Hierarchical Policy Delegation in Multiple-Authority ABE

Peng Wang and Chinya Ravishankar

Department of Computer Science and Engineering
University of California, Riverside
{wangpe, ravi@cs.ucr.edu}

Abstract. We present HM-ABE, a hierarchical multi-authority attribute-based encryption scheme with policy delegation, that generalizes current work significantly. Current methods require encryptors to build ciphertext access policies themselves, using attributes published by authority domains. This causes problems, both since authorities may not publish sensitive attributes, and since users may not understand their internal policies. We permit encryptors to delegate parts of their access policies to authorities, who can construct appropriate policies on their behalf, using sensitive attributes, if needed. Delegation can be recursive.
Delegation helps encryptors build more accurate access policies, especially when they must include attributes from multiple authorities. HM-ABE greatly reduces the chances that ineligible users gain access to data, or that eligible users are denied. Delegation lets authorities hide sensitive attributes, while still allowing users indirect access to their semantics.
We show that HM-ABE achieves recursive attribute delegation, selective attribute hiding, and prove that it is secure.

Keywords: Attribute-based encryption, policy delegation, multiple authorities.

1 Introduction

Standard public-key encryption associates a ciphertext with a single public key, so only users holding the matching private key can decrypt the ciphertext. Attribute-Based encryption (ABE) [1–6] is a more powerful approach that permits expressive *ciphertext access policies* defining which combinations of *attributes* empower users to decrypt ciphertexts. Access policies in ABE can incorporate multiple attributes, **and**, **or** and threshold gates. In Ciphertext-Policy ABE (CP-ABE) [1, 7], each user holds a set of attributes, and each ciphertext is associated with an access policy, expressed as a Boolean formula over attributes. A central authority (CA) publishes the public key corresponding to each attribute, for use by encryptors in creating ciphertexts. Each user is granted a capability (which we call a *warrant*) corresponding to each attribute she holds by the authority maintaining the attribute. Only decryptors whose attributes satisfy the access policy can decrypt the ciphertext.

But every coin has two sides. Expressive policies facilitate fine-grained access control but make policy building difficult. Current CP-ABE schemes are hard to use and error-prone; they require encryptors, who may be common end-users, to master complex policy-making rules, and build policies themselves. Patients building access policies for encrypted Personal Health Records (PHRs), for example, must specify who may decrypt these PHRs through carefully crafted Boolean expressions over user attributes. However, patients may not have enough medical knowledge to build right access policies.

The emergence of Multi-authority Attribute-Based Encryption (M-ABE) [8–11] complicates matters considerably. M-ABE permits multiple authorities, each maintaining its own set of attributes, and publishing public keys for them. Each authority is allowed to have its own rules for how its attributes are to be used in building policies. Encryptors in M-ABE may have to build policies consistent across several domains, and must master internal rules for these authorities. This is hard even for experts, and simply impossible for most users.

Another problem of current M-ABE schemes is that authorities may see some attributes as too sensitive to publish public keys or policy building rules for these attributes. Consider some companies cooperate to provide services to a large number of customers, who for privacy reasons, submit encrypted requests for service. Each company has its own authority and maintains its attributes. Customers' access polices may contain attributes from several authorities. In a standard implementation of M-ABE, customers would directly encode, in the access policy for each request, the attributes of the service personnel (or groups) responsible for handling that request. But this is clearly unreasonable. Customers should not be required to understand the details of how each type of request is handled within each company. This approach also leaks information on each company's internal processes and structures, perhaps even compromises trade secrets.

Access policies based on the Linear Secret Sharing Scheme (LSSS) of [12] are currently regarded as the most expressive [2, 7]. In current M-ABE schemes, only [8, 13] support such access policies.

1.1 Our Contributions

We present a new Hierarchical Multi-Authority ABE scheme (HM-ABE) based on decentralized ABE (DABE) [8] to address these issues. We preserve all of DABE's advantages, and enhance it, adding *policy delegation* and *attribute hiding*.

Policy delegation allows an encryptor to delegate the task of building part of her policy to a trusted authority, using attributes it controls. Authorities in HM-ABE may publish *delegated* attributes, to serve as placeholders for access policies delegated to them. Encryptors can use delegated attributes in their policies. Authorities then build their own sub-policies to replace such delegated attributes. Encryptors securely send high-level descriptions of their intended policies to these authorities so they can build the right sub-policies. Such descriptions are high-level and non-formal, and sufficient only for policy-building.

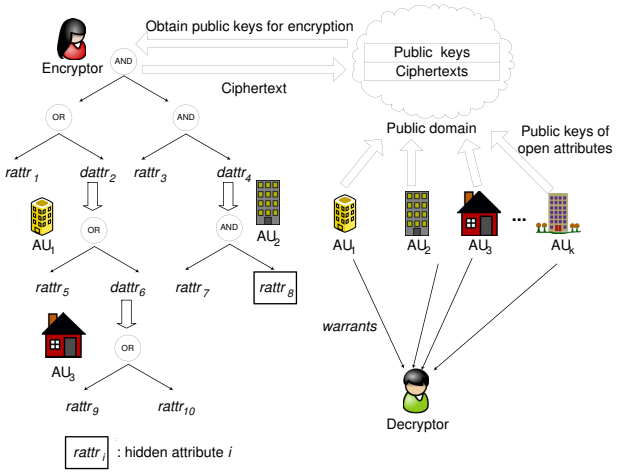

Fig. 1: The model of HM-ABE

Attribute hiding in HM-ABE lets authorities safeguard sensitive attributes by selectively publishing or hiding any of their local attributes. No public key is published for a hidden attribute, and only the authority managing the hidden attribute can use the hidden attribute's public key in encryption. Only decryptors explicitly authorized with the warrant for a hidden attribute can recognize it in polices. Encryptors access hidden attributes indirectly through policy delegation. Authorities can use policy delegation to use sensitive attributes in internal sub-policies, without their becoming public.

HM-ABE is significantly harder to prove correct than DABE, since the security game used in the proof is more complex, and HM-ABE supports recursive delegation. Nonetheless, we succeed in proving that HM-ABE is secure, even given a more powerful adversary.

HM-ABE is the only M-ABE scheme having features such as policy delegation, and attributing hiding. HM-ABE is based on DABE, and supports LSSS-based access policies too. [13] is another M-ABE scheme supports LSSS-based policies, but it does not have the other features of HM-ABE. Other M-ABE schemes [9,11] do not even support LSSS-based access policies. Besides, schemes like [9, 13] require the existence of trusted CA(s) which cannot be corrupted, while our scheme and DABE do not have this restriction.

The rest of the paper is organized as follows. The model of HM-ABE appears in Sec. 2. Sec. 3 discusses preliminaries. We describe the differences between HM-ABE and DABE in Sec. 4. Sec. 5 presents our scheme. Related work appears in Sec. 7. Sec. 8 concludes the paper. We present the security proof of HM-ABE in Appendix.

2 Overview of Our Scheme

Our scheme consists of two kinds of entities: *users* and *authorities*. Users may be encryptors or decryptors, and are known by global *user identifiers* (UIDs). Users have *attributes*, which are numbered. Authorities manage attributes and their assignment to users. They publish public keys for *open* attributes for encryptors to use. Open attributes can be used in an access policy by any encryptor. Nothing is published for *hidden* attributes. A hidden attribute may be used only by the authority managing it, in its sub-policies. For each attribute i that it manages, an authority generates a *master secret key* σ_i, and publishes the matching public key π_i if i is open. If user UID holds attribute i, he gets a *warrant* $\varpi_{\mathrm{UID},i}$ from the authority managing attribute i. The authority computes $\varpi_{\mathrm{UID},i}$ from σ_i and UID.

Authorities also handle delegation requests sent from encryptors or other authorities. Encryptors build policies using attributes they know, and encrypt their data based on these policies. Decryptors will decrypt ciphertexts using warrants they have been granted.

Attributes may be *regular* or *delegated*. Regular attributes describe users, and correspond to attributes as in current M-ABE schemes. A delegated attribute serves as a proxy for the sub-policy an authority designs for an encryptor or another authority. Encryptors may indirectly use hidden attributes through using delegated attributes in their policies since authorities can build sub-policies including their hidden attributes.

Fig. 1 shows the model of HM-ABE. Let $rattr_i$ and $dattr_j$ denote regular attribute i and delegated attribute j. Policies are encoded as *access structures,* which are boolean formulas over attributes. *Access trees* are equivalent structures, whose leaves are attributes and internal nodes are **and** or **or** gates. The access structure created by the encryptor is called the *main access structure*.

In Fig. 1, the encryptor has specified the main access structure ($rattr_1$ **or** $dattr_2$) **and** ($rattr_3$ **and** $dattr_4$). She first obtains attribute public keys stored publicly, and encrypts the data with the public keys of regular attributes $rattr_1$ and $rattr_3$. She publishes these ciphertexts, say in the cloud. Decryptors holding $rattr_3$ cannot decrypt the ciphertext, since they hold no warrant for the delegated attribute $dattr_4$. They must wait for AU_2 to replace $dattr_4$ by a sub-policy.

The encryptor then solicits the help of AU_1 and AU_2, sending them descriptions of her desired policy, encrypted with the public keys of delegated attributes $dattr_2$ and $dattr_4$, respectively. AU_1 and AU_2 may receive different policy descriptions. Each of them decrypts the description, and builds access sub-policies appropriate to its domain, in the form of access sub-structures or access subtrees. Each authority creates the ciphertext for its sub-structure, and sends the ciphertext to the cloud.

Delegated attributes are replaced by the corresponding access sub-structures. $dattr_2$ is replaced by ($rattr_5$ **or** $dattr_6$) within AU_1, and $dattr_4$ by ($rattr_7$ **and** $rattr_8$) within AU_2. Here $rattr_8$ is a hidden attribute. A delegated attribute in an access sub-structure can also be replaced by a sub-structure. Thus, $dattr_6$ is replaced by ($rattr_9$ **and** $rattr_{10}$) within AU_3. The ciphertext is now associated with a more

complex access structure called a *complete access structure* containing thirteen nodes, while the main structure contained only seven. Moreover, the new policy uses hidden attributes, which are unavailable to the encryptor without delegation.

A user satisfies an access structure under the following conditions. Each attribute in the access structure is treated as a Boolean **true** if and only if it is also held by the user. The access structure is satisfied if and only if the corresponding Boolean expression evaluates to **true**.

3 Preliminaries and Overview

3.1 Access Structures

Let $\mathcal{U} = \{i_1, i_2, \ldots, i_n\}$ be a set of attributes. A collection $\mathbb{A} \subseteq 2^{\mathcal{U}}$ is *monotone* if whenever $B \subseteq C$ and $B \in \mathbb{A}$, we also have $C \in \mathbb{A}$.

Definition 1 *An access structure is a monotone collection $\mathbb{A}$ of non-empty subsets of $\mathcal{U}$. The sets in $\mathbb{A}$ are the authorized sets. All other sets are unauthorized.*

3.2 Bilinear Maps

Let G and G_T denote two multiplicative cyclic groups of order N. In HM-ABE, N is the product of three distinct primes. Let g be a generator of G, and $e : G \times G \to G_T$ be a bilinear map with the following properties:

1. *Bilinearity*: for all $u, v \in G$ and $a, b \in \mathbb{Z}_N$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2. *Non-degeneracy*: $e(g, g) \neq 1$.
3. *Computability*: There is an efficient algorithm to compute $e(u, v)$ for any $u, v \in G$.

3.3 Linear Secret-Sharing Schemes (LSSS)

We use the definitions of LSSS in [12]:

Definition 2 *Let the attributes in access structure $\mathbb{A}$ be from $\mathcal{U}$. A secret sharing scheme $\mathbb{\Lambda}$ for $\mathbb{A}$ is linear (over $\mathbb{Z}_N$) if*

1. *The shares of the attributes form a vector over $\mathbb{Z}_N$.*
2. *There is an $l \times m$ share-generating matrix A and a function ρ mapping row numbers in A to attributes in $\mathcal{U}$. To share secret $s \in \mathbb{Z}_N$, we first randomly choose $r_2, \ldots, r_m \in \mathbb{Z}_N$, and form $v = (s, r_2, \ldots, r_m)^T$. The l shares of s according to $\mathbb{\Lambda}$ are given by the vector Av. If $[Av]_x$ denotes the xth element in Av, $[Av]_x$ is the secret share belonging to attribute $\rho(x)$. [8] shows how to obtain A and ρ from $\mathbb{A}$.*

Every LSSS $\mathbb{\Lambda} = (A, \rho)$ has the *linear reconstruction* property. Let $S \in \mathbb{A}$ be an authorized set, and $I \subset \{1, 2, \ldots, l\}$ be defined as $I = \{x : \rho(x) \in S\}$. Let $[A]_x$ denote row x of A. There are constants $\{c_x \in \mathbb{Z}_N\}_{x \in I}$ with $\sum_{x \in I} c_x [A]_x = (1, 0, \ldots, 0)$. If $\{\lambda_x\}_{x \in I}$ are the shares of s corresponding to attributes in S, we have $\sum_{x \in I} c_x \lambda_x = s$. These c_x can be found in time polynomial in the size of the matrix A.

3.4 Outline Of DABE

We briefly describe DABE [8]. Let p_1, p_2, p_3 be three distinct large primes, and $N = p_1 p_2 p_3$. Let G and G_T denote two cyclic groups of order N, and $e : G \times G \to G_T$ denote a bilinear map. Let G_{p_i} denote the subgroup of order p_i in G for $1 \leq i \leq 3$, and $G_{p_i p_j}$ denote the subgroup of order $p_i p_j$ in G for $1 \leq i < j \leq 3$. G_{p_1} is used in the construction of DABE. The other groups are used in proofs. Let g_1 be a generator of G_{p_1}. Define $\varepsilon = e(g_1, g_1)$. DABE operates as follows. Let $\mathbb{A}$ be an access structure. Let $\mathbb{T}$ be its access tree form, with its leaves representing attributes.

The Basic Idea A encryptor E encrypts message M by choosing a random secret $s \in \mathbb{Z}_N$ and obtain M's ciphertext as $C_M = M\varepsilon^s$. Any user who can recover ε^s can recover $M = C_M / \varepsilon^s$. DABE implements the policy encoded into $\mathbb{T}$ by using it to dictate who can recover ε^s, as follows.

Let $\mathbb{\Lambda} = (A, \rho)$ be an LSSS matching $\mathbb{T}$'s structure. If ε^s is shared under $\mathbb{\Lambda}$, the linear reconstruction property guarantees efficiently computable c_x, λ_x for each attribute $\rho(x)$ represented by a leaf in $\mathbb{T}$, such that $\sum_{x \in I} c_x \lambda_x = s$ for any authorized set S and corresponding $I = \{x : \rho(x) \in S\}$. Since $\mathbb{\Lambda}$ is public, any user can efficiently compute the c_x from A.

Let us give ε^{λ_x} to each user holding attribute $\rho(x)$. A user who holds all the attributes in an authorized set S can compute $\{c_x\}_{x \in I}$ from A, followed by $\prod_{x \in I} \left(\varepsilon^{\lambda_x}\right)^{c_x} = \varepsilon^{\sum_{x \in I} \lambda_x c_x} = \varepsilon^s$. Hence, he can recover ε^s and subsequently, message M, if and only if he holds all attributes in an authorized set.

Preventing Collusions This method is secure against individual users, but if users who can jointly cover an authorized set S collude, they can recover ε^s by pooling ε^{λ_x} values. Collusions are prevented by encoding user UIDs with the attribute "shares" ε^{λ_x} as follows.

Let 0 be shared under $\mathbb{\Lambda}$, and the share corresponding to attribute $\rho(x)$ be ω_x. Let H denote a hash function that maps global identities UID to elements of G. Define $\varepsilon_{\text{UID}} = e(H(\text{UID}), g_1)$ for each user UID. If user U_1 has attribute $\rho(x)$, he gets $\varepsilon^{\lambda_x} \cdot \varepsilon_{\text{U}_1}^{\omega_x}$. Now, if user U_1 holds all attributes in authorized set S, he can compute $\prod_{x \in I} \left(\varepsilon^{\lambda_x} \varepsilon_{\text{U}_1}^{\omega_x}\right)^{c_x} = \varepsilon^{\sum_{x \in I} \lambda_x c_x} \cdot \varepsilon_{\text{U}_1}^{\sum_{x \in I} \omega_x c_x}$. Since the ω_x are all shares of 0 under $\mathbb{\Lambda}$, $\sum_{x \in I} \omega_x c_x = 0$, and the above expression reduces to ε^s. However, collusion between users U, V is now ruled out, since $\varepsilon^{\lambda_x} \varepsilon_{\text{U}}^{\omega_x}$ and $\varepsilon^{\lambda_{x'}} \varepsilon_{\text{V}}^{\omega_{x'}}$ do not combine in the above manner.

Setup:
Let $\mathcal{A}$ denote the adversary and $\mathcal{C}$ denote the challenger. $\mathcal{C}$ initializes a set $\mathcal{U}$ of attributes and a set $\mathcal{S}$ of authorities. $\mathcal{A}$ selects a set $\mathcal{S}' \subset \mathcal{S}$ of corrupted authorities. $\mathcal{A}$ gets public keys of all attributes.
Phase 1:
$\mathcal{A}$ requests and gets from $\mathcal{C}$ a series of warrant $\varpi_{\mathrm{UID},i}$, for UIDs and attributes i of $\mathcal{A}$'s choice.
Challenge Phase:
$\mathcal{A}$ creates an access structure $\mathbb{A}$ such that $\mathcal{A}$ cannot satisfy $\mathbb{A}$ using just the $\varpi_{\mathrm{UID},i}$ he obtained in Phases 1 and 2, and those that corrupted authorities can generate.
$\mathcal{A}$ creates two messages M_0, M_1. $\mathcal{C}$ chooses a bit β, encrypts M_β under $\mathbb{A}$, and sends the ciphertext to $\mathcal{A}$.
In HM-ABE's security games, $\mathcal{A}$ has all abilities of the adversary in DABE's security game, besides, $\mathcal{A}$ obtains the security shares λ_x, ω_x for each attribute $\rho(x)$ controlled by corrupted authorities. In DABE's security games, $\mathcal{A}$ does not have these shares.
Phase 2:
$\mathcal{A}$ asks $\mathcal{C}$ for more warrants $\varpi_{\mathrm{UID},i}$.
Guess:
$\mathcal{A}$ outputs a guess β' for β, and wins if $\beta' = \beta$.

Game 1: General structure of security games of DABE and HM-ABE

Definition 3 *We call each $\varepsilon^{\lambda_x}\varepsilon_{\mathrm{U}}^{\omega_x}$ value a U-share of secret s for attribute $i = \rho(x)$, denoting it as $s_{\langle \mathrm{U},i \rangle}$.*

4 From DABE to HM-ABE

We choose DABE as the base to build our scheme, and encryption and decryption in our scheme mirror DABE. The security games of HM-ABE are significantly more complex than those of DABE because of our added features. Proving these features are secure is difficult. We describe how to achieve our new features in DABE to obtain HM-ABE in the following.

4.1 Handling Delegation

As shown in Sec. 3.4, a decryptor needs to combine all U-shares of attributes in an authorized set to generate ε^s. Since a policy may incorporate sub-policies from several authorities, we must ensure U-shares can be combined correctly, whether they originate from the encryptor or from various authorities.

We solve this problem differently from DABE. In both schemes, the encryptor creates a ciphertext using a secret s, and generates shares λ_x and ω_x of s for each attribute $\rho(x)$ used in the access policy. In DABE, these shares are embedded into the policy ciphertext, but not passed directly to any entity. In contrast, we pass both λ_x, ω_x for a delegated attribute $\rho(x)$ directly to the authority that controls $\rho(x)$.

Say encryptor E delegates a sub-policy to authority AU_j via delegated attribute δ. E encrypts message M with secret s, specifying δ in his LSSS $\mathbb{\Lambda} =$

(A, ρ). Let $\rho(x) = \delta$. He creates a policy description $\text{Pol}_{M,\delta}$, gets the attribute shares λ_x, ω_x, and encrypts $\text{Pol}_{M,\delta}$, λ_x, ω_x by δ's public key π_δ. Then the encrypted message is sent to AU_j. Please note here that E directly gives secrets λ_x, ω_x to AU_j, while in DABE, the adversary can only recover a U-share $s_{\langle \text{U},\delta \rangle} = \varepsilon^{\lambda_x} \varepsilon_{\text{U}}^{\omega_x}$ for any user ID U.

AU_j creates a local access structure $\mathbb{A}_j$ and LSSS $\mathbb{\Lambda}_j$ for this delegation, in line with $\text{Pol}_{M,\delta}$ and AU_j's policies, and shares λ_x, ω_x under $\mathbb{\Lambda}_j$. As Section 5 shows, a decryptor U who satisfies the access structure $\mathbb{A}_j$ can recover $s_{\langle \text{U},\delta \rangle}$. Delegation can be recursive.

Directly giving authorities secret shares λ_x, ω_x enables policy delegation but complicates the security game. The boxed figure Game 1 shows the overall structure of the security games of DABE and HM-ABE. They are different in the challenge phase, where the adversary obtains secret shares λ_x, ω_x of delegated attributes controlled by corrupted authorities in the security game of HM-ABE while he cannot in the security game of DABE. We must prove that HM-ABE is secure even if the adversary is more powerful.

Another challenge is that HM-ABE allows hierarchical delegation, so one authority may delegate a policy to another authority. This feature does not exist in DABE. We must prove that hierarchical delegation does not compromise security. We prove this in two steps in Appendix. First, we prove that HM-ABE is secure when no delegation is allowed. We show in this case that it is safe to release λ_x and ω_x to authorities. Next, we prove inductively that HM-ABE is secure with an arbitrary number of recursive delegations.

4.2 Handling Hidden Attributes

We achieve attribute hiding by modifying the function ρ for hidden attributes as follows. For each row $[A]_x$ in A, if x is associated with an open attribute i, $\rho(x)$ will map x to the corresponding attribute's ID $\rho(x) = i$. Let C_M be the main ciphertext and $h_{\mathcal{C}}()$ be a hash function. If x is associated with a hidden attribute i, we define $\rho(x) = h_{\mathcal{C}}(C_M || i)$, where $||$ denotes concatenation. Only users holding the warrant for i are given i's ID, and can know that the row x is associated with attribute i.

5 Construction of HM-ABE

For convenience, we associate an attribute $\widehat{v}$ with each node v in an access tree $\mathbb{T}$. If v is a leaf, $\widehat{v}$ is a regular or delegated attribute. If v is an internal node, $\widehat{v}$ is a *virtual attribute*, which we will find useful in the exposition. The root of tree $\mathbb{T}$ is associated with the virtual attribute $\widehat{\mathbb{T}}$.

Definition 4 *A user U satisfies an access tree $\mathbb{T}$ for a secret encryption key s, if the following holds.*

1. *If $\mathbb{T}$ consists of a single node v (a leaf), he obtains his U-share $s_{\langle \text{U},\widehat{v} \rangle}$ of s for attribute $\widehat{v}$.*

2. *If* $\mathbb{T}$ *is an OR node with children* $\mathbb{L}$ *and* $\mathbb{R}$*, satisfies either* $\mathbb{L}$ *or* $\mathbb{R}$*. In these cases, respectively,* $\mathbb{T}$*'s U-share is* $s_{\langle \mathrm{U},\widehat{\mathbb{T}}\rangle} = s_{\langle \mathrm{U},\widehat{\mathbb{L}}\rangle}$*, or* $s_{\langle \mathrm{U},\widehat{\mathbb{T}}\rangle} = s_{\langle \mathrm{U},\widehat{\mathbb{R}}\rangle}$*.*
3. *If* $\mathbb{T}$ *is an AND node with children* $\mathbb{L}$ *and* $\mathbb{R}$*, satisfies both* $\mathbb{L}$ *and* $\mathbb{R}$*. Now,* $s_{\langle \mathrm{U},\widehat{\mathbb{T}}\rangle} = s_{\langle \mathrm{U},\widehat{\mathbb{L}}\rangle} \cdot s_{\langle \mathrm{U},\widehat{\mathbb{R}}\rangle}$*.*

As described in Sec. 3, we apply linear secret sharing over access structures defined over attributes, ensuring that a user can recover a secret only if he has U-shares for all attributes in some authorized set.

5.1 Notation and Algorithms

Let $\mathcal{O}_j$ and $\mathcal{C}_j$ be the set of open and hidden attributes managed by authority AU_j, and let $\mathcal{U}_j = \mathcal{O}_j \cup \mathcal{C}_j$. Let $\mathcal{O} = \bigcup \mathcal{O}_j$ be the set of all open attributes, and $\mathcal{R}, \mathcal{D} \subseteq \mathcal{O}$ be the set of regular and delegated attributes, respectively. Let $\mathbb{A}$ and $\mathbb{A}_j$ denote the main access structure created by the encryptor and the sub-structure created by AU_j. Let $\mathbb{\Lambda} = (A, \rho)$ and $\mathbb{\Lambda}_j = (A_j, \rho_j)$ denote LSSS schemes defined for $\mathbb{A}$ and $\mathbb{A}_j$, respectively.

For each attribute i, we create a pair (σ_i, π_i), where σ_i is a *master secret key* and π_i is an *attribute public key.* All σ_i are kept secret. π_i is published if i is open. Let $\Sigma_j = \{\sigma_i | \, i \in \mathcal{U}_j\}$ and $\Pi_j = \{\pi_i | \, i \in \mathcal{U}_j\}$ be the set of secret keys and public keys for all attributes AU_j manages. Let $\Pi = \bigcup \Pi_j$ be the set of all public keys. We use the following algorithms.

Global Initialization Two cyclic groups G, G_T of order $N = p_1 p_2 p_3$ are selected based on the the security parameter κ. $e : G \times G \to G_T$ is a bilinear map. The set Γ of global public parameters comprises N and a generator g_1 of subgroup G_{p_1} of G. A hash function $H : \{0,1\}^* \to G$ that maps global identities UID to elements of G is published. Another hash function $h_{\mathcal{C}} : \{0,1\}^* \to \mathbb{Z}_N$ that maps the concatenation of main ciphertexts and attribute identities $C_M || ID$ to elements of $\mathbb{Z}_N$ is also published.

Authority Initialization For each attribute $i \in \mathcal{U}_j$ that it manages, AU_j randomly chooses $\sigma_i = (\alpha_i, \chi_i) \in \mathbb{Z}_N^2$ as its master secret key, kept secret for all i. If i is open, both i and its public key $\pi_i = (e(g_1, g_1)^{\alpha_i}, g_1^{\chi_i})$ are published. If i is hidden, its public key and ID will not be published, but a hash $h_{\mathcal{C}}(C_M || i)$ is published in encryption, as described in Section 3.4.

Warrants A user D who holds attribute i gets a *warrant* for i, computed as $\varpi_{D,i} = g_1^{\alpha_i} H(\mathrm{D})^{\chi_i}$. Warrants are used to recover the U-shares $s_{\langle \mathrm{D},i\rangle}$. If i is hidden, the user holding the warrant for i also gets attribute i's ID.

Main Encryption Algorithm Let the LSSS of the main access structure be $\mathbb{\Lambda} = (A, \rho)$. To encrypt message M, choose a random $s \in \mathbb{Z}_N$ and generate random vectors $v \in \mathbb{Z}_N^m$ with s as its first entry and $w \in \mathbb{Z}_N^m$ with 0 as its first

How E **encrypts message** M**:**
Given access structure $\mathbb{A}$ and LSSS $\mathbb{\Lambda} = (A, \rho)$, pick a random key s. Compute ciphertext $C_M = Me(g_1, g_1)^s$.
For each attribute $i = \rho(x)$ in the access structure $\mathbb{A}$, compute the triple $(Y_{1,x}, Y_{2,x}, Y_{3,x})$, and store with the leaf in $\mathbb{T}$ corresponding to attribute i.

How D **decrypts ciphertext** C_M**:**
For each attribute $i = \rho(x)$ that D holds in access tree $\mathbb{T}$, use warrant $\varpi_{D,i}$ to decrypt the triple $(Y_{1,x}, Y_{2,x}, Y_{3,x})$ to get the U-share $s_{\langle \mathrm{D},i \rangle}$. Aggregate the $s_{\langle \mathrm{D},i \rangle}$, proceeding up the tree, and recover $e(g_1, g_1)^s$. Compute $M = C_M / e(g_1, g_1)^s$.

Algorithm 1: Outline of Encryption and Decryption.

entry. Let $\lambda_x = [A]_x \cdot v$, $\omega_x = [A]_x \cdot w$. Randomly select $r_x \in \mathbb{Z}_N$. We generate C_M, the actual ciphertext for M, as well as a triple $(Y_{1,x}, Y_{2,x}, Y_{3,x})$ for each regular attribute $i = \rho(x)$ in $\mathbb{A}$. These triples will yield the U-shares $s_{\langle \mathrm{D},i \rangle}$ for each attribute i when decrypted with the warrant $\varpi_{\mathrm{D},i}$, and are associated with the leaf of the access tree $\mathbb{T}$ corresponding to attribute i.

$$\begin{aligned} C_M &= Me(g_1, g_1)^s, & Y_{1,x} &= e(g_1, g_1)^{\lambda_x + r_x \alpha_{\rho(x)}} \\ Y_{2,x} &= g_1^{r_x}, & Y_{3,x} &= g_1^{\omega_x + r_x \chi_{\rho(x)}} \end{aligned} \tag{1}$$

Encryptor's use of Delegation Let encryptor E's policy use attribute δ delegated by AU_j. Let $\mathrm{Pol}_{M,\delta}$ be a description of the policy E delegates to AU_j via δ under the LSSS $\mathbb{\Lambda} = (A, \rho)$. Let $\rho(x) = \delta$. E chooses a random $q_x \in \mathbb{Z}_N$ and encrypts $\Delta_{M,\delta} = \{\mathrm{Pol}_{M,\delta} || (\lambda_x, \omega_x)\}$ with a new generated symmetric key $K_{M,\delta}$. $K_{M,\delta}$ then is encrypted with the public key π_δ. Let $\langle \Delta_{M,\delta} \rangle_{K_{M,\delta}}$ denote the message $\Delta_{M,\delta}$ encrypted by the symmetric key $K_{M,\delta}$. E obtains the delegation ciphertext $Y'_{M,\delta} = (Y'_{1,x}, Y'_{2,x}, Y'_{3,x})$ as follows.

$$\begin{aligned} Y'_{1,x} &= \langle \Delta_{M,\delta} \rangle_{K_{M,\delta}} \\ Y'_{2,x} &= K_{M,\delta} e(g_1, g_1)^{q_x \alpha_{\rho(x)}}, \quad Y'_{3,x} = g_1^{q_x} \end{aligned} \tag{2}$$

Processing delegations Let authority AU_j receive the encrypted delegation messages $Y'_{1,x}, Y'_{2,x}, Y'_{3,x}$ in Eqn. 3. AU_j first recovers $K_{M,\delta}$ as:

$$K_{M,\delta} = \frac{Y'_{2,x}}{e(Y'_{3,x}, g_1^{\alpha_{\rho(x)}})} = \frac{K_{M,\delta} e(g_1, g_1)^{q_x \alpha_{\rho(x)}}}{e(g_1^{q_x}, g_1^{\alpha_{\rho(x)}})} \tag{3}$$

Then AU_j uses $K_{M,\delta}$ to decrypt $Y'_{1,x}$, and obtains $\Delta_{M,\delta} = \{\mathrm{Pol}_{M,\delta} || (\lambda_x, \omega_x)\}$.

A local access structure $\mathbb{A}_j$ and an LSSS $\mathbb{\Lambda}_j = (A_j, \rho_j)$ are created as per $\mathrm{Pol}_{M,\delta}$. For each row $[A_j]_y$ of A_j, let i be the attribute associated with $[A_j]_y$. If i is a hidden attribute, $\rho_j(y) = h_{\mathcal{C}}(C_M || i)$, otherwise $\rho_j(y) = i$. Let A_j be $l_j \times m_j$. As before, we create triples to allow decryptors to recover their U-shares for each attribute. Choose random vectors $v_j, u_j \in \mathbb{Z}_N^{m_j}$ such that v_j and u_j have λ_x and

ω_x as their first entries, respectively. Let $\lambda_y = [A_j]_y \cdot v_j$ and $\omega_y = [A_j]_y \cdot u_j$. For each row $[A_j]_y$ of A_j, choose a random $r_y \in \mathbb{Z}_N$. The triples are

$$\begin{aligned} Z_{1,y} &= e(g_1, g_1)^{\lambda_y + r_y \alpha_i}, \quad Z_{2,y} = g_1^{r_y}, \\ Z_{3,y} &= g_1^{\omega_y + r_y \chi_i} \end{aligned} \tag{4}$$

There is no C_M. U-shares are our only concern.

Decryption algorithm Let $\mathbb{A}$ and $\{\mathbb{A}_j\}$ denote the main access structure and all sub-structures associated with the ciphertext. The decryptor D first builds the complete access structure $\mathbb{A}^*$, replacing delegated attributes with sub-structures. If D's attributes satisfy $\mathbb{A}^*$, he first generates his U-shares $s_{\langle \mathrm{D},i \rangle}$ using his warrant $\varpi_{D,i}$, as we shall explain presently. He then recovers M as follows.

If $\mathbb{\Lambda} = (A, \rho)$ is an LSSS for $\mathbb{A}$, the attributes of D that satisfy $\mathbb{A}$ define a subset $[A]_x$ of A's rows and constants c_x such that $\sum_x c_x [A]_x = (1, 0, \ldots, 0)$. Having generated the U-share $s_{\langle \mathrm{D},i \rangle} = e(g_1, g_1)^{\lambda_x} e(H(D), g_1)^{\omega_x}$ for each such x (see below), D finds $c_x \in \mathbb{Z}_N$ and computes:

$$\begin{aligned} e(g_1, g_1)^s &= \prod_x \left(e(g_1, g_1)^{\lambda_x} e(H(D), g_1)^{\omega_x} \right)^{c_x} \\ &= e(g_1, g_1)^{\sum_x c_x \lambda_x} e(H(D), g_1)^{\sum_x c_x \omega_x} \end{aligned} \tag{5}$$

$$M = C_M / e(g_1, g_1)^s \tag{6}$$

D generates the U-share for an open regular attribute i from ciphertexts in Eqns. 1 and 2 and his warrant $\varpi_{D,i}$. Let $\rho(x) = i$. Following DABE [8],

$$\begin{aligned} s_{\langle \mathrm{D},i \rangle} &= Y_{1,x} \cdot e(H(D), Y_{3,x}) / e(\varpi_{D,i}, Y_{2,x}) \\ &= \frac{e(g_1, g_1)^{\lambda_x + r_x \alpha_{\rho(x)}} e(H(D), g_1^{\omega_x + r_x \chi_{\rho(x)}})}{e(g_1^{\alpha_{\rho(x)}} H(\mathrm{D})^{\chi_{\rho(x)}}, g_1^{r_x})} \\ &= e(g_1, g_1)^{\lambda_x} \cdot e(H(D), g_1)^{\omega_x}. \end{aligned} \tag{7}$$

For a delegated attribute δ where $\rho(x) = \delta$, let $\mathbb{\Lambda}_j = (A_j, \rho_j)$ be the LSSS for the sub-structure $\mathbb{A}_j$ replacing δ. D's attributes satisfying $\mathbb{A}_j$ define a subset $[A_j]_y$ of A_j's rows and constants $c_y \in \mathbb{Z}_N$ such that $\sum_y c_y [A_j]_y = (1, 0, \ldots, 0)$. Having generated the U-share $s_{\langle \mathrm{D},i \rangle} = e(g_1, g_1)^{\lambda_y} e(H(D), g_1)^{\omega_y}$ for each such y, D finds $c_y \in \mathbb{Z}_N$ and computes:

$$\begin{aligned} s_{\langle \mathrm{D},\delta \rangle} &= \prod_y \left(e(g_1, g_1)^{\lambda_y} e(H(D), g_1)^{\omega_y} \right)^{c_y} \\ &= e(g_1, g_1)^{\sum_y c_y \lambda_y} e(H(D), g_1)^{\sum_y c_y \omega_y} \\ &= e(g_1, g_1)^{\lambda_x} e(H(D), g_1)^{\omega_x} \end{aligned} \tag{8}$$

For the attribute i associated with $[A_j]_y$, if i is an open regular attribute, D generates the U-share from ciphertexts in Eqn. 4 and his warrant $\varpi_{D,i}$. We

have:

$$s_{\langle \mathrm{D},i \rangle} = Z_{1,y} \cdot e(H(D), Z_{3,y})/e(\varpi_{D,i}, Z_{2,y}) \\ = e(g_1, g_1)^{\lambda_y} \cdot e(H(D), g_1)^{\omega_y}. \tag{9}$$

If i is hidden, D generates $s_{\langle \mathrm{D},i \rangle}$ from $\rho(y) = h_{\mathcal{C}}(C_M||i)$ as follows. If D holds the hidden attributes $c_1, c_2, \ldots, c_t$, he holds the warrants $\{\varpi_{D,c_k} | k = 1, \ldots, t\}$ and IDs $\{c_k | k = 1, \ldots, t\}$. First, D computes $h_{\mathcal{C}}(C_M||c_k)$ for each c_k. If there is a match, he infers the warrant $\varpi_{D,i}$, and uses Eqn. 9 to generates the proper U-share. Unless a decryptor already holds i's ID, he learns nothing about which hidden attribute was used by the encryptor. Delegated attributes within $\mathbb{A}_j$ are handled recursively using Eqn. 8.

6 Security Analysis

We first prove that it is safe to release secret shares to authorities when no delegation is allowed in HM-ABE, then we prove inductively that HM-ABE is secure with an arbitrary number of recursive delegations.

6.1 Security Model

We fix the security parameter κ at the outset. We show the security of HM-ABE by considering a series of games $\mathcal{G}_i$ between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

In the boxed figure Game 1, $\mathcal{A}$ creates two messages M_0 and M_1. $\mathcal{C}$ picks random bit β, encrypts M_β, and sends it to $\mathcal{A}$, who makes a guess β' for β.

Definition 5 *$\mathcal{A}$, who wins the game if $\beta' = \beta$, has the advantage* $\Pr[\beta' = \beta] - \frac{1}{2}$.

Definition 6 *$\mathcal{A}$ satisfies an access tree $\mathbb{T}$ for an encryption key s if $\mathcal{A}$ controls a* UID *that satisfies $\mathbb{T}$.*

Definition 7 *An encryption scheme is secure if all polynomial-time adversaries have only a negligible advantage in its security game $\mathcal{G}$.*

6.2 Assumptions

Our proofs are based on a set of assumptions, and assume the inability of a polynomial algorithm to distinguish between certain tuples $(\mathcal{D}, T_1)$ and $(\mathcal{D}, T_2)$ with non-negligible advantage. We say that a polynomial-time algorithm **A**'s advantage in distinguishing tuples $(\mathcal{D}, T_1)$ and $(\mathcal{D}, T_2)$ is

$$\mathfrak{A} = |\Pr[\mathbf{A}(\mathcal{D}, T_1) = 1] - \Pr[\mathbf{A}(\mathcal{D}, T_2) = 1]| \tag{10}$$

We follow the group construction in Sec. 5.1. Below, we adapt the assumptions in [8].

Assumption 1: Let T_1 be a random generator of G, and T_2, g_1 be random generators of G_{p_1}. No probabilistic polynomial-time algorithm **A** can distinguish

the tuples $(\mathcal{D} = \{N, G, G_T, e, g_1\}, T_1)$ and $(\mathcal{D}, T_2)$ with non-negligible advantage, the probability being taken over the random choice of g_1, T_1, T_2, and the random bits consumed by **A**.

Assumption 2: Let g_1, T_1, X_1 be random generators of G_{p_1}, X_2 be a random generator of G_{p_2}, g_3 be a random generator of G_{p_3}, and T_2 be a random generator of $G_{p_1p_2}$. No probabilistic polynomial-time algorithm **A** can distinguish between the two tuples $(\mathcal{D} = \{N, G, G_T, e, g_1, g_3, X_1X_2\}, T_1)$ and $(\mathcal{D}, T_2)$ with non-negligible advantage, the probability being taken over the random choice of g_1, g_3, X_1, X_2, T_1, T_2, and the random bits consumed by **A**.

Assumption 3: Let g_1, X_1 be random generators of G_{p_1}, Y_2 be a random generator of G_{p_2}, X_3, Y_3 be random generators of G_{p_3}, T_1 be a random generator of $G_{p_1p_2}$, and T_2 be a random generator of $G_{p_1p_3}$. No probabilistic polynomial-time algorithm **A** can distinguish tuple $(\mathcal{D} = \{N, G, G_T, e, g_1, X_1X_3, Y_2Y_3\}, T_1)$ from tuple $(\mathcal{D}, T_2)$ with non-negligible advantage, the probability being taken over random choices of g_1, g_3, X_1, X_3, Y_2, Y_3, T_1, T_2, and the random bits consumed by **A**.

Assumption 4: Let g_1, g_2, and g_3 be random generators of G_{p_1}, G_{p_2}, and G_{p_3}, respectively. Let a, b, c, d be random numbers in $\mathbb{Z}_N$. Let $T_1 = e(g_1, g_1)^{abc}$, and let T_2 be a random generator of G_T. No probabilistic polynomial-time algorithm **A** can distinguish tuples $(\mathcal{D} = \{N, G, G_T, e, g_1, g_2,\ g_3, g_1^a, g_1^b g_3^b, g_1^c, g_1^{ac} g_3^d\}, T_1)$ and $(\mathcal{D}, T_2)$ with non-negligible advantage, the probability being taken over the random choice of g_1, g_2, g_3, a, b, c, T_1, T_2, and the random bits consumed by **A**.

6.3 Proof of Security

Let $\mathcal{G}_i$ denote the security game parameterized as i, and $\mathfrak{A}(\mathcal{G}_i)$ denote $\mathcal{A}$'s advantage in game $\mathcal{G}_i$. Our proof is based on the following security games.

- $\mathcal{G}_R$ DABE's security game. Delegation forbidden. $\mathfrak{A}(\mathcal{G}_R)$ is known to be negligible [8].
- $\mathcal{G}_{R'}$ Security game $\mathcal{G}_R$ modified so $\mathcal{C}$ gives $\mathcal{A}$ all secret shares $\{\lambda_x, \omega_x\}$ of attribute $\rho(x)$ controlled by corrupted authorities. The access structure contains regular attributes only. Delegation forbidden.
- $\mathcal{G}_{L_n}$ Security game of HM-ABE when the access tree is of n-level delegation.

Boxed figure Game 1 shows the general structure of these games. We first prove that $\mathfrak{A}(\mathcal{G}_{R'})$ is negligible. We next show $\mathfrak{A}(\mathcal{G}_{L_1}) - \mathfrak{A}(\mathcal{G}_{R'})$ is negligible, so that $\mathfrak{A}(\mathcal{G}_{L_1})$ is also negligible. We then prove by induction that $\mathfrak{A}(\mathcal{G}_{L_n})$ is negligible for any $n > 1$.

Proof that $\mathfrak{A}(\mathcal{G}_{R'})$ is negligible

Theorem 1 *Under assumptions 1, 2, 3, every polynomial time adversary has negligible advantage in* $\mathcal{G}_{R'}$.

We have two games $\mathcal{G}_1$, $\mathcal{G}_2$. If there is a polynomial-time adversary $\mathcal{A}$ such that $\mathfrak{A}(\mathcal{G}_1) - \mathfrak{A}(\mathcal{G}_2) = \epsilon$, we can use $\mathcal{A}$ to construct a polynomial-time challenger $\mathcal{C}$ with advantage ϵ in breaking one of Assumptions 1–4.

Challenger $\mathcal{C}$ is given the tuple $(\mathcal{D}, T)$, which he uses to generate keys, ciphertexts and other parameters to simulate a game. T is either T_1 or T_2, but not known to $\mathcal{C}$, who simulates $\mathcal{G}_1$ if $T = T_1$, and $\mathcal{G}_2$ otherwise.

In the challenge phase, $\mathcal{A}$ sends two messages M_0, M_1 and an access structure he cannot satisfy to $\mathcal{C}$. $\mathcal{C}$ randomly selects $\beta \in \{0, 1\}$, encrypts M_β under the access structure and sends the encrypted M_β back to $\mathcal{A}$.

$\mathcal{A}$ guesses β with advantage $\Pr[\beta' = \beta] - \frac{1}{2}$. $\mathcal{A}$'s advantage in the game is $\mathfrak{A}(\mathcal{G}_1)$ if $T = T_1$, otherwise it is $\mathfrak{A}(\mathcal{G}_2)$. Now $\mathcal{C}$ can use $\mathcal{A}$ to obtain advantage $\mathfrak{A}(\mathcal{G}_1) - \mathfrak{A}(\mathcal{G}_2) = \epsilon$ in distinguish T_1 and T_2, and break one of Assumptions 1–4 with the advantage ϵ.

Proof 1: Proof overview of Lemma 2 to 6.

Proof. The boxed figure Game 1 shows the differences between $\mathcal{G}_R$ and $\mathcal{G}_{R'}$. We enhance the proof of $\mathcal{G}_R$ in [8] to prove that $\mathfrak{A}(\mathcal{G}_{R'})$ is negligible, even when $\mathcal{A}$ is more powerful. The major technique we use is dual system encryption which is a new proof technique based on the idea of semi-functional keys and ciphertexts [14]. The security game $\mathcal{G}_{R'}$ is identical to the generic game in box figure Game 1, except that the access structure $\mathbb{A}$ in the Challenge Phase uses no delegated attributes. We define following games to prove $\mathfrak{A}(\mathcal{G}_{R'})$ is negligible.

- $\mathcal{G}_{R''}$ Identical to $\mathcal{G}_{R'}$ except that the random oracle maps UID to elements in G_{p_1}, instead of G.
- $\mathcal{G}_0$ Identical to $\mathcal{G}_{R''}$, except that the ciphertext for M_β given to $\mathcal{A}$ is semi-functional.
- $\mathcal{G}_{j,1}$ M_β's ciphertext is semi-functional. The queried $\varpi_{\mathrm{UID},i}$ are type-2 semi-functional keys for the first $j-1$ UIDs, and a type-1 semi-functional key for the jth UID. Other $\varpi_{\mathrm{UID},i}$ are normal.
- $\mathcal{G}_{j,2}$ The ciphertext is a semi-functional encryption of a random message. The queried $\varpi_{\mathrm{UID},i}$ are type-2 semi-functional keys for the first j UIDs. Other $\varpi_{\mathrm{UID},i}$ are normal.
- $\mathcal{G}_{\boldsymbol{final}}$ The ciphertext is a semi-functional encrypted random message. All queried $\varpi_{\mathrm{UID},i}$ are type-2 semi-functional keys.

We first prove that the differences between $\mathfrak{A}(\mathcal{G}_{R'})$ and $\mathfrak{A}(\mathcal{G}_{R''})$, $\mathfrak{A}(\mathcal{G}_{R''})$ and $\mathfrak{A}(\mathcal{G}_0)$, $\mathfrak{A}(\mathcal{G}_{j-1,2})$ and $\mathfrak{A}(\mathcal{G}_{j,1})$, $\mathfrak{A}(\mathcal{G}_{j,1})$ and $\mathfrak{A}(\mathcal{G}_{j,2})$, $\mathfrak{A}(\mathcal{G}_{q,2})$ and $\mathfrak{A}(\mathcal{G}_{final})$ are all negligible. Since $\mathfrak{A}(\mathcal{G}_{final}) = 0$, we prove $\mathfrak{A}(\mathcal{G}_{R'})$ is negligible. Due to space limitation, we only show the proof of Lemma 6.

We give the overview of our proof for Lemma 2 to 6 in the box figure Proof. 1.

Lemma 2 *Let there exist a polynomial time algorithm $\mathcal{A}$ with $\mathfrak{A}(\mathcal{G}_{R'}) - \mathfrak{A}(\mathcal{G}_{R''}) = \epsilon$. We can then construct a polynomial time algorithm $\mathcal{C}$ with advantage ϵ in breaking Assumption 1.*

Lemma 3 *Let there exists a polynomial time algorithm $\mathcal{A}$ with $\mathfrak{A}(\mathcal{G}_{R''}) - \mathfrak{A}(\mathcal{G}_0) = \epsilon$. We can then construct a polynomial time algorithm $\mathcal{C}$ with advantage ϵ in breaking Assumption 1.*

Proof sketch. In the Setup Phase, $\mathcal{C}$ receives N, g_1, T. Phase 1 proceeds as in Game 1.

In the Challenge Phase, let B and $\bar{B}$ denote the subset of rows in A corresponding to attributes managed by corrupted authorities, and good authorities, respectively. $\mathcal{C}$ constructs semi-functional ciphertext for B and $\bar{B}$, exactly as in Lemma 8 of [8]. $\mathcal{C}$ sends the resulting ciphertext and shares λ_x, ω_x of $[A]_x \in B$ to $\mathcal{A}$.

Phase 2 proceeds as in Game 1. As Lemma 8 in [8] shows, the ciphertext is randomly distributed. $\mathcal{A}$ obtains all shares from corrupt authorities and some additional shares from good authorities in Phases 1 and 2. However, there is at least one row $[A]_i$ controlled by a good authority, without which the space spanned by the rows $[A]_j$ controlled by $\mathcal{A}$ cannot include $(1, 0, \ldots, 0)$, preventing him from recovering the secret s. $\mathcal{A}$ can derive ω_i if he controls all rows except for $[A]_i$ in an authorized set of $\mathbb{A}$. However, since shares of s and 0 are uncorrelated, $\mathcal{A}$ cannot derive λ_i from ω_i.

When $T \in G_{p_1}$, $\mathcal{C}$ simulates $\mathcal{G}_{R'}$, and if $T \in G$, $\mathcal{C}$ will simulate $\mathcal{G}_0$ with probability negligibly close to 1. We conclude that $\mathcal{C}$ can use $\mathcal{A}$ to obtain advantage negligibly close to 0 in breaking Assumption 1. □

Lemma 4 *Let there exist a polynomial time algorithm $\mathcal{A}$ such that $\mathfrak{A}(\mathcal{G}_{j-1,2}) - \mathfrak{A}(\mathcal{G}_{j,1}) = \epsilon$. Then we can construct a polynomial time algorithm $\mathcal{C}$ with advantage ϵ in breaking Assumption 1.*

Lemma 5 *Let there exist a polynomial time algorithm $\mathcal{A}$ such that $\mathfrak{A}(\mathcal{G}_{j,1}) - \mathfrak{A}(\mathcal{G}_{j,2}) = \epsilon$. Then we can construct a polynomial time algorithm $\mathcal{C}$ with advantage ϵ in breaking Assumption 3.*

Lemma 6 *Let there exist a polynomial time algorithm $\mathcal{A}$ such that $\mathfrak{A}(\mathcal{G}_{q,2}) - \mathfrak{A}(\mathcal{G}_{\text{final}}) = \epsilon$. Then we can construct a polynomial time algorithm $\mathcal{C}$ with advantage ϵ in breaking Assumption 4.*

Sketch of Lemma 6's proof: In the Setup phase, $\mathcal{C}$ receives N, g_1, g_2, g_3, g_1^a, $g_1^b g_3^b$, g_1^c, $g_1^{ac} g_3^d$, T. $\mathcal{C}$'s task is to determine whether T is a random value or $e(g_1, g_1)^{abc}$. $\mathcal{C}$ chooses random values α_i', $y_i' \in \mathbb{Z}_N$ and using the values it have been given, generates public keys of good authorities as $e(g_1, g_1)^{\alpha_i} = e(g_1^a, g_1^b g_3^b) e(g_1, g_1)^{\alpha_i'}$, $g_1^{\chi_i} = g_1^a g_1^{\chi_i'}$. Please note $e(g_1, g_3) = 1$.

Phase 1 proceeds as in Game 1. The warrants $\varpi_{\text{UID},i}$ are constructed as in Lemma 11 of [8].

In the Challenge Phase, $\mathcal{C}$ picks a bit β, and computes $C_M = M_\beta T$. When $T = e(g_1, g_1)^{abc}$, this ciphertext corresponds to encryption with the secret key $s = abc$. We construct the triples $(Y_{1,x}, Y_{2,x}, Y_{3,x})$ just as in [8].

Now, $\mathcal{C}$ must send to $\mathcal{A}$ shares λ_x, ω_x of $s = abc$ for attributes $\rho(x)$ managed by corrupted authorities. However, $\mathcal{C}$ does not know $s = abc$. As per the proof

in [8], $\mathcal{C}$ indirectly creates such λ_x and ω_x, choosing random vectors u_2, u_3 with 0 as their first element, and computing $\lambda_x = [A]_x \cdot u_2$ and $\omega_x = [A]_x \cdot u_3$ for corrupted authorities.

Phase 2 proceeds as in Game 1. As before, there is at least one row $[A]_i$ controlled by a good authority, without which the span of the rows $[A]_j$ controlled by $\mathcal{A}$ cannot include $(1, 0, \ldots, 0)$, preventing him from recovering the secret s. $\mathcal{A}$ can derive ω_i if he controls all rows except for $[A]_i$ in an authorized set of $\mathbb{A}$. However, since λ_i and ω_i are uncorrelated, $\mathcal{A}$ cannot infer one from the other.

If $T = e(g_1, g_1)^{abc}$, $\mathcal{C}$ simulates $\mathcal{G}_{q,2}$, and if T is a random value, $\mathcal{C}$ simulates $\mathcal{G}_{final}$. $\mathcal{C}$ can use $\mathcal{A}$ to obtain advantage ϵ in breaking assumption 4. □

From Lemma 2 to 6, we prove Theorem 1.

Lemma 7 *If $\mathcal{A}$ cannot satisfy access tree $\mathbb{T}$ in $\mathcal{G}_{R'}$, the probability of his obtaining $\mathbb{T}$'s root U-share is negligible.*

Proof. By Theorem 1, if $\mathcal{A}$ cannot satisfy access tree $\mathbb{T}$ in $\mathcal{G}_{R'}$, his advantage in $\mathfrak{A}(\mathcal{G}_{R'})$ is negligible. By Definition 7, the encryption system which $\mathcal{G}_{R'}$ is based on is hence secure. Hence, $\mathcal{A}$ only has negligible advantage in decrypting the ciphertext, or equivalently, in obtaining the U-share of $\mathbb{T}$'s root.

Proof that $\mathfrak{A}(\mathcal{G}_{L_n})$ is negligible We show that $\mathcal{A}$'s advantage in $\mathcal{G}_{L_1}$ is negligible, and proceed by induction. We define $\mathcal{G}_{L_1'}$ to prove $\mathfrak{A}(\mathcal{G}_{L_1})$ is negligible. In $\mathcal{G}_{L_1'}$, the access structure may include delegated attributes, and 1-level delegation is allowed. Unlike $\mathcal{G}_{L_1}$, $\mathcal{A}$ in $\mathcal{G}_{L_1'}$ gets secret shares of each attribute controlled by corrupted authorities, regardless it is regular or delegated.

Lemma 8 *Any polynomial time adversary $\mathcal{A}$ has only negligible advantage in $\mathcal{G}_{L_1'}$.*

Proof. We first define $\mathcal{G}_{L_1'}$ as follows. Setup and Phase 1 in $\mathcal{G}_{L_1'}$ are exactly as in boxed figure Game 1.

In the Challenge Phase, $\mathcal{A}$ creates a complete access structure $\mathbb{A}^*$, which includes the main access structure and all level-1 access sub-structures $\{\mathbb{A}_j\}$. $\mathcal{C}$ encrypts M_β under $\mathbb{A}^*$ and sends the resulting ciphertext to $\mathcal{A}$. $\mathcal{C}$ also sends shares λ_x, ω_x of attributes managed by corrupted authorities to $\mathcal{A}$. Phase 2 is identical to that in Game 1.

Since $\mathcal{A}$ cannot satisfy $\mathbb{A}^*$, he cannot satisfy its access tree $\mathbb{T}^*$. By construction, $\mathbb{T}^*$ uses 1-level delegation. Let $\mathbb{T}^0$ be the main access tree with access subtrees eliminated. $\mathbb{T}^0$ uses no delegation. $\mathcal{A}$ fails to satisfy $\mathbb{T}^*$ in two cases. In the first case, $\mathcal{A}$ is unable to satisfy a leaf associated with a regular attribute on $\mathbb{T}^0$, so that $\mathcal{A}$ still only has negligible advantage in game $\mathcal{G}_{R'}$ played with the challenge access tree $\mathbb{T}^0$. Hence, $\mathcal{A}$ only has negligible advantage in $\mathcal{G}_{L_1'}$. In the second case, $\mathcal{A}$ satisfies all regular attributes in $\mathbb{T}^0$, but fails to satisfy a leaf on some subtree $\mathbb{T}_\delta$ of $\mathbb{T}^*$. Let $\mathbb{T}_\delta$'s root be v_δ. By Lemma 7, $\mathcal{A}$'s advantage ϵ' in obtaining v_δ's U-share is negligible. If $\mathcal{A}$ cannot obtain v_δ's U-share, he cannot satisfy $\mathbb{T}^0$, and $\mathfrak{A}(\mathcal{G}_{R'}) = \epsilon$ is negligible with the challenge access tree $\mathbb{T}^0$.

Let $\mathbf{v}_\delta$ and $\overline{\mathbf{v}_\delta}$, respectively, be the events that $\mathcal{A}$ can and cannot obtain v_δ's U-share. We have $\Pr[\mathbf{v}_\delta] = \epsilon'$, $\Pr[\overline{\mathbf{v}_\delta}] = 1 - \epsilon'$ and $\Pr[\beta' = \beta \mid \mathbf{v}_\delta] \leq 1$. Now,

$$\begin{aligned}
\mathfrak{A}(\mathcal{G}_{L_1'}) &= \Pr[\beta' = \beta] - \frac{1}{2} \\
&\leq (\Pr[\beta' = \beta \mid \overline{\mathbf{v}_\delta}] \Pr[\overline{\mathbf{v}_\delta}] + \Pr[\beta' = \beta \mid \mathbf{v}_\delta] \Pr[\mathbf{v}_\delta]) - \frac{1}{2} \\
&\leq \left(\frac{1}{2} + \epsilon\right)(1 - \epsilon') + 1 \cdot \epsilon' - \frac{1}{2} \\
&= \frac{1}{2}\epsilon' + \epsilon - \epsilon\epsilon'
\end{aligned}$$

So that $\mathfrak{A}(\mathcal{G}_{L_1'})$ is negligible. The $\leq$ arises because $\mathcal{A}$ may fail to satisfy several subtrees simultaneously.

Lemma 9 *Any polynomial-time adversary has only negligible advantage in* $\mathcal{G}_{L_1}$.

Proof. $\mathcal{A}$ obtains more information in $\mathcal{G}_{L_1'}$ than in in $\mathcal{G}_{L_1}$, so that $\mathfrak{A}(\mathcal{G}_{L_1'}) \geq \mathfrak{A}(\mathcal{G}_{L_1})$.

Lemma 10 *Any polynomial-time adversary has only negligible advantage in* $\mathcal{G}_{L_n}$.

Proof. We first assume that $\mathcal{A}$ has only negligible advantage in $\mathcal{G}_{L_{n-1}}$. Setup and Phase 1 in $\mathcal{G}_{L_n}$ are exactly as in Game 1. The Challenge Phase proceeds as in Game 1. The challenge access structure $\mathbb{A}^*$ includes the main access structure and all access sub-structures $\{\mathbb{A}_j\}$. In addition to the ciphertext for M_β, $\mathcal{C}$ sends shares λ_x, ω_x of attributes managed by corrupted authorities to $\mathcal{A}$. Phase 2 proceeds as in Game 1. $\mathcal{A}$ outputs a guess β' of β.

Since $\mathcal{A}$ cannot satisfy $\mathbb{A}^*$, he cannot satisfy its access tree $\mathbb{T}^n$, which uses n-level delegation. Let $\mathbb{T}^{n-1}$ represent the first $n-1$ levels of delegation in $\mathbb{T}^n$, with the last delegation level eliminated. $\mathcal{A}$ fails to satisfy $\mathbb{T}^n$ in two cases. First, a leaf associated with a regular attribute on $\mathbb{T}^{n-1}$ is not satisfied, so that $\mathcal{A}$ only has a negligible advantage in game $\mathcal{G}_{L_{n-1}}$ with the challenge access tree $\mathbb{T}^{n-1}$. Hence, $\mathcal{A}$ only has a negligible advantage in $\mathcal{G}_{L_n}$. In the second case, $\mathcal{A}$ satisfies all regular attributes in $\mathbb{T}^{n-1}$, but fails to satisfy a leaf in some subtree $\mathbb{T}_\delta$ corresponding to a delegated attribute δ occurring as a leaf in $\mathbb{T}^{n-1}$. Let $\mathbb{T}_\delta$'s root be $v_\delta \in \mathbb{T}^{n-1}$.

By Lemma 7, $\mathcal{A}$ only has a negligible advantage ϵ' in obtaining v_δ's U-share that can be used to obtain $\mathbb{T}$'s root's U-share. When $\mathcal{A}$ cannot obtain v_δ's U-share, he cannot satisfy the access tree $\mathbb{T}^{n-1}$, and $\mathfrak{A}(\mathcal{G}_{L_{n-1}}) = \epsilon$ is negligible with the challenge access tree $\mathbb{T}^{n-1}$.

We have $\Pr[\mathbf{v}_\delta] = \epsilon'$, $\Pr[\overline{\mathbf{v}_\delta}] = 1 - \epsilon'$ and $\Pr[\beta' = \beta \mid \mathbf{v}_\delta] \leq 1$. Now

$$\begin{aligned}
\mathfrak{A}(\mathcal{G}_{L_n}) &= \Pr[\beta' = \beta] - \frac{1}{2} \\
&\leq (\Pr[\beta' = \beta \mid \overline{\mathbf{v}_\delta}] \Pr[\overline{\mathbf{v}_\delta}] + \Pr[\beta' = \beta \mid \mathbf{v}_\delta] \cdot \Pr[\mathbf{v}_\delta]) - \frac{1}{2} \\
&\leq \left(\frac{1}{2} + \epsilon\right)(1 - \epsilon') + 1 \cdot \epsilon' - \frac{1}{2} \\
&= \frac{1}{2}\epsilon' + \epsilon - \epsilon\epsilon'
\end{aligned}$$

So that $\mathfrak{A}(\mathcal{G}_{L_n})$ is negligible. Our final result follows.

Theorem 11 *If Assumptions 1, 2, 3 hold, the HM-ABE encryption scheme is secure against all polynomial time adversaries.*

7 Related Work

Fuzzy Identity-Based Encryption (IBE), the first attribute based encryption scheme, was proposed in [15]. Fuzzy IBE evolved from the IBE scheme in [16,17]. Being a precursor scheme, fuzzy IBE does not support very expressive access policies.

Schemes such as KP-ABE [2] and CP-ABE [1] are more expressive. In KP-ABE, each ciphertext is associated with a set of attributes. Each user's private key is associated with an access structure, which is an access tree where attributes are leaves and internal nodes are threshold gates. A user can decrypt a ciphertext when the ciphertext's attributes satisfy the user's access structure. In [1], KP-ABE is transformed into CP-ABE. User private key is associated with attributes and each ciphertext is associated with an access structure. A user can decrypt a ciphertext if the attributes that the user has can satisfy the ciphertext's access structure.

Predicate encryption [18] is an ABE scheme achieving policy hiding. A user evaluates whether he satisfies a ciphertext's access policy, by decrypting the ciphertext. Whether this decryption succeeds or fails, he does not learn what the access policy is. However, this scheme does not allow expressive access policies.

The first multi-authority scheme, proposed in [9], recognizes k attribute authorities (AA), each maintaining a disjoint attribute set. This scheme still requires a trusted CA to combine secret shares to prevent collusion attacks. Unless the CA or all AAs are compromised, the system is secure. Access structures in this scheme are not very expressive. Chase and Chow later proposed an improved scheme in [10] to remove the requirement of the CA.

This M-ABE scheme in [8] allows each ciphertext to be associated with an LSSS-based access structure containing attributes from different authorities. Each user has a global identifier. An authority generates a private key for an (attribute, identity) pair. Users obtain private keys matching their attributes and identities from authorities. The scheme is secure unless all authorities are

corrupted. Dual system encryption [14] [19] is used to prove the security of this system.

Liu et al. improved DABE by removing the random oracle [13]. However, their scheme introduces some additional trusted CAs involved in key generation.

LSSS [12] is used in [2, 7] to express access policies. LSSS-based access policies generalize previous access policies, and are regarded as the most expressive. In [20], Frikken et al. proposed an ABE scheme with Hidden Policies and Hidden Credentials, which hides encryptors' policies from decryptors, and hides decryptors' credentials from encryptors.

Yu et al. introduced proxy-based attribute revocation scheme in [21]. The authority uses proxy re-encryption [22] to delegate most work of attribute revocation to a semi-trusted proxy server.

8 Conclusion

We have presented HM-ABE, a new hierarchical distributed attribute-based encryption scheme, which permits policy delegation. Recursive policy delegation is supported. Delegation allows more precise access policy to be built, and permits authorities to hide their sensitive attributes by designating them as hidden attributes, and not publishing the public keys. We prove our scheme is secure.

References

1. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy. (2007) 321–334
2. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: CCS. (2006) 89–98
3. Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures. In: CCS. (2007) 195–203
4. Goyal, V., Jain, A., Pandey, O., Sahai, A.: Bounded ciphertext policy attribute based encryption. In: ICALP. (2008) 579–591
5. Okamoto, T., Takashima, K.: Fully secure functional encryption with general relations from the decisional linear assumption. In: CRYPTO. (2010) 191–208
6. Lewko, A.B., Waters, B.: Unbounded hibe and attribute-based encryption. In: EUROCRYPT. (2011) 547–567
7. Waters, B.: Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In: PKC. (2011) 53–70
8. Lewko, A.B., Waters, B.: Decentralizing attribute-based encryption. In: EUROCRYPT. (2011) 568–588
9. Chase, M.: Multi-authority attribute based encryption. In: TCC. (2007) 515–534
10. Chase, M., Chow, S.S.M.: Improving privacy and security in multi-authority attribute-based encryption. In: CCS. (2009) 121–130
11. Lin, H., Cao, Z., Liang, X., Shao, J.: Secure threshold multi authority attribute based encryption without a central authority. Inf. Sci. **180** (2010) 2618–2632
12. Beimel, A.: Phd thesis, Israel Institute of Technology, Technion, Haifa, Israel (1996)

13. Liu, Z., Cao, Z., Huang, Q., Wong, D.S., Yuen, T.H.: Fully secure multi-authority ciphertext-policy attribute-based encryption without random oracles. In: ESORICS. (2011) 278–297
14. Waters, B.: Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In: CRYPTO. (2009) 619–636
15. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: EUROCRYPT. (2005) 457–473
16. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. SIAM J. Comput. **32** (2003) 586–615
17. Shamir, A.: Identity-based cryptosystems and signature schemes. In: CRYPTO. (1984) 47–53
18. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: EUROCRYPT. (2008) 146–162
19. Lewko, A.B., Waters, B.: New techniques for dual system encryption and fully secure hibe with short ciphertexts. In: TCC. (2010) 455–479
20. Frikken, K., Atallah, M., Li, J.: Attribute-based access control with hidden policies and hidden credentials. IEEE Trans. Comput. **55** (2006) 1259–1270
21. Yu, S., Wang, C., Ren, K., Lou, W.: Attribute based data sharing with attribute revocation. In: ASIACCS. (2010) 261–270
22. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Trans. Inf. Syst. Secur. **9** (2006) 1–30