W_i}\boldsymbol{h} + \boldsymbol{\mu_i}}) \cdot \hat{e}({g_1}^{\sum_{i=1}^{n} v_i(x_i\mathbf{U}^\intercal + \mathbf{W_i}^\intercal)\mathbf{A}\boldsymbol{s}}, {g_2}^{\boldsymbol{h}}) \\
&= \hat{e}(g_1, g_2)^{\boldsymbol{\alpha}^\intercal \mathbf{A}\boldsymbol{s} - \sum_{i=1}^{n} v_i \boldsymbol{h}^\intercal \mathbf{W_i}^\intercal \mathbf{A}\boldsymbol{s}} \cdot \hat{e}(g_1, g_2)^{\langle \boldsymbol{x}, \boldsymbol{v} \rangle \boldsymbol{h}^\intercal \mathbf{U}^\intercal \mathbf{A}\boldsymbol{s} + \sum_{i=1}^{n} v_i \boldsymbol{h}^\intercal \mathbf{W_i}^\intercal \mathbf{A}\boldsymbol{s}} \\
&= \hat{e}(g_1, g_2)^{\boldsymbol{\alpha}^\intercal \mathbf{A}\boldsymbol{s}} \cdot \hat{e}(g_1, g_2)^{\langle \boldsymbol{x}, \boldsymbol{v} \rangle \boldsymbol{h}^\intercal \mathbf{U}^\intercal \mathbf{A}\boldsymbol{s}}
\end{aligned}$$

If $\langle \boldsymbol{x}, \boldsymbol{v} \rangle = 0$, we obtain $\hat{e}(g_1, g_2)^{\boldsymbol{\alpha}^\intercal \mathbf{A}\boldsymbol{s}}$ and can recover the message.

Note 2. Looking at the key format it is easy to see why this construction, in general, requires the masking terms $\boldsymbol{\mu_i}$, and why their generation requires taking $\boldsymbol{v}$ as input. Without it, an adversary can ask for keys corresponding to $v_i = 0$

for $\forall i = 1..n$, obtaining ${g_2}^{\boldsymbol{\alpha}}$. That, in turn, enables to decrypt any ciphertext by pairing with C_0. By examining the vector $\boldsymbol{v}$, and tying the generated key to it, the authorities ensure that the adversary doesn't obtain a key to an all-zeros vector ($\boldsymbol{v} = \boldsymbol{0}$).

It also becomes clear why we need to trust the $l+1$ authority to refuse issuing keys for $v_{l+1} = 0$. If that would have been the case, an adversary colluding with a corrupt i-th authority would request keys from all other authorities presenting $\boldsymbol{v} = \boldsymbol{e_i}$ in the request, while in fact obtaining a key for $v_i = 0$ from the corrupt authority. A formal proof of security is provided in Appendix B.

3.3 Receiver privacy

each authority it requests a key from. As a result, the authority learns not only whether the user has the attribute which it controls, but also all other attributes it possesses. This is an apparent violation of the user's privacy in a decentralized setting.

We propose an enhancement that provides this additional privacy protection. While we want to ensure consistency of the keys issued by different authorities, and some properties of the vector they were issued for, we can avoid providing $\boldsymbol{v}$ in the clear. We satisfy consistency and privacy using commitments. Proving possession of attributes and certain properties of $\boldsymbol{v}$ is done by partial openings.

First, we propose to provide a vector $\boldsymbol{c} \in {\mathbb{Z}_p}^n$, consisting of one-bit Pedersen commitments [23] to the values $\{v_i\}_{i=1}^n$, instead of $\boldsymbol{v}$ itself, when requesting a key. This method is useful when we do not use randomizing-polynomial encodings (discussed in section 5) in the ciphertext, relying on honest authorities. In this case, valid receiver attribute vectors are binary, consisting of 0 and 1 elements, and authorities need to verify this property. This property is enforced by one-bit Pedersen commitments, with each element checked by a different authority. For encodings that require the receiver to request keys for arbitrary values, generalized Pedersen commitments can be used. In both cases, the input to the hash is the sequence of commitments.

Second, we propose to reduce the communication between the receiver and the authority by compressing the commitment vector into a single value using an accumulation technique. Concise *vector commitments* were first introduced by Libert and Yung [19], and further developed by Catalano and Fiore [8]. They enable committing to an ordered sequence of values, and later on opening the commitment in a certain position, proving that no other value would have resulted in the previously supplied commitment. This is called *position-binding*. The authors propose two different constructions —one based on the Computational Diffie-Hellman assumption (CDH), and another one based on the RSA assumption. Both constructions result in a constant size commitment. A formal definition of a VC scheme and of the position-binding property is provided in Appendix A.

We use it to hide the set of receiver attributes ($\boldsymbol{v}$) from the authorities, while guaranteeing that only key parts issued for the same $\boldsymbol{v}$ can be combined to a valid key. Concretely, we use the vector commitment C as an input to the hash functions $\mathsf{H}_1, \ldots, \mathsf{H}_{k+1}$. To request a key from authority i, the receiver send C, along with an opening in position i. The authority verifies the proof, and generates the key using $\mathsf{H}(\mathsf{GID}, C)$. Former security guarantees are maintained by the fact that the commitment is binding, while attribute-privacy is achieved by the fact that the commitment is hiding.

Application to our scheme. The modified Setup algorithm of our ABE scheme uses VC.Setup in order to generate the public parameters for the vector commitments. Prior to requesting keys, the user executes VC.Commit to produce a commitment C to $\boldsymbol{v}$. Upon requesting a key-part from authority i, it executes VC.Open to produce a proof P_i for the value at the i-th position, and supplies C and P_i along with the key request. The authority runs VC.Verify to verify the proof against the commitment C, and uses $\mathsf{H}(\mathsf{GID}, C)$ to generate the key.

Security. We argue that an adversary cannot mix-and-match keys issued for different attribute vectors. Since the commitment to the vector is binding, the adversary is unable, except with negligible probability, to find two inputs that would yield the same commitment. Therefore, every unique attribute vector results in a different input to the random oracle that is used by KeyGen.

4 Decentralized Policy-Hiding ABE

We use the constructed inner-product predicate encryption to build policy-hiding multi-authority ABE. The naïve encodings are simple, assuming the authorities are trusted to issue keys as specified below. In the following, we explain how the sender encodes the policy, and how the receiver issues key requests to the authorities.

We begin with describing how to build an attribute-based encryption from inner-product predicate encryption. We use an inner-product predicate encryption scheme as a building block, and demonstrate encodings for conjunctions, threshold policies, and hidden-vector encryption (HVE).

Exact threshold ABE. Let $A = \{1, \ldots, l\}$ be the enumeration of all supported attributes. Let S be the subset of attributes in the ciphertext policy. Let S' be the subset of attributes possessed by a party attempting to decrypt the ciphertext. We require that it would be possible for a party to decrypt the ciphertext if it possesses exactly t of those attributes; *i.e.*, if $|S \cap S'| = t$.

We instantiate a $l+1$ dimensional inner-product predicate encryption scheme. To encrypt a messages under such policy we construct a vector $\boldsymbol{x} \in \mathbb{Z}_p^{l+1}$ as follows

1. Set the first l entries such that $x_i = \begin{cases} 1 & i \in S \\ 0 & i \notin S \end{cases}$
2. Set the $l+1$ entry to $-t$; *i.e.*, $x_{l+1} = -t \pmod p$

and output the ciphertext $CT_{\boldsymbol{x}} = \mathsf{Encrypt}_{PK}\left(\boldsymbol{x}, M\right)$. To obtain a decryption key for the attributes in S', the receiver constructs a vector $\boldsymbol{v} \in \mathbb{Z}_N^{l+1}$ as follows

1. Set the first l entries such that $v_i = \begin{cases} 1 & i \in S' \\ 0 & i \notin S' \end{cases}$
2. Set the $l+1$ entry to 1; *i.e.*, $v_{l+1} = 1$

and execute $\mathsf{GenKey}\left(\boldsymbol{v}\right)$ to obtain $SK_{\boldsymbol{v}}$. Since $|S \cup S'| = t$ we have exactly t matching entries with the value 1 that cancel out with $-t$, yielding $\langle \boldsymbol{x}, \boldsymbol{v} \rangle = 0$, thereby satisfying the predicate and enabling decryption.

This encoding is only secure in the absence of corrupt authorities. An adversary, that does not have enough required attributes to satisfy the policy, may collude with a corrupt authority and ask it to provide it with a key for a value $v_i > 1$ such that $\langle \boldsymbol{v}, \boldsymbol{x} \rangle = 0$. In Section 5 we suggest another, less straightforward encoding, that immunizes the scheme against corrupt authorities that are willing to generate keys for arbitrary values.

Threshold ABE. A general threshold algorithm requires $l - t + 1$ invocations of the exact threshold decryption in the worst case, or $O\left(l\right)$ invocations if t is small compared to l. The receiver starts with the subset of its first t attributes, denoted S'_t and constructs the corresponding vector $\boldsymbol{v}_t$. It requests the corresponding secret key, and attempts decryption. If decryption fails, it knows that it did not hit the exact threshold of common attributes. It adds another attribute, forming the set S'_{t+1}, and constructs the corresponding vector $\boldsymbol{v}_{t+1}$. Once again, it requests the corresponding secret key and attempts to decrypt. It continues until it hits the exact threshold, or until all possessed attributes are included.

Corrupt Authorities. Matters become more complicated in the presence of corrupt authorities colluding with the adversary. In case the encryption policy includes an attribute controlled by a corrupted authority, the adversary can use it to issue a secret key for any value v_i and break the naïve construction.

Let us consider a threshold-policy t-out-of-n, and let the sender include attribute i in the ciphertext policy; *i.e.*, $x_i = 1$. If the adversary has some prior knowledge that this attribute is included in the ciphertext policy, it can request a key component corresponding to $v_i = t$. Then it combines it with key components corresponding to $v_j = 0$ for all $j \neq i$ and the key component corresponding to the threshold entry $v_{l+1} = 1$. In the inner product, $\sum_{i=1}^{l} x_i v_i = t$ and $x_{l+1} v_{l+1} = -t$ cancel out, resulting in $\langle \boldsymbol{x}, \boldsymbol{v} \rangle = 0$ and thus successful decryption despite not possessing enough attributes. The attack stems from the ability to request key components for arbitrary inputs.

The $l+1$ authority has to be honest since it controls the threshold setting. If this authority is corrupt, it can issue a secret key component corresponding to a lower (or a zero) threshold - a condition that is much easier (or trivial) for the adversary to satisfy.

In Section 5, we propose a threshold encoding that is secure in the presence of corrupt authorities. However, this scheme, while more restrictive, requires linear decoding time, and in certain cases may be preferable to the scheme in Section 5.

Conjunctions. Conjunctions are an important class of policies that state that the receiver must possess a certain set of attributes in order to decrypt the message. They are one of the most useful policies in real-world scenarios, since access policies would often specify a combination of several properties that the receiver must have. Suppose we have a total set of attributes $\varSigma$, indexed from 1 to l, and we require possession of a subset S. We encode it as a vector $\boldsymbol{x} \in \mathbb{Z}_p^{l+1}$ as follows:

1. Set the first l entries such that $x_i = \begin{cases} r_i \xleftarrow{\$} \mathbb{Z}_p & i \in S \\ 0 & i \notin S \end{cases}$
2. Set the $l+1$ entry to $-\sum_{i=1}^{l} r_i \pmod{p}$.

Given the receiver's set of attributes R, the vector $\boldsymbol{v}$ is set as follows:

1. Set the first l entries such that $v_i = \begin{cases} 1 & i \in R \\ 0 & i \notin R \end{cases}$
2. Set the $l+1$ entry to 1

We set the elements corresponding to attributes in S to random values, and the last element to minus their sum. Thus, an inner-product with a vector that has 1-s in all indices corresponding to the required attributes, yields 0, resolving the policy, as illustrated below:

$$\begin{array}{|c|c|c|c|c|c|} \hline v_1 & v_2 & v_3 & \dots & v_l & v_{l+1} \\ \hline 1 & 1 & 0 & \dots & 0 & 1 \\ \hline \end{array} \quad \cdot \quad \begin{array}{|c|c|c|c|c|c|} \hline x_1 & x_2 & x_3 & \dots & x_l & x_{l+1} \\ \hline r_1 & r_2 & 0 & \dots & 0 & -(r_1 + r_2) \\ \hline \end{array} \quad = \quad 0$$

Note that in this case the encoding itself immunizes the scheme against corrupted authorities. Normally, an honest authority should only issue keys for values $v_i = 0$ or $v_i = 1$, indicating absence or possession of attribute i. However, a corrupt authority can provide an adversary with a key issued for an adversarially chosen value v_i, in an attempt to satisfy the policy without actually having all necessary attributes. By encoding the required attributes using randomly sampled r_i-s over a large field, we provide information theoretic security against an attempt to craft a key by adversarially picking a value v_i that would result in a zero inner-product. Security of this encoding is captured by Definition 3. It readily follows from the probability that the adversary correctly guesses which value it should craft to cancel out the last entry, which is negligible ($\frac{1}{p}$).

Hidden-Vector Encryption. Hidden-Vector Encryption (HVE) was first introduced by Boneh and Waters [7]. Given a set of attributes $\varSigma$, let $\varSigma_* = S \cup \{*\}$, and the HVE predicate is

$$\mathsf{P}^{hve}_{a_1,\dots,a_l}(x_1, \dots, x_l) = \begin{cases} 1 & \forall i : a_i = x_i \vee a_i = * \\ 0 & \text{otherwise} \end{cases} \quad .$$

Simply put, this is a pattern matching on an input, where $a_i = *$ denotes a wildcard ("don't care"), indicating that at position i, the input vector is allowed to have an arbitrary value. For l attributes, we need to use vectors of size $l+1$. The ciphertext policy vector is constructed by sampling l random values $r_i \xleftarrow{\$} \mathbb{Z}_p$, and setting

$$x_i = \begin{cases} r_i & X_i \neq * \\ 0 & X_i = * \end{cases} : \forall i = 1..l \qquad \text{and} \qquad x_{l+1} = -\sum_{i=1}^{l} r_i X_i \pmod p \ .$$

The receiver attribute vector is given by

$$v_i = \begin{cases} a_i & i \in S \\ 0 & i \notin S \end{cases} : \forall i = 1..l$$

where S is the set of attributes possessed by the receiver, and $v_{l+1} = 1$.

As in the encoding for conjunctions, this encoding is secure even in the presence of corrupt authorities. The proof of security is similar to the one for conjunctions.

5 Randomizing-Polynomial Encodings

We propose encoding policies using polynomials with random coefficients as a way to protect the scheme against corrupt authorities that extend the adversary's degrees of freedom in obtaining keys. Specifically, a corrupt authority i enables the adversary to obtain a key-part K_i corresponding to an arbitrary value v_i, instead of being limited to 0 or 1.

By weak attribute-hiding, the adversary cannot infer the vector $\boldsymbol{x}$ used to encode the access policy. The sender generates a randomized multivariate polynomial P expressing the policy, and sets x_i to its coefficients, and $x_{l+1} = \pm P(0, \ldots, 0)$, depending on the policy type.

The receiver does not know the polynomial, and in order to obtain 0 it has to evaluate P at $(0, \ldots, 0)$. It requests keys from the authorities for either 0 (when it doesn't have the corresponding attribute), or some non-zero value depending on the type of encoded policy (in case it has the attribute). The $l+1$ authority is special in that it only issues a key for $v_{l+1} = 1$. Attempting reconstruction using any other coefficients would result in a non-zero inner-product with high probability. In the following, we specify concrete encodings and receiver procedures for several useful access policies.

5.1 Examples of encodings for different policies

Threshold policy. Let S be the set of attributes that are considered admissible by the sender (out of the total l attributes), and $n = |S|$. To implement a threshold policy t-out-of-n ($n \leq l$), the sender samples t random coefficients $a_i \xleftarrow{\$} \mathbb{Z}_p$ that define a monic polynomial $P(x)$ of degree t:

$$P(x) = x^t + a_{t-1}x^{t-1} + \cdots + a_1 x + a_0 \pmod p$$

The sender generates n shares of P at publicly known points $\{z_i : i \in S\}$ and sets

$$x_i = \begin{cases} P(z_i) & i \in S \\ 0 & i \notin S \end{cases} : \forall i = 1..l \qquad \text{and} \qquad x_{l+1} = -P(0) = -a_0$$

For example, a ciphertext policy vector can be

x_1	x_2	x_3	...	x_l	x_{l+1}
$P(z_1)$	$P(z_2)$	0	...	$P(z_l)$	$-P(0)$

The receiver computes Lagrange polynomials λ_i at 0, using $\{z_i\}$ corresponding to a subset of t attributes in its possession, and requests the corresponding keys from the attribute authorities:

$$v_i = \begin{cases} \lambda_i & i \in S \\ 0 & i \notin S \end{cases} : \forall i = 1..l \qquad \text{and} \qquad v_{l+1} = 1$$

The decryption procedure effectively performs Lagrange interpolation in the exponent, over the shares encoded in the ciphertext. If decryption fails, we form another subset of t attributes, recompute the Lagrange polynomials and request the corresponding keys, and retry decrypting. The receiver repeats this until it succeeds, or until it used all attributes in its possession. A receiver that is not able to decrypt does not learn the set of admissible attributes S. A drawback of this method, is that it requires attempting $O\binom{l}{t} \leq l^t$ attribute subsets. It is polynomial in the overall number of attributes, and exponential in the threshold parameter. Hence, it is practical for small thresholds.

CNF and DNF formulas. Boolean CNF and DNF formulas can be represented by multivariate polynomials. We illustrate it with a simple example using three attributes A_1, A_2 and A_3. Any policy over this attribute set can be expressed using a polynomial in three variables x, y, z. In the general CNF case, the polynomial can have the terms xyz, xy, xz, yz, x, y, z and a free coefficient. Some terms may have a zero coefficient. For example, consider the CNF formula $(A_1 \vee A_2) \wedge A_3$, which can be expressed as

$$P(x, y, z) = r_1(x-1)(y-1) + r_2(z-1) = r_1xy - r_1x - r_1y + r_2z + (r_1 - r_2)$$

The corresponding ciphertext policy vector is given by

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
x	y	z	xyz	xy	xz	yz	$P(0, \ldots, 0)$
$-r_1$	$-r_1$	r_2	0	r_1	0	0	$r_1 - r_2$

Regular authorities, controlling actual attributes, are responsible for issuing the keys corresponding to the terms x, y and z. In addition, special authorities are responsible for issuing keys corresponding to the cross-terms xyz, xy, xz, yz

and the free coefficient. The authority corresponding to the free coefficient only issues keys for $v_{l+1} = 1$. The trusted authorities, given $\boldsymbol{v}$, enforce that the values requested for the cross-terms are consistent with those requested for x, y and z. The policy above can be written in its DNF form, namely $(A_1 \wedge A_3) \vee (A_2 \wedge A_3)$, which can be expressed as

$$P(x, y, z) = [r_1(x-1) + r_3(z-1)] \cdot [r_2(y-1) + r_3(z-1)]$$

and encoded in a similar manner to the CNF representation.

Note that the encodings for CNF and DNF formulae can be seen as a randomized version of the encodings in [17]. Also, note that the authorities responsible for the cross-terms learn sensitive information about the attributes possessed by the receiver. To verify the values requested for the the cross-term, these authorities need to see the relevant inputs. It is possible to improve receiver privacy using commitments to the input values, and a zero-knowledge proof of the requested value being equal to the output of the corresponding boolean circuit. However, the cross-term value itself reveals considerable information and narrows down the solution space for possibile inputs.

5.2 Security of Randomizing-Polynomial Encodings

Essentially, security of randomizing-polynomial encodings relies on the negligible probability $(\frac{1}{p})$ that the adversary crafts an attribute vector $\boldsymbol{v'}$ that is different from a valid attribute-vector satisfying the policy. With overwhelming probability, this reduces to the security of the underlying basic scheme. The formal proof of security is the same as for conjunctions and HVE, and is given in Appendix C.

6 Related Work

ABE in a multi-authority setting was initially studied by Chase [9], who proposed to to prevent collusion by incorporating a global user identifier into the key-generation procedure. Further improvements were proposed by Müller *et al.* [20] and Chase and Chow [10]. A fully decentralized scheme was proposed by Lewko and Waters [18]. Those constructions do not hide the encryption policy.

Agrawal *et al.* constructed an inner-product PE [2] and a Fuzzy-IBE [1] based on the learning-with-errors assumption (LWE). Lattice-based constructions often naturally hide the encryption policy, and it would be interesting to construct a decentralized scheme, based on LWE. Katz *et al.* introduce a zero-inner-product PE scheme that is fully-hiding [17], meaning the policy remains hidden even for a receiver who can decrypt the ciphertext.

The notion of vector commitments is related to cryptographic accumulators, first introduced by Benaloh and de Mare [4]. Accumulators are compact representations of a set of values, enabling to verify that a given element was used to compute the representation. As an alternative to the VC scheme we used, vector commitments can also be constructed using commitments to polynomials [16], by setting the polynomial coefficients to the vector elements.

Wichs and Zirdelis [29] and Goyal *et al.* [14], independently introduced *Lockable Obfuscation* for Compute-and-Compare programs, based on LWE. A corollary of lockable obfuscation is a transformation of any ABE scheme into one that is weakly attribute-hiding. However, it requires obfuscating a circuit corresponding to the decryption procedure of the underlying ABE scheme. This is highly impractical for the currently known multi-authority ABE schemes, and is not nearly as efficient as our direct construction. However, it is worth mentioning that those constructions theoretically solve the problem of decentralized policy-hiding ABE in a setting where the authorities don't need to know each other at all, and only refer to common public parameters.

Okamoto and Takashima constructed a decentralized ABE scheme, where the authorities do not need to be aware of one-another [22]. Their work claims payload-hiding, but not policy-hiding, and it is left to be studied whether their decentralized scheme can be proven weakly-hiding for the case of inner-product policies. In addition, our scheme enjoys a ciphertext that is at least two times shorter in the number of group elements, and a decryption algorithm that involves only a small constant number of pairings instead of a number proportional to the vector size. Our scheme, however, requires the authorities to publish public keys that are visible to the other authorities, whereas the scheme in [22] does not require any coordination between authorities except for referring to the same public parameters.

7 Future work

Our schemes require coordination between the authorities during setup, namely, advertising public keys to compute the mutually canceling masking terms μ_i. It implies that receivers have to reissue key-requests to all authorities, once the overall set of supported attributes has changed. The desirable level of decentralization is that of the decentralized ABE schemes in [18] and [22], where the authorities do not need to be aware of each other at all. The only link is their referral to common public parameters. Constructing such a *fully-decentralized* ABE scheme that is proven to be policy-hiding remains an open problem.

Another desirable property is *strong attribute-hiding*, meaning that the encryption policy remains hidden even from receivers that are authorized and capable of decrypting the message. Such strong hiding is provided in a single-authority setting by the inner-product predicate encryption scheme of Katz *et al.* [17]. Finally, building on inner-product PE limits us to the types of policies that can be expressed inner-products. Supporting more general policies like linear span programs is highly desirable.

8 Conclusion

We address the problem of decentralized attribute-hiding attribute-based encryption. Starting off the work of Chen *et al.* [11], we constructed a decentralized inner-product predicate encryption scheme. We use it to instantiate a decentralized ABE scheme that hides the ciphertext policy, and show that, in the presence of corrupted authorities, it is not enough to prove security of the underlying PE scheme, but also to properly encode policies. We provide encodings for multiple useful policies. Finally, we propose an extra measure to protect receiver privacy, by using commitments to the attribute vector.

References

1. Agrawal, S., Boyen, X., Vaikuntanathan, V., Voulgaris, P., Wee, H.: Functional encryption for threshold functions (or fuzzy ibe) from lattices. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 280–297. Springer, Heidelberg (May 2012)
2. Agrawal, S., Freeman, D.M., Vaikuntanathan, V.: Functional encryption for inner product predicates from learning with errors. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 21–40. Springer, Heidelberg (Dec 2011)
3. Barthe, G., Danezis, G., Grégoire, B., Kunz, C., Zanella-Beguelin, S.: Verified computational differential privacy with applications to smart metering. In: Computer Security Foundations Symposium (CSF), 2013 IEEE 26th. pp. 287–301. IEEE (2013)
4. Benaloh, J.C., de Mare, M.: One-way accumulators: A decentralized alternative to digital sinatures (extended abstract). In: Helleseth, T. (ed.) EUROCRYPT'93. LNCS, vol. 765, pp. 274–285. Springer, Heidelberg (May 1994)
5. Benhamouda, F., Joye, M., Libert, B.: A new framework for privacy-preserving aggregation of time-series data. ACM Trans. Inf. Syst. Secur. 18(3), 10:1–10:21 (Apr 2016)
6. Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (Mar 2011)
7. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (Feb 2007)
8. Catalano, D., Fiore, D.: Vector commitments and their applications. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 55–72. Springer, Heidelberg (Feb / Mar 2013)
9. Chase, M.: Multi-authority attribute based encryption. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 515–534. Springer, Heidelberg (Feb 2007)
10. Chase, M., Chow, S.S.M.: Improving privacy and security in multi-authority attribute-based encryption. In: Al-Shaer, E., Jha, S., Keromytis, A.D. (eds.) ACM CCS 09. pp. 121–130. ACM Press (Nov 2009)
11. Chen, J., Gay, R., Wee, H.: Improved dual system ABE in prime-order groups via predicate encodings. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 595–624. Springer, Heidelberg (Apr 2015)
12. Chen, J., Wee, H.: Dual system groups and its applications — compact HIBE and more. Cryptology ePrint Archive, Report 2014/265 (2014), http://eprint.iacr.org/2014/265
13. Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.L.: An algebraic framework for Diffie-Hellman assumptions. Journal of Cryptology 30(1), 242–288 (Jan 2017)

14. Goyal, R., Koppula, V., Waters, B.: Lockable obfuscation. In: 58th FOCS. pp. 612–621. IEEE Computer Society Press (2017)
15. Halevi, S.: A sufficient condition for key-privacy. Cryptology ePrint Archive, Report 2005/005 (2005), http://eprint.iacr.org/2005/005
16. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 177–194. Springer, Heidelberg (Dec 2010)
17. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. Journal of Cryptology 26(2), 191–224 (Apr 2013)
18. Lewko, A.B., Waters, B.: Decentralizing attribute-based encryption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 568–588. Springer, Heidelberg (May 2011)
19. Libert, B., Yung, M.: Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In: Theory of Cryptography Conference. pp. 499–517. Springer (2010)
20. Müller, S., Katzenbeisser, S., Eckert, C.: Distributed attribute-based encryption. In: Lee, P.J., Cheon, J.H. (eds.) ICISC 08. LNCS, vol. 5461, pp. 20–36. Springer, Heidelberg (Dec 2009)
21. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Consulted pp. 1–9 (2008), http://s.kwma.kr/pdf/Bitcoin/bitcoin.pdf
22. Okamoto, T., Takashima, K.: Decentralized attribute-based signatures. Cryptology ePrint Archive, Report 2011/701 (2011), http://eprint.iacr.org/2011/701
23. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO'91. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (Aug 1992)
24. Rahulamathavan, Y., Phan, R.C.W., Rajarajan, M., Misra, S., Kondoz, A.: Privacy-preserving blockchain based IoT ecosystem using attribute-based encryption. In: IEEE International Conference on Advanced Networks and Telecommunications Systems. Odisha, India (Dec 2017)
25. Roberts, F.: UK/India consortium explore blockchain for healthcare IoT security. https://internetofbusiness.com/consortium-blockchain-iot-security/
26. Sahai, A., Waters, B.: Slides on functional encryption. PowerPoint presentation (2008), http://www.cs.utexas.edu/~bwaters/presentations/files/functional.ppt
27. Sahai, A., Waters, B.R.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (May 2005)
28. Shi, E., Chan, T.H.H., Rieffel, E.G., Chow, R., Song, D.: Privacy-preserving aggregation of time-series data. In: NDSS 2011. The Internet Society (Feb 2011)
29. Wichs, D., Zirdelis, G.: Obfuscating compute-and-compare programs under LWE. In: 58th FOCS. pp. 600–611. IEEE Computer Society Press (2017)

A Vector Commitments

A vector commitment scheme consists of the following algorithms:

- $\mathsf{VC.Setup}(1^\lambda, n)$: On input of a security parameter λ and the vector size n, output the public parameters pp which implicitly define the message space $\mathcal{M}$.

- $\mathsf{VC.Commit}_{\mathsf{pp}}(\boldsymbol{v})$: On input the public parameters and a sequence of values $\boldsymbol{v} = (v_1, \ldots, v_n)$, where $v_1, \ldots, v_n \in \mathcal{M}$, output a commitment string C and auxiliary information $\mathtt{aux}$. The auxiliary information is simply a vector of the underlying hiding per-element commitments.
- $\mathsf{VC.Open}_{\mathsf{pp}}(v, i, \mathtt{aux})$: Run by the committer (that requests the keys in our setting) to produce a proof P_i that v is the i-th committed value.
- $\mathsf{VC.Verify}_{\mathsf{pp}}(C, v, i, P_i)$: Outputs *true* only if P_i is a valid proof that C was created to a sequence of values $v_1, \ldots, v_n$ s.t. $v_i = v$.

Definition 5 (Position-Binding). *A vector commitment scheme* VC *is position-binding if* $\forall i = 1, \ldots, n$ *and for every efficient adversary* $\mathcal{A}$, $\mathcal{A}$ *has negligible probability of producing a tuple* (C, v, v', i, P, P') *where* $v \neq v'$ *s.t.*

$$\mathsf{VC.Verify}_{\mathsf{pp}}(C, v, i, P) \ \wedge \ \mathsf{VC.Verify}_{\mathsf{pp}}(C, v', i, P')$$

While position-binding ensures that only key-parts issued for the same vector $\boldsymbol{v}$ can be combined to obtain a functional decryption key, we require the vector-commitment scheme to be hiding. Note that the vector commitment scheme is hiding only when composed with a standard commitment scheme to generate a hiding commitment for each element, that are in turn input to the vector commitment scheme. As suggested in [8], we combine the generic VC scheme with a hiding commitment scheme. Depending on whether valid attribute vectors contain only 0 and 1 or arbitrary values, we can use Pedersen's one-bit commitment scheme [23] or generalized Pedersen commitments.

B Proof of Security for the Construction in Section 3.2

We prove that the scheme in section 3 is *weakly attribute-hiding*, according to Definition 2, under Assumption 2.

Proof intuition. We define a sequence of games, starting with the actual scheme and ending up with a challenge ciphertext that encodes a random message using a random predicate vector. We argue that the games are indistinguishable to the adversary, concluding that the scheme is attribute-hiding. The transition between the hybrids is based on switching to "semi-functional" keys, and a "semi-functional" challenge ciphertext that cannot be decrypted using those keys, and looks uniformly random to the adversary.

A semi-functional ciphertext is one where instead of $A\boldsymbol{s}$ in the exponent, we have a random vector $\boldsymbol{z} \in \mathbb{Z}_p^{k+1}$. A semi-functional key is one where $\boldsymbol{\alpha}$ is replaced by $\boldsymbol{\alpha} + \boldsymbol{a}^{\perp}\hat{t}$ ($\hat{t} \in \mathbb{Z}_p$ and $\boldsymbol{h} = \mathbf{B}\boldsymbol{r}$, where $\boldsymbol{r} \in \mathbb{Z}_p^{k}$). It is easy to check that a semi-functional key cannot decrypt a semi-functional ciphertext. That, in turn, enables us to switch from a game where an actual message m is encrypted under an actual policy vector $\boldsymbol{x}$, to one where a random message m' is encrypted under a random policy vector $\boldsymbol{x}^{\star}$. This sequence is similar to the one in the proof of the weak attribute-hiding scheme in [11], with certain additions, modifications and reordering of games.

For simplicity, we use the fact that the terms $g_2{}^{\boldsymbol{\mu_i}}$ are random in $(\mathbb{G}_2)^n$, in the adversarial view, as long as there are at least 2 honest authorities, unless canceled by summation in the exponent. In fact, the key combining is similar to the technique for privacy-preserving aggregation, as proposed by Shi *et al.* [28]. We can therefore refer to their security proof to justify this step. Barthe *et al.* [3] also used a similar technique in their privacy-preserving aggregation protocol. A tighter security reduction, linear in the number of adversarial queries, can be achieved using Smooth Projective Hash Functions (SPHF), as in [5].

As a result, K_i are only useful as a product $K = \prod_{i=1}^{n} K_i$. Therefore, we prove the security of a scheme where the challenger computes K directly and hands it to the adversary, since it can always split it to n random shares in $\mathbb{G}_2$. It is easy to show that if there is an efficient adversary $\mathcal{A}$ that wins the game in Definition 2 against $\mathcal{S}'$, $\mathcal{S}'$ can use it to win the modified game against $\mathcal{S}$. When $\mathcal{A}$ requests a key for $(\mathsf{GID}, i, \boldsymbol{v})$, $\mathcal{S}'$ asks $\mathcal{S}$ for a key for the whole vector $\boldsymbol{v}$, splits it to random multiplicative shares, and serves the correct share to $\mathcal{A}$. It can later use the other shares for subsequent requests corresponding to the same GID and $\boldsymbol{v}$. It is therefore enough to prove security of this modified game.

Notation. Throughout the proof, we denote $\boldsymbol{\alpha} = \sum_{i=1}^{n} \boldsymbol{\alpha}_i$. When $a^\perp$ and $b^\perp$ are used, they are assumed to be chosen such that $A^T a^\perp = 0$ and $B^T b^\perp = 0$.

Game Sequence. We outline the sequence of games:

- Game_0 is the real security game as in Definition 2.
- Game_1 is the same as Game_0, except that the random oracle queries are answered in the following way: the challenger picks a random $\boldsymbol{r} \in \mathbb{Z}_p^k$ and computes $\boldsymbol{h} = B\boldsymbol{r}$. It stores the value $g_2^{\boldsymbol{h}}$ to answer future queries with the same input, and returns it to the adversary.
- $\mathsf{Game}_{2,j,1}$ is the same as Game_1, except that the challenger samples $A, a^\perp$ s.t. $A^T a^\perp = 0$, $B, b^\perp$ s.t. $B^T b^\perp = 0$ and a random $\hat{t} \in \mathbb{Z}_p$, and the key queries are answered as follows:
  - The first $j-1$ keys are given by
    $$K = g_2^{\boldsymbol{\alpha} + a^\perp \hat{t} - \sum_{i=1}^{n} v_i W_i \boldsymbol{h}}$$
    where $\boldsymbol{h}$ is constructed as in Game_1, i.e. by picking a random $\boldsymbol{r}$ and computing $\boldsymbol{h} = B\boldsymbol{r}$.
  - The j-th key is given by
    $$K = g_2^{\boldsymbol{\alpha} - \sum_{i=1}^{n} v_i W_i \boldsymbol{h}}$$
    where $\boldsymbol{h}$ is constructed by sampling random $\boldsymbol{r} \in \mathbb{Z}_p^k$ and $\hat{r} \in \mathbb{Z}_p$, and computing $\boldsymbol{h} = B\boldsymbol{r} + a^\perp \hat{r}$.
  - The last $q-j$ keys are constructed as
    $$K = g_2^{\boldsymbol{\alpha} - \sum_{i=1}^{n} v_i W_i \boldsymbol{h}}$$
    where $\boldsymbol{h}$ is in the span of B, i.e. $\boldsymbol{h} = B\boldsymbol{r}$ for some random $\boldsymbol{r} \in \mathbb{Z}_p^k$.

Note that adversarial random oracle queries are answered in the same manner. If the adversary queries for a given input $\boldsymbol{v}$, $\boldsymbol{h}$ is constructed according to the current query counter. When a key for the same input is requested, the challenger will reuse the stored value of $\boldsymbol{h}$, and in addition use $\boldsymbol{\alpha}$ or $\boldsymbol{\alpha} + a^{\perp}\hat{t}$ according to the query counter value at the time of querying the random oracle.

- $\mathsf{Game}_{2,j,2}$ is the same as $\mathsf{Game}_{2,j,1}$, except that the j-th key query is answered by picking random $\boldsymbol{r} \in \mathbb{Z}_p^k$ and $\hat{r} \in \mathbb{Z}_p$ and outputting

$$K = g_2^{\boldsymbol{\alpha} + a^{\perp}\hat{t} - \sum_{i=1}^{n} v_i W_i \boldsymbol{h}}$$

  where $\boldsymbol{h} = B\boldsymbol{r} + a^{\perp}\hat{r}$.
- $\mathsf{Game}_{2,j,3}$ is the same as $\mathsf{Game}_{2,j,2}$, except that the first j key queries are answered by picking random $\boldsymbol{r} \in \mathbb{Z}_p^k$ and outputting

$$K = g_2^{\boldsymbol{\alpha} + a^{\perp}\hat{t} - \sum_{i=1}^{n} v_i W_i \boldsymbol{h}}$$

  where $\boldsymbol{h} = B\boldsymbol{r}$. Note that $\mathsf{Game}_{2,0,3}$ is exactly the same as Game_1.
- Game_3 is the same as Game_2, except that the challenge ciphertext is semi–functional. It is constructed by picking a random $\boldsymbol{z} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$ and generating

$$C_0 = g_1^{\boldsymbol{z}} \qquad C_{1,i} = g_1^{(x_{b,i}U^T + W_i^T)\boldsymbol{z}} \qquad C' = m \cdot \hat{e}(g_1, g_2)^{\boldsymbol{\alpha}^T \boldsymbol{z}}$$

  where $\boldsymbol{\alpha} = \sum_{i=1}^{n} \boldsymbol{\alpha}_i$.
- Game_4 is the same as Game_3 except that the challenge ciphertext is a semi–functional encryption of a random message in $\mathbb{G}_T$.
- Game_5 is the same as Game_4, except that the predicate vector $\boldsymbol{x}_b$ is replaced by a random vector $\boldsymbol{x}^{\star} \xleftarrow{\$} \mathbb{Z}_p^n$.

In Game_5 the adversarial view is statistically independent of the challenger's choice of b, and the adversary has no advantage in winning the game.

Lemmas about indistinguishability between the subsequent games and their proofs are provided below.

The condition that the adversary cannot ask for keys corresponding to $\boldsymbol{v}$ s.t. $\langle \boldsymbol{x}_b, \boldsymbol{v} \rangle = 0$ is necessary for the transition to games $2, j, 1$ and on. The semi-functional keys can decrypt any self-produced ciphertext. However, they cannot be successfully applied to the semi-functional challenge ciphertext. Had the adversary been allowed to ask for a key corresponding to a set of attributes that satisfy the policy, it would have been able to distinguish the two games, by the success or failure of the challenge decryption.

Key-Privacy. Key-privacy in Game_4 can also be argued based on Halevi's sufficient condition for key-privacy [15]. It states that data privacy provides key privacy if the distribution of a random encryption of a random message is independent of the public key. While in Game_0 (the actual scheme) the encryption is not independent of the public-key $g_1^A, \{g_1^{W_i^T A}\}_{i=1}^n$, the semi-functional challenge ciphertexts in subsequent games are independent of it, since a random $\boldsymbol{z}$ is used instead of $A\boldsymbol{s}$. Hence, key-privacy follows readily from Halevi's condition being satisfied by the semi-functional encryption of a random message in Game_4.

In the following we provide proofs of indistinguishability between the subsequent games defined previously.

Lemma 1 (Indistinguishability of Game_0 and Game_1). *For any adversary $\mathcal{A}$ that distinguishes between the two games with a non-negligible advantage, there exists $\mathcal{S}$ that has a non-negligible advantage in solving the k-linear decision problem in $\mathbb{G}_2$, i.e.*

$$|Adv_0(\lambda) - Adv_1(\lambda)| \leq Adv_{\mathcal{S}}^{k-\mathsf{Lin}}(\lambda)$$

Proof. $\mathcal{S}$ receives an instance of a k-linear decision problem g_2^B and $g_2^{\boldsymbol{h}}$, where $\boldsymbol{h}$ is either $B\boldsymbol{r}$ for some random $\boldsymbol{r} \in \mathbb{Z}_p^k$ or random in $\mathbb{Z}_p^{k+1}$. The public parameters, authority public keys and challenge ciphertext are constructed as in the actual scheme. The key queries are answered by computing

$$K = g_2^{\boldsymbol{\alpha}} \cdot \prod_{i=1}^{n} (g_2^{\boldsymbol{h}})^{-v_i W_i} = g_2^{\boldsymbol{\alpha} - \sum_{i=1}^{n} v_i W_i \boldsymbol{h}}$$

When $\boldsymbol{h} \in span(B)$ the keys and random oracle responses are distributed exactly as in Game_1, whereas in the other case the distribution is exactly as in Game_0. By the k-linear assumption, $\mathcal{A}$ has only negligible advantage in distinguishing between the two.

Lemma 2 (Indistinguishability of $\mathsf{Game}_{2,j-1,3}$ and $\mathsf{Game}_{2,j,1}$). *For $j = 1,..,q$, for any adversary $\mathcal{A}$ that makes at most q key queries, and distinguishes between the two games with a non-negligible advantage, there exists $\mathcal{S}$ that has a non-negligible advantage in solving the k-linear decision problem in $\mathbb{G}_2$, i.e.*

$$|Adv_{2,j,1}(\lambda) - Adv_{2,j-1,3}(\lambda)| \leq Adv_{\mathcal{S}}^{k-\mathsf{Lin}}(\lambda)$$

Proof. $\mathcal{S}$ receives an instance of a k-linear decision problem g_2^B and $g_2^{\boldsymbol{h}}$ where $\boldsymbol{h}$ is either equal to $B\boldsymbol{t}$ for some $\boldsymbol{t} \in \mathbb{Z}_p^k$ or uniform in $\mathbb{Z}_p^{k+1}$.

Setup. $\mathcal{S}$ picks random $\boldsymbol{\alpha}_i \in \mathbb{Z}_p^{k+1}$, a random $\hat{t} \in \mathbb{Z}_p$, random $A, a^{\perp}, W_i, U$ as in the actual scheme, and publishes the public parameters $g_1^A, g_1^{U^T A}$ and the authority public keys $g_1^{W_i^T A}$ and $\hat{e}(g_1, g_2)^{\boldsymbol{\alpha}_i^T A}$.

Key Queries. On input of the m-th key query, $\mathcal{S}$ outputs

$$K = \begin{cases} g_2^{\boldsymbol{\alpha} + a^{\perp}\hat{t} - \sum_{i=1}^{n} v_i W_i B\boldsymbol{r}} & m < j \\ g_2^{\boldsymbol{\alpha} - \sum_{i=1}^{n} v_i W_i \boldsymbol{h}} & m = j \\ g_2^{\boldsymbol{\alpha} - \sum_{i=1}^{n} v_i W_i B\boldsymbol{r}} & m > j \end{cases}$$

where $\boldsymbol{r} \xleftarrow{\$} \mathbb{Z}_p^k$.

Ciphertext. The challenge ciphertext is constructed by picking a random $\boldsymbol{s} \in \mathbb{Z}_p^s$ and computing

$$C_0 = g_1^{A\boldsymbol{s}} \qquad C_{1,i} = g_1^{(x_{b,i}U^T + W_i^T)A\boldsymbol{s}} \qquad C' = m_b \cdot \hat{e}(g_1, g_2)^{\boldsymbol{\alpha}^T A\boldsymbol{s}}$$

When $\boldsymbol{h} \in span(B)$ the keys are distributed exactly as in $\mathsf{Game}_{2,j-1,3}$, whereas in the other case they are distributed as in $\mathsf{Game}_{2,j,1}$, By the k-linear assumption $\mathcal{A}$ has only non-negligible advantage in distinguishing between the two cases.

Lemma 3 (Indistinguishability of $\mathsf{Game}_{2,j,1}$ and $\mathsf{Game}_{2,j,2}$). *For $j = 1, .., q$, any adversary $\mathcal{A}$ that makes at most q key queries, has no advantage in distinguishing between the two games, i.e.*

$$|Adv_{2,j,1}(\lambda) - Adv_{2,j,2}(\lambda)| = 0$$

Proof. [4] The difference between the two games is that in the latter we multiply the j-th key by $g_2^{a^\perp \hat{t}}$ where $\hat{t}$ is sampled randomly from $\mathbb{Z}_p$ by $\mathcal{S}$. On the j-th query $\mathcal{S}$ is given the vector $\boldsymbol{v}$ for which the adversary requests a key, and it performs the following change of variables:

$$W_i' = \begin{cases} W_i + \left(v_i {a^\perp}^T b^\perp\right)^{-1} a^\perp {b^\perp}^T & v_i \neq 0 \\ W_i & v_i = 0 \end{cases}$$

The distributions of W_i and W_i' are clearly identical. We can see that had the ciphertexts and authorities' public keys been generated using W_i' instead of W_i, there would be no difference: for the public keys we have

$$g_1^{W_i'^T A} = g_1^{W_i^T A + (v_i {a^\perp}^T b^\perp)^{-1} (a^\perp {b^\perp}^T)^T A} = g_1^{W_i^T A}$$

and for the ciphertext

$$C_{1,i} = g_1^{(x_i U^T + W_i'^T)A\boldsymbol{s}} = g_1^{(x_i U^T + W_i^T)A\boldsymbol{s} + (v_i {a^\perp}^T b^\perp)^{-1} (a^\perp {b^\perp}^T)^T A\boldsymbol{s}} = g_1^{(x_i U^T + W_i^T)A\boldsymbol{s}}$$

The j-th key query is answered using W_i' as follows

$$K = g_2^{\boldsymbol{\alpha} - \sum_{i=1}^n v_i W_i' \boldsymbol{h}}$$

where $\boldsymbol{h} = B\boldsymbol{r} + \frac{\hat{t}}{n} a^\perp$ for some $\boldsymbol{r} \in \mathbb{Z}_p^k$ and $\hat{t} \in \mathbb{Z}_p$. We have

$$\begin{aligned} W_i' \boldsymbol{h} &= W_i \boldsymbol{h} + (v_i {a^\perp}^T b^\perp)^{-1} (a^\perp {b^\perp}^T)(B\boldsymbol{r} + \frac{\hat{t}}{n} a^\perp) \\ &= W_i \boldsymbol{h} + v_i^{-1} ({a^\perp}^T b^\perp)^{-1} (a^\perp {b^\perp}^T) a^\perp \frac{\hat{t}}{n} \\ &= W_i \boldsymbol{h} + v_i^{-1} ({a^\perp}^T b^\perp)^{-1} a^\perp ({b^\perp}^T a^\perp) \frac{\hat{t}}{n} = W_i \boldsymbol{h} + v_i^{-1} a^\perp \frac{\hat{t}}{n} \end{aligned}$$

[4] The proof technique of this lemma draws on the substitution technique used in the proof of lemma 7 in [11].

and the key is therefore given by

$$K = g_2^{\boldsymbol{\alpha} - \sum_{i=1}^{n} (v_i W_i \boldsymbol{h} + a^{\perp} \frac{\hat{t}}{n})} = g_2^{\boldsymbol{\alpha} + a^{\perp} \hat{t} - \sum_{i=1}^{n} v_i W_i \boldsymbol{h}}$$

and we obtain the exact distribution of $\mathsf{Game}_{2,j,2}$.

Lemma 4 (Indistinguishability of $\mathsf{Game}_{2,j,2}$ and $\mathsf{Game}_{2,j,3}$). *For $j = 1, .., q$, for any adversary $\mathcal{A}$ that makes at most q key queries, and distinguishes between the two games with a non-negligible advantage, there exists $\mathcal{S}$ that has a non-negligible advantage in solving the k-linear decision problem in $\mathbb{G}_2$, i.e.*

$$|Adv_{2,j,2}(\lambda) - Adv_{2,j,3}(\lambda)| \leq Adv_{\mathcal{S}}^{k-\mathsf{Lin}}(\lambda)$$

Proof. $\mathcal{S}$ receives an instance of a k-linear decision problem g_2^B and $g_2^{\boldsymbol{h}}$ where $\boldsymbol{h}$ is either equal $B\boldsymbol{t}$ for some $\boldsymbol{t} \in \mathbb{Z}_p^k$ or uniform in $\mathbb{Z}_p^{k+1}$.

Setup. $\mathcal{S}$ picks a random $\boldsymbol{\alpha}_i \in \mathbb{Z}_p^{k+1}$, a random $\hat{t} \in \mathbb{Z}_p$, random $A, a^{\perp}, W_i, U$ as in the actual scheme, and publishes the public parameters $g_1^A, g_1^{U^T A}$ and the authority public keys $g_1^{W_i^T A}$ and $\hat{e}(g_1, g_2)^{\boldsymbol{\alpha}_i^T A}$.

Key Queries. On input the m-th key query, $\mathcal{S}$ outputs

$$K = \begin{cases} g_2^{\boldsymbol{\alpha} + a^{\perp} \hat{t} - \sum_{i=1}^{n} v_i W_i \boldsymbol{h}} & m \leq j \\ g_2^{\boldsymbol{\alpha} - \sum_{i=1}^{n} v_i W_i B \boldsymbol{r}} & m > j \end{cases}$$

where $\boldsymbol{r} \xleftarrow{\$} \mathbb{Z}_p^k$.

Ciphertext. The challenge ciphertext is constructed by picking a random $\boldsymbol{s} \in \mathbb{Z}_p^k$ and computing

$$C_0 = g_1^{A\boldsymbol{s}} \qquad C_{1,i} = g_1^{(x_{b,i} U^T + W_i^T) A \boldsymbol{s}} \qquad C' = m_b \cdot \hat{e}(g_1, g_2)^{\boldsymbol{\alpha}^T A \boldsymbol{s}}$$

When $\boldsymbol{h} \in span(B)$ the keys are distributed exactly as in $\mathsf{Game}_{2,j,3}$, whereas in the other case they are distributed as in $\mathsf{Game}_{2,j,2}$, By the k-linear assumption $\mathcal{A}$ has only non-negligible advantage in distinguishing between the two cases.

Lemma 5 (Indistinguishability of $\mathsf{Game}_{2,q,3}$ and Game_3). *For any adversary $\mathcal{A}$ that distinguishes between the two games with a non-negligible advantage, there exists $\mathcal{S}$ that has a non-negligible advantage in solving the k-linear decision problem in $\mathbb{G}_1$, i.e.*

$$|Adv_{2,q,3}(\lambda) - Adv_3(\lambda)| \leq Adv_{\mathcal{S}}^{k-\mathsf{Lin}}(\lambda)$$

Proof. $\mathcal{S}$ receives a k-linear challenge consisting of g_1^A and $g_1^{\boldsymbol{z}}$, where $\boldsymbol{z} = A\boldsymbol{s}$ for some random $\boldsymbol{s} \in \mathbb{Z}_p^k$, or uniform in $\mathbb{Z}_p^{k+1}$. $\mathcal{S}$ samples random matrices U

and $\{W_i\}_{i=1}^n$, and vectors $\boldsymbol{\alpha}_i$ as in the actual scheme, and outputs the public parameters $g_1^A, g_1^{U^T A}$ and authority public keys $g_1^{W_i^T A}$ and

$$\hat{e}(g_1^A, g_2^{\boldsymbol{\alpha}_i}) = \hat{e}(g_1, g_2)^{\boldsymbol{\alpha}_i^T A}$$

The challenge ciphertext is given by

$$C_0 = g_1^{\boldsymbol{z}} \qquad C_{1,i} = g_1^{(x_{b,i} U^T + W_i^T)\boldsymbol{z}} \qquad C' = m_b \cdot \hat{e}(g_1, g_2)^{\boldsymbol{\alpha}^T \boldsymbol{z}}$$

When $\boldsymbol{z} \in span(A)$ the challenge is distributed exactly as in Game_1, whereas in the other case the distribution is as in Game_2. The keys are constructed as in Game_1. By the k-linear assumption $\mathcal{A}$ has only negligible advantage in distinguishing between the two. Note that $\mathcal{S}$ knows neither A nor $a^\perp$, however it doesn't need this knowledge to simulate the games.

Lemma 6 (Indistinguishability of Game_3 and Game_4). *The two games are statistically close and the adversary has no advantage in distinguishing between the two, i.e.*

$$|Adv_3(\lambda) - Adv_4(\lambda)| = 0$$

Proof. $\mathcal{S}$ samples $A, a^\perp$ s.t. $A^T a^\perp = 0$ and a random vector $\boldsymbol{\alpha} \in \mathbb{Z}_p^{k+1}$ and a random $\hat{t} \in \mathbb{Z}_p$. $\mathcal{S}$ computes $\hat{\alpha} = \boldsymbol{\alpha} - a^\perp \hat{t}$.

Setup. The public parameters are given by $g_1^A, g_1^{U^T A}$ and the authority public keys by $g_1^{W_i^T A}$ and $\hat{e}(g_1, g_2)^{\hat{\alpha}_i^T A} = \hat{e}(g_1, g_2)^{\boldsymbol{\alpha}_i^T A}$.

Key Queries. These are answered with normal keys constructed using $\hat{\alpha}$:

$$K = g_2^{\hat{\alpha} - \sum_{i=1}^n v_i W_i \boldsymbol{h}}$$

Ciphertext. $\mathcal{S}$ samples $\boldsymbol{s} \in \mathbb{Z}_p^k$ and $\hat{s} \in \mathbb{Z}_p$ and sets $\boldsymbol{z} = A\boldsymbol{s} + b^\perp \hat{s}$. It computes

$$m' = m_b \cdot \hat{e}(g_1, g_2)^{(a^\perp \hat{t})^T \boldsymbol{z}} = m_b \cdot \hat{e}(g_1, g_2)^{\hat{t}\hat{s}(a^\perp)^T b^\perp}$$

which is a random element in $\mathbb{G}_T$ given that $(a^\perp)^T b^\perp \neq 0$ with high probability, and random $\hat{t}, \hat{s}$. The ciphertext is constructed as

$$\begin{aligned} C_0 &= g_1^{\boldsymbol{z}} \qquad C_{1,i} = g_1^{(x_i U^T + W_i^T)\boldsymbol{z}} \\ C' &= m' \cdot \hat{e}(g_1, g_2)^{\hat{\alpha}^T \boldsymbol{z}} = m_b \cdot \hat{e}(g_1, g_2)^{\hat{t}\hat{s}(a^\perp)^T b^\perp} \cdot \hat{e}(g_1, g_2)^{\boldsymbol{\alpha}^T \boldsymbol{z} - \hat{t}(a^\perp)^T b^\perp \hat{s}} \\ &= m_b \cdot \hat{e}(g_1, g_2)^{\boldsymbol{\alpha}^T \boldsymbol{z}} \end{aligned}$$

We can see that this game is distributed exactly as Game_3, and therefore there is no advantage for the adversary in distinguishing between the two.

Lemma 7 (Indistinguishability of Game_4 and Game_5). *The two games are statistically close and the adversary has no advantage in distinguishing between the two, i.e.*

$$|Adv_4(\lambda) - Adv_5(\lambda)| = 0$$

Proof. $C_{1,i}$ is given by

$$C_{1,i} = g_1^{(x_{b,i}U^T + W_i^T)(A\boldsymbol{s}+b^\perp \hat{s})} = g_1^{x_{b,i}U^T(A\boldsymbol{s}+b^\perp \hat{s})} \cdot g_1^{W_i^T(A\boldsymbol{s}+b^\perp \hat{s})}$$

for some A, $\boldsymbol{s} \in \mathbb{Z}_p^k$, $b^\perp \in \mathbb{Z}_p^k$ s.t. $B^T b^\perp = 0$ and $\hat{s} \in \mathbb{Z}_p$. The term $g_1^{W_i^T(A\boldsymbol{s}+b^\perp \hat{s})}$ is uniform in $\mathbb{G}_1$ given $g_1^A, g_1^{A\boldsymbol{s}+b^\perp \hat{s}}, \{g_1^{W_i^T A}\}_{i=1}^n$ and g_2^B. Therefore $C_{1,i}$ is uniform in $\mathbb{G}_1$ and the adversary has no advantage in identifying the case where the predicate vector is replaced by a random vector $\boldsymbol{x}^\star$.

C Proof of Security for Randomized Encodings

The challenger generates masking terms μ_i for good authorities, as specified in the construction. Therefore, as long as there is more than one good authority, the key parts corresponding to them are random in the view of the adversary, unless canceled by multiplication with $\{g_2^{\boldsymbol{\mu}_i} : \forall i \in \mathcal{A}^\star\}$. Therefore, the challenger only has to simulate $K_{\mathsf{good}} = \prod_{i \notin \mathcal{A}^\star} K_i$.

We denote $\boldsymbol{\alpha}_{\mathsf{good}} = \sum_{i \notin \mathcal{A}^\star} \boldsymbol{\alpha}_i$. We then have $\boldsymbol{\alpha} = \boldsymbol{\alpha}_{\mathsf{good}} + \sum_{i \in \mathcal{A}^\star} \boldsymbol{\alpha}_i$. The proof consists of a sequence of games, similar to that in appendix B, except that $\mathcal{S}$ generates K_{good} (and $\boldsymbol{\alpha}_{\mathsf{good}}$) instead of K (and $\boldsymbol{\alpha}$).

Game Sequence. We outline the sequence of games:

- Game_0 is the real security game as in section 3.
- Game_1 is the same as Game_0, except that the random oracle queries are answered in the following way: the challenger picks a random $\boldsymbol{r} \in \mathbb{Z}_p^k$ and computes $\boldsymbol{h} = B\boldsymbol{r}$. It stores the value $g_2^{\boldsymbol{h}}$ to answer future queries with the same input, and returns it to the adversary.
- $\mathsf{Game}_{2,j,1}$ is the same as Game_1, except that the challenger samples $A, a^\perp$ s.t. $A^T a^\perp = 0$, $B, b^\perp$ s.t. $B^T b^\perp = 0$ and a random $\hat{t} \in \mathbb{Z}_p$, and the key queries are answered as follows:
  - The first $j-1$ keys are given by

    $$K_{\mathsf{good}} = g_2^{\boldsymbol{\alpha}_{\mathsf{good}} + a^\perp \hat{t} - \sum_{i \notin \mathcal{A}^\star}(v_i W_i \boldsymbol{h} + \boldsymbol{\mu}_i)}$$

    where $\boldsymbol{h}$ is constructed as in Game_1, i.e. by picking a random $\boldsymbol{r}$ and computing $\boldsymbol{h} = B\boldsymbol{r}$.
  - The j-th key is given by

    $$K_{\mathsf{good}} = g_2^{\boldsymbol{\alpha}_{\mathsf{good}} - \sum_{i \notin \mathcal{A}^\star}(v_i W_i \boldsymbol{h} + \boldsymbol{\mu}_i)}$$

    where $\boldsymbol{h}$ is constructed by sampling random $\boldsymbol{r} \in \mathbb{Z}_p^k$ and $\hat{r} \in \mathbb{Z}_p$, and computing $\boldsymbol{h} = B\boldsymbol{r} + a^\perp \hat{r}$.
  - The last $q-j$ keys are constructed as

    $$K_{\mathsf{good}} = g_2^{\boldsymbol{\alpha}_{\mathsf{good}} - \sum_{i \notin \mathcal{A}^\star}(v_i W_i \boldsymbol{h} + \boldsymbol{\mu}_i)}$$

    where $\boldsymbol{h}$ is in the span of B, i.e. $\boldsymbol{h} = B\boldsymbol{r}$ for some random $\boldsymbol{r} \in \mathbb{Z}_p^k$.

- $\mathsf{Game}_{2,j,2}$ is the same as $\mathsf{Game}_{2,j,1}$, except that the j-th key query is answered by picking random $\boldsymbol{r} \in \mathbb{Z}_p^k$ and $\hat{r} \in \mathbb{Z}_p$ and outputting

$$K_{\mathsf{good}} = g_2^{\boldsymbol{\alpha}_{\mathsf{good}} + a^{\perp}\hat{t} - \sum_{i \notin \mathcal{A}^\star}(v_i W_i \boldsymbol{h} + \boldsymbol{\mu}_i)}$$

  where $\boldsymbol{h} = B\boldsymbol{r} + a^{\perp}\hat{r}$.
- $\mathsf{Game}_{2,j,3}$ is the same as $\mathsf{Game}_{2,j,2}$, except that the first j key queries are answered by picking random $\boldsymbol{r} \in \mathbb{Z}_p^k$ and outputting

$$K_{\mathsf{good}} = g_2^{\boldsymbol{\alpha}_{\mathsf{good}} + a^{\perp}\hat{t} - \sum_{i \notin \mathcal{A}^\star}(v_i W_i \boldsymbol{h} + \boldsymbol{\mu}_i)}$$

  where $\boldsymbol{h} = B\boldsymbol{r}$. Note that $\mathsf{Game}_{2,0,3}$ is exactly the same as Game_1.
- Game_3 is the same as $\mathsf{Game}_{2,q,3}$, except that the challenge ciphertext is semi–functional. It is constructed by picking a random $\boldsymbol{z} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$ and generating

$$C_0 = g_1^{\boldsymbol{z}} \qquad C_{1,i} = g_1^{(x_{b,i}U^T + W_i^T)\boldsymbol{z}} \qquad C' = m \cdot \hat{e}(g_1, g_2)^{\boldsymbol{\alpha}^T \boldsymbol{z}}$$

  where $\boldsymbol{\alpha} = \sum_{i=1}^n \boldsymbol{\alpha}_i$.
- Game_4 is the same as Game_3 except that the challenge ciphertext is a semi–functional encryption of a random message in $\mathbb{G}_T$.
- Game_5 is the same as Game_4, except that the predicate vector $\boldsymbol{x}_b$ is replaced by a vector $\boldsymbol{x}^\star$ such that $x_i \xleftarrow{\$} \mathbb{Z}_p : \forall i \notin \mathcal{A}^\star$, i.e. it has random values in entries corresponding to good authorities.

In Game_5 the adversarial view is statistically independent of the challenger's choice of b, and the adversary has no advantage in winning the game.

The proofs of indistinguishability between Game_0 and Game_1, $\mathsf{Game}_{2,j-1,3}$ and $\mathsf{Game}_{2,j,1}$, $\mathsf{Game}_{2,j,1}$ and $\mathsf{Game}_{2,j,2}$, $\mathsf{Game}_{2,j,2}$ and $\mathsf{Game}_{2,j,3}$, Game_3 and Game_4, and between Game_4 and Game_5, are the same as in appendix B.

Lemma 8 (Indistinguishability of $\mathsf{Game}_{2,j-1,3}$ and $\mathsf{Game}_{2,j,1}$). *For any PPT adversary $\mathcal{A}$ that distinguishes between the two games with a non-negligible advantage, there exists $\mathcal{S}$ that has a non negligible advantage in solving the k-linear decision problem in $\mathbb{G}_2$, i.e.*

$$|Adv_{2,j-1,3}(\lambda) - Adv_{2,j,1}(\lambda)| \leq Adv_{\mathcal{S}}^{k-\mathsf{Lin}}(\lambda) + \frac{d}{p}$$

where d is the number of decryption attempts made by the adversary.

Lemma 9 (Indistinguishability of $\mathsf{Game}_{2,j,1}$ and $\mathsf{Game}_{2,j,2}$). *Any PPT adversary $\mathcal{A}$ that makes at most q key queries, has only negligible advantage in distinguishing between the two games, i.e.*

$$|Adv_{2,j,2}(\lambda) - Adv_{2,j,1}(\lambda)| = \frac{d}{p}$$

where d is the number of decryption attempts made by the adversary.

Lemma 10 (Indistinguishability of $\mathsf{Game}_{2,j,2}$ and $\mathsf{Game}_{2,j3}$). *Any PPT adversary $\mathcal{A}$ that distinguishes between the two games with a non-negligible advantage, there exists $\mathcal{S}$ that has a non negligible advantage in solving the k-linear decision problem in $\mathbb{G}_2$, i.e.*

$$|Adv_{2,j,2}(\lambda) - Adv_{2,j,3}(\lambda)| \leq Adv_{\mathcal{S}}^{k-\mathsf{Lin}}(\lambda) + \frac{d}{p}$$

where d is the number of decryption attempts made by the adversary.

Proof. The proof is similar to that in appendix B, except that in addition to key queries, the adversary is allowed to perform a polynomial number (d) of decryption attempts using attribute vectors $\boldsymbol{v}$ where the entries corresponding to the corrupt authorities in $\mathcal{A}^{\star}$ can have arbitrary values. The probability that a given attribute vector $\boldsymbol{v} \in \mathbb{F}_p^{l+1}$ is orthogonal to the challenge ciphertext policy vector $\boldsymbol{x}_b$ is

$$Pr\left[\langle \boldsymbol{v}, \boldsymbol{x}_b \rangle = 0\right] = \frac{1}{p}$$

, which can be shown using the Schwartz-Zippel lemma.

D Inferring the Policy from Lewko-Waters Decentralized ABE Ciphertexts

We show that it is possible for an adversary to learn which attributes were included in the encryption policy of a Lewko-Waters ciphertext [18], without having the access matrix. We strengthen that the construction in [18] does not claim to protect the encryption policy, and therefore the result is quite expected. We simply illustrate that not providing the access policy is not enough to protect it, and the importance of proving that a scheme is actually policy-hiding.

Assume an adversarial coalition of colluding users that combine their knowledge of the public parameters $\{PK_i\}$ of the attribute-authorities. It picks coefficients $c_x \in \mathbb{Z}_N$, that correspond to including certain attributes as part of the policy. Next, it computes

$$e\left(\prod_x C_{3,x}^{c_x}, g_1\right) = \prod_x e\left(g_1, g_1\right)^{c_x y_{\rho(x)} r_x}$$

and tests whether it equals to

$$\prod_x e\left(C_{2,x}, g_1^{y_{\rho(x)}}\right)^{c_x} = \prod_x e\left(g_1, g_1\right)^{c_x y_{\rho(x)} r_x} \quad .$$

If the two are equal, the adversary has successfully guessed the encryption policy with high probability.