White-Box Traceable Attribute-Based Encryption with Hidden Policies and Outsourced Decryption

1st Dominik Ziegler
Know-Center GmbH
Graz, Austria
dominik.ziegler@tugraz.at

2nd Alexander Marsalek
IAIK, Graz University of Technology
Graz, Austria
alexander.marsalek@iaik.tugraz.at

3rd Gerald Palfinger
A-SIT
Vienna, Austria
gerald.palfinger@a-sit.at

***Abstract*—We address three practical problems of *Attribute-Based Encryption* (ABE) in this paper: performance, accountability and privacy. To do so, we present a novel *Ciphertext-Policy Attribute-Based Encryption* (CP-ABE) approach, which combines white-box accountability, hidden policies and outsourced decryption. In contrast to existing schemes, the proposed construction is not only more flexible, but also efficient enough to be used in more resource-constrained environments. This is because our construction is designed around efficient Type III bilinear maps and relies on a dedicated proxy to perform computationally expensive operations. Furthermore, we provide an implementation of the proposed design. The conducted evaluation demonstrates the practicality of the approach under realistic assumptions.**

***Index Terms*—access control, accountability, revocation, attribute-based encryption, hidden policies, privacy**

I. Introduction

Attribute-Based Encryption (ABE) provides fine-grained access control on encrypted data. It achieves this by either attaching an access policy to a user's secret key (*Key-Policy Attribute-Based Encryption* (KP-ABE) [1]) or to the ciphertext itself (*Ciphertext-Policy Attribute-Based Encryption* (CP-ABE) [2]). Only those users who hold the necessary attributes to fulfil an access structure can decrypt a ciphertext. This unique construction allows data owners to encrypt data once without knowing the recipients in advance. As a result, ABE can be especially helpful in highly dynamic environments, e.g. where users' permissions may change daily or where parties regularly join or leave the system.

Problem statement. Although ABE can provide access control on a fine-granular basis, it still suffers from significant issues which hinder wide adoption. For example, while ABE can reduce computational burden when encrypting for multiple (unknown) parties, it can introduce a *significant overhead* for decryption operations. This is because it heavily relies on so-called bilinear pairings, which allow mapping elements from two cryptographic groups to a third. In fact, bilinear pairings have been found to be two to three times computationally more expensive than scalar multiplications [3]. As a result, ABE is typically not suitable for resource-constrained devices. Another issue ABE entails is *privacy of users*. Indeed, ciphertexts do not reveal any information about the plaintext. Their attached policy or attributes, however, may reveal sensitive information about the recipient or originator. Let us consider a data-sharing system for medical data, for example. Any third party storage provider, which hosts a ciphertext can also learn the access policy or attributes. Depending on the attached information (e.g. *role="patient" AND condition="severe"*), unauthorised third parties might infer information about the recipients health. Finally, malicious users can reveal their key to unauthorised parties or abuse their credentials (e.g. before leaving a company). However, in ABE-based systems, attributes and, thus, the privilege of a key are usually shared with multiple users. Holding malicious users *accountable* for their actions can, therefore, be a challenging endeavour. In turn, central authorities can, in theory, issue arbitrary well-formed decryption keys or frame other users in the system without being caught.

Our approach. While most of the aforementioned issues of ABE systems have been addressed individually in previous work, no solution addresses them in a holistic and practical approach. More specifically, we believe that *performance*, *privacy* and *accountability* do not necessarily have to be mutually exclusive in ABE-based systems. As a result, we present a novel CP-ABE approach, which addresses these limitations from four angles: First, we leverage features of cloud computing and allow clients to outsource expensive decryption operations to more powerful proxies, as shown by Green et al. [4]. This approach leaves clients only with a single exponentiation to decrypt a ciphertext. Second, we design our approach around so-called *Type-III* or *asymmetric pairings* and prime order groups. In contrast to Type-I or Type-II pairings, pairings of this type offer good performance and flexibility for high-security parameters [5]. Third, we deal with privacy issues by allowing clients to hide access policies from third parties. On each outsourced decryption attempt, clients can hide and randomise the embedded access policy of ciphertexts without revealing any information to the proxy. To do so, we leverage *Dual Pairing Vector Spaces* (DPVS) [6], a technique to achieve orthogonality in prime-order groups. These allow us to use the cancelling properties of bilinear maps, otherwise only available in so-called composite order groups. Most notable, the security of composite order groups relies on the hardness of factoring the group order. As a result, protocols using composite order groups require large group

978-1-XXXX-XXXX-X/20/$31.00 ©2021 IEEE

orders, which, in turn, results in inefficient pairing operations. Finally, we provide *white-box traceability* [7] of malicious users and the central authority. This means that secret keys leaked by either a malicious user or an authority can be traced back to the originator by an auditor.

Our contributions. Our contributions are in summary as follows.

- We present a novel CP-ABE approach, which combines outsourced decryption, hidden policies and white-box accountability.
- We leverage Type-III pairings and DPVS to provide the best tradeoff between performance and flexibility.
- Our scheme allows holding malicious users and authorities accountable for their actions in the white-box model.
- We provide an efficient revocation strategy to cope with requirements imposed by highly dynamic environments.
- We provide a security proof under the *Symmetric External Diffie-Hellman* (SXDH) assumption and prove the accountability property of the system under the *q-Strong Diffie-Hellman Assumption* (q-SDH) and *Discrete Logarithm Problem* (DLP) assumptions.
- Finally, we provide an implementation of our solution and evaluate it, both theoretically and practically.

The rest of this paper is organised as follows. First, we define the terminology and provide necessary background information and assumptions. Second, we define the algorithms for an accountable CP-ABE system with hidden policies and outsourced decryption. We, furthermore, provide the security model and give a concrete construction of the proposed approach. In Section IV, we prove the security of our scheme under the SXDH assumption. In addition, we prove the accountability property under the q-SDH and DLP assumptions. Next, we provide an implementation of our approach and compare it with existing work. Finally, we summarise existing work and discuss the relation to our approach. We conclude this paper in Section VII.

II. Preliminaries

We define the notation and assumptions used throughout this document.

A. (Asymmetric) Bilinear Groups of Prime Order

We adapt the notation of Boneh and Shacham [8]. Let $\mathbb{G}_1$, $\mathbb{G}_2$ be (two) multiplicative cyclic groups of prime order p. g_1 and g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a computable map with the following two properties:

- Bilinearity: for all $u \in \mathbb{G}_1$, $v \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p^*$, $e(u^a, v^b) = e(u, v)^{ab}$.
- Non-degeneracy: $e(g_1, g_2) \neq 1$.

In asymmetric pairings $\mathbb{G}_1 \neq \mathbb{G}_2$. We define vectors of group elements: Let $\boldsymbol{v} = (v_1, \ldots, v_n)$ be a vector and $g_i \in \mathbb{G}_i$ a generator. $g_i^{\boldsymbol{v}}$ describes the tuple of elements: $g_i^{\boldsymbol{v}} = (g_i^{v_1}, g_i^{v_2}, \ldots, g_i^{v_n})$. For any element $a \in \mathbb{Z}_p^*$ we define $g_i^{a\boldsymbol{v}} = (g_i^{av_1}, \ldots, g_i^{av_n})$. Likewise, for two vectors $\boldsymbol{v}, \boldsymbol{w}$ the following applies: $g_i^{\boldsymbol{v}+\boldsymbol{w}} = (g_i^{v_1+w_1}, \ldots, g_i^{v_n+w_n})$. The componentwise pairing is defined as follows: $e_n(g_1^{\boldsymbol{v}}, g_2^{\boldsymbol{w}}) = \prod_{i=1}^{n} e(g_1^{v_i}, g_2^{w_i}) = e(g_1, g_2)^{\boldsymbol{v}\cdot\boldsymbol{w}}$.

B. Access structure & Linear Secret Sharing Scheme

Definition 1. *(Access Structure). Let $\mathcal{S} = \{a_1, a_2, \ldots, a_n\}$ be a set of attributes. An access structure is a collection $\mathbb{A}$ of a non-empty subset of $\{a_1, a_2, \ldots, a_n\}$, i.e. $\mathbb{A} \subseteq 2^{\{a_1,\ldots,a_n\}} \setminus \{\emptyset\}$. We say that $\mathbb{A}$ is monotone, if $\forall B, C : if B \in \mathbb{A}$ and $B \subset C$ then $C \in \mathbb{A}$. We refer to set sets in $\mathbb{A}$ as the* authorised set*, and the sets not in $\mathbb{A}$ as the* unauthorised sets*. We henceforth refer to an access structure as a monotone access structure.*

Definition 2. *(*Linear Secret Sharing Scheme *(LSSS)). We adapt the definition from Sahai et al. [9]. A LSSS policy $\Pi = (\mathbb{A}, \rho)$ is a pair, where $\mathbb{A}$ is a $n \times l$ matrix over the base field $\mathbb{F}$. ρ is the map from [n] to the attribute space Σ. An attribute set $\mathcal{S} \in \Sigma$ satisfies an access structure $(\mathbb{A}, \rho)$ if $1 = (1, 0, 0, \ldots, 0) \in \mathbb{F}^l$ is contained in $Span_{\mathbb{F}}(\mathbb{A}_i : \rho(i) \in \mathcal{S})$, where $\mathbb{A}_i$ is the i^{th} row of $\mathbb{A}$.*

C. Dual Pairing Vector Spaces

Okamoto and Takashima [6] introduced the concept *Dual Pairing Vector Spaces* (DPVS). It uses two *dual orthonormal* bases $\mathbb{B} = (\boldsymbol{b_1}, \ldots, \boldsymbol{b_n})$, $\mathbb{B}^* = (\boldsymbol{b_1^*}, \ldots, \boldsymbol{b_n^*})$ in $\mathbb{Z}_n^*$. The system has the following properties: $\boldsymbol{b_i} \cdot \boldsymbol{b_j^*} = 0 \pmod{p}$ if $i \neq j$, $\boldsymbol{b_i} \cdot \boldsymbol{b_i^*} = \psi$ if $i = j$, where $\psi \xleftarrow{R} \mathbb{Z}_p^*$. Likewise, $e_n(g_1^{\boldsymbol{b_i}}, g_2^{\boldsymbol{b_j^*}}) = 1$, whenever $i \neq j$.

D. Symmetric External Diffie-Hellman (SXDH) Assumption

Given a bilinear group G of prime order p, the SXDH [10] assumption states that the *Decisional Diffie–Hellman* (DDH) problems are intractable in $\mathbb{G}_1$ and $\mathbb{G}_2$. Formally, we define the DDH problem for a group as follows:

Definition 3. *Given $G = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ and generators $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$ and elements $a, b, c \xleftarrow{R} \mathbb{Z}_p^*$. Let:*

$$\boldsymbol{D} = \langle G, g_1, g_2, g_1^a, g_2^b \rangle$$

Given a distribution $\boldsymbol{D}$, we say that any polynomial time algorithm $\mathcal{B}$ that outputs $\{0, 1\}$ has an advantage ϵ if:

$$\left| Pr[\mathcal{B}(\boldsymbol{D}, T = g_1^{ab}) = 1] - Pr[\mathcal{B}(\boldsymbol{D}, T = g_1^{ab+c}) = 1] \right| \geq \epsilon$$

We henceforth refer to the above definition $DDH1$. For $DDH2$ the roles of $\mathbb{G}_1$ and $\mathbb{G}_2$ are reversed.

E. (Asymmetric) Subspace Assumption

Given a bilinear group G of prime order p and an element $g^{\boldsymbol{v}}$, the *decisional subspace assumption* [11] states that there does not exist any efficient polynomial time algorithm, which can distinguish, whether $\boldsymbol{v}$ is chosen randomly from a span $\boldsymbol{b_1^* b_2^*}$ or a larger span $\boldsymbol{b_1^* b_2^* b_3^*}$.

Formally, we describe the SXDH assumption in $\mathbb{G}_1$ as follows:

Definition 4. *Given $G = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$, $(\mathbb{B}, \mathbb{B}^*) \xleftarrow{R} Dual(\mathbb{Z}_n^*)$ and elements $\tau_1, \tau_2, \mu_1, \mu_2 \xleftarrow{R} \mathbb{Z}_p^*$. Let:*

$$U_1 = g_2^{\mu_1 b_1^* + \mu_2 b_{k+1}^*}, U_2 = g_2^{\mu_1 b_2^* + \mu_2 b_{k+2}^*}, \dots, U_k = g_2^{\mu_1 b_k^* + \mu_2 b_{2k}^*},$$
$$V_1 = g_1^{\tau_1 b_1}, V_2 = g_1^{\tau_1 b_2}, \dots, V_k = g_1^{\tau_1 b_k}$$
$$W_1 = g_1^{\tau_1 b_1 + \tau_2 b_{k+1}}, W_2 = g_1^{\tau_1 b_2 + \tau_2 b_{k+2}}, \dots, W_k = g_1^{\tau_1 b_k \tau_2 b_{2k}}$$
$$\boldsymbol{D} = \langle G, g_2^{b_1 *}, \dots, g_2^{b_k *}, g_2^{b_{2k+1}}, \dots, g_2^{b_n^*}, g_1^{b_1}, \dots, g_1^{b_n},$$
$$U_1, \dots, U_k, \mu_2 \rangle$$

Above definition is analogous to the subspace assumption given in [10]. The subspace assumption in $\mathbb{G}_2$ is identical, with the roles of $\mathbb{G}_1$ and $\mathbb{G}_2$ being reversed. A polynomial-time algorithm has an advantage ϵ in solving the subspace problem, if:

$$\Big| Pr[\mathcal{B}(\boldsymbol{D}, T = V_1, \dots, V_k) = 1] - Pr[\mathcal{B}(\boldsymbol{D}, T = W_1, \dots, W_k) = 1] \Big| \geq \epsilon$$

We henceforth refer to DS1 and DS2 as the decisional subspace *assumptions in $\mathbb{G}_1$ and $\mathbb{G}_2$.*

F. q-Strong Diffie-Hellman Assumption

Given a randomly chosen element $x \in \mathbb{Z}_p$, a random generator $g_2 \in \mathbb{G}_2$, and a distribution $(g_1, g_2, g_2^x, g_2^{x^2}, \dots, g_2^{x^q}) \in \mathbb{G}_1 \times \mathbb{G}_2^{q+1}$, the q-Strong Diffie-Hellman Assumption *(q-SDH)* assumption [12, 13] states that there is no polynomial time algorithm, which can compute the tuple $(c, g_1^{1/(x+c)})$ efficiently. Formally, we define the q-SDH assumption as follows:

Definition 5. *Given $G = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$, and $x \xleftarrow{R} \mathbb{Z}_p$, let:*

$$\boldsymbol{D} = \langle (g_1, g_1^x, g_1^{x^2}, \dots, g_1^{x^q}, g_2, g_2^x) \in \mathbb{G}_1^{q+1} \times \mathbb{G}_2^2 \rangle$$

A polynomial-time algorithm has an advantage ϵ in solving the q-SDH problem, if:

$$\Big| Pr[\mathcal{B}(\boldsymbol{D}, T = (c, g_1^{1/(x+c)})) = 1] \Big| \geq \epsilon$$

III. System Definition

We, begin by defining the necessary algorithms for a white-box accountable CP-ABE scheme with outsourced decryption and hidden policies. Next, we describe the security model and present a concrete implementation of the proposed approach.

A. Algorithms

A white-box accountable CP-ABE scheme with outsourced decryption and hidden policies consists of the following algorithms:

Setup$(1^\lambda, \Sigma) \rightarrow MPK, MSK$. The *Setup* algorithm generates necessary system parameters. It takes as input the security parameter λ and the attribute space Σ. It outputs the master public key MPK, the master secret key MSK and the set of attribute group keys $\mathcal{K}$ associated with Σ.

KeyGen$(MPK, MSK, \mathcal{S}, ID) \rightarrow SK_\mathcal{S}$. The *KeyGen* algorithm generates a secret key for a given set of attributes and a user ID. It takes as input the master secret key MSK, a set of (user) attributes $\mathcal{S}$ and a user id ID. It returns the secret key $SK_\mathcal{S}$ associated with $\mathcal{S}$ and ID.

Encrypt$(M, \Pi, MPK) \rightarrow \overline{CT}$. The *Encrypt* algorithm encrypts a message under a given access structure. It takes as input a message M, a LSSS $\Pi = (\mathbb{A}, \rho)$ and the master public key MPK. It returns the initial ciphertext $\overline{CT}$.

ReEncrypt$(\overline{CT}, MPK, \mathcal{K}) \rightarrow CT$. The *ReEncrypt* embeds the attribute group keys $\mathcal{K}$ in the ciphertext. Furthermore, it adds random group elements to the ciphertext parts. It takes as input the initial ciphertext $\overline{CT}$, the master public key MPK and a set of attribute group keys $\mathcal{K}$. It returns the ciphertext CT.

GenTK$(MPK, SK_\mathcal{S}, k, CT, \mathcal{K}_I) \rightarrow TK, RK_\mathcal{S}$. The *GenTK* algorithm generates a ciphertext-specific transformation key, where any attribute information is hidden. It takes as input the master public key MPK, a secret key $SK_\mathcal{S}$ and a re-encrypted ciphertext CT. It returns the transformation key TK and a retrieving number value $RK_\mathcal{S}$.

GenCT$(CT, SK_\mathcal{S}) \rightarrow CT'$. The *GenCT* algorithm generates a ciphertext by combining user's secret attribute information. It takes as inoput a reencrypted ciphertext CT and a $SK_\mathcal{S}$. It returns a temporary ciphertext CT'.

OutsourcedDecrypt$(MPK, TK, CT') \rightarrow CT''$. The *OutsourcedDecrypt* algortihm transforms a ciphertext by performing expensive pairing operations. It takes as input the master public key MPK, a transformation key TK and a transformed ciphertext CT'. It returns the pairing free ciphertext CT''.

Decrypt$(MPK, CT, CT'', RK_\mathcal{S}) \rightarrow M$. The *Decrypt* algorithm recovers the plain text message. It takes as input the master public key MPK, the re-encrypted ciphertext CT, the pairing free ciphertext CT'' and the retrieving number value $RK_\mathcal{S}$. It outputs the plaintext message M.

Trace$(SK_\mathcal{S}) \rightarrow \perp$ *or (ID, k).* The *Trace* algorithm indicates whether a user with ID or an authority was dishonest. It takes as input a suspected $SK_\mathcal{S}$. It outputs $\perp$ if the key is not well-formed. Otherwise, it outputs the tuple (ID, k).

Revoke(ID, $\mathcal{S}_R$) The *Revoke* algorithm revokes a (sub-) set of attribute for a given user. It takes as input a user ID and a set of to be revoked attributes $\mathcal{S}_R$. It removes the user with ID from all the attribute groups in $\mathcal{S}_R$ and invalidates affected ciphertexts. It, furthermore, generates new group keys for all affected attribute-groups.

B. Security Model

We can now formally describe the security requirements of a of a white-box accountable CP-ABE scheme with hidden policies, revocation and outsourced decryption. Informally, *security* means that an adversary does not learn anything about the plaintext. *Hidden policies* means that a third party, performing outsourced decryption, does not learn the access policy. *White-box accountability* entails that shared keys can be traced back to the originator.

The *security* of a of a white-box accountable CP-ABE scheme with hidden policies, revocation and outsourced decryption is described by the following game:

Setup. First, the challenger C runs the ***Setup*** algorithm. It generates MPK and MSK and sends MPK to the adversary $\mathcal{A}$.

Phase 1. The adversary $\mathcal{A}$ queries $q1$ keys for attribute sets $\mathcal{S}_1, ... \mathcal{S}_{q1}$.
Challenge. The adversary generates two equal length messages M_0 and M_1. Furthermore, it generates two equal length access structures $\mathbb{A}_0^*$, $\mathbb{A}_1^*$. They may not be satisfied by any previously queried attribute set $\mathcal{S}_i$. C flips a coin $\beta \in \{0, 1\}$ and encrypts M_β under the challenge access structure $\mathbb{A}_\beta^*$. It sends the ciphertext CT to $\mathcal{A}$.
Phase 2. $\mathcal{A}$ runs key generation and decryption queries. To do so, it queries keys for attribute sets $\mathcal{S}_{q1+1}, ..., \mathcal{S}_q$. However, they may not satisfy previously generated access structures $\mathbb{A}^*$.
Guess. The adversary outputs a guess β' for β.

The winning advantage of an adversary in this game is defined to be $Pr[\beta = \beta'] - \frac{1}{2}$.

Definition 6. *A white-box accountable ABE scheme with hidden policies, revocation and outsourced decryption is* Chosen-Plaintext Attack *(CPA) secure and provides hidden policies if the advantage over all* Probabilistic Polynomial-Time *(PPT) algorithms in the aforementioned game is negligible.*

The *white-box* accountability property can be described by two games: A *dishonest authority game* and a *dishonest user game*.

The *dishonest authority game* assumes that a malicious authority frames other users. It is described as follows:
Setup. First, the adversary $\mathcal{A}$ executes the ***Setup*** algorithm and generates MPK and MSK. Then, it sends the generated parameters to the challenger C. C aborts, if the parameters are not well-formed.
Key Generation. C commits to a value $k \xleftarrow{R} \mathbb{Z}_p^*$. Now $\mathcal{A}$ runs the *KeyGen* algorithm returns the generated key $SK_\mathcal{S}$ to C. C aborts if $SK_\mathcal{S}$ is not well-formed.
Key Forgery. The adversary $\mathcal{A}$ generates a secret key $SK'_{\mathcal{S}'}$. C aborts if $SK_\mathcal{S}$ is not well-formed.

The *dishonest user game* consists of two parts: The first game, describes a dishonest user which manages to generate a $SK_\mathcal{S}$ for a different ID.
Setup. The challenger C runs the ***Setup*** algorithm and generates MPK and MSK. It sends the MPK to the adversary $\mathcal{A}$.
Key Query. The adversary $\mathcal{A}$ queries q keys for a set of IDs and attributes $(ID_i, \mathcal{S}_i)$. For every tuple, related to $\{c_i\}_{i\in[q]}$, C runs the *KeyGen* algorithm and returns the generated key $SK_{\mathcal{S}_i}$ to $\mathcal{A}$.
Key Forgery. $\mathcal{A}$ generates a secret key $SK_\mathcal{S}$ related to (ID, c). If (ID, c) has not been queried before and $SK_\mathcal{S}$ is well-formed, $\mathcal{A}$ wins the game.

The second game, describes a dishonest user which manages to tamper with the ID.
Setup. The challenger C runs the ***Setup*** algorithm and generates MPK and MSK. It sends the MPK to the adversary $\mathcal{A}$.
Key Query. The adversary $\mathcal{A}$ queries q keys for a set of IDs and attributes $(ID_i, \mathcal{S}_i)$. For every tuple, related to $\{c_i, k_i\}_{i\in[q]}$, where k_i is bound to ID, C runs the *KeyGen* algorithm and returns the generated key $SK_{\mathcal{S}_i}$ to $\mathcal{A}$.
Key Forgery. $\mathcal{A}$ generates a secret key $SK_\mathcal{S}$ related to (ID, c, k). If (ID, c) has not been queried before, k has not been seen before, and $SK_\mathcal{S}$ is well-formed, $\mathcal{A}$ wins the game.

Definition 7. *A white-box accountable ABE scheme with hidden policies, revocation and outsourced decryption fulfils the white-box accountability property if the advantages over all PPT algorithms in the games mentioned above is negligible.*

C. Scheme Construction

In this section, we propose a novel CP-ABE scheme based on (but not limited to) Ziegler and Marsalek's [14] and Li et al.'s [15] construction. The enhanced scheme provides traceability of malicious users while offering hidden policies and outsourced decryption.
Setup$(1^\lambda, \Sigma) \rightarrow MPK, MSK$. The algorithm performs the following steps:

1. Let $G = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$
2. Sample dual orthonormal bases $(\mathbb{D}, \mathbb{D}^*) \xleftarrow{R} Dual(\mathbb{Z}_p^4)$
3. Let $\boldsymbol{d_1}, ..., \boldsymbol{d_4}$ denote elements of $\mathbb{D}$
4. Let $\boldsymbol{d_1^*}, ..., \boldsymbol{d_4^*}$ denote elements of $\mathbb{D}^*$
5. Choose $g_1, f_1, ..., f_U \xleftarrow{R} \mathbb{G}_1$, $g_2 \xleftarrow{R} \mathbb{G}_2$
6. Choose $a, \alpha \xleftarrow{R} \mathbb{Z}_p^*$.
7. Choose group keys $\kappa_{\lambda_1} ... \kappa_{\lambda_U} \xleftarrow{R} \mathbb{Z}_p^*$ for each $U_i \in \mathcal{U}$
8. Choose $\mathbb{H}_T : \mathbb{G}_T \rightarrow \mathbb{Z}_p^*$
9. Output the master public key MPK, the master secret key MSK and the group keys $\mathcal{K}$:

$$MPK : \langle G, g_1^{\boldsymbol{d_1}}, g_2^{\boldsymbol{d_1^*}}, g_2^{\boldsymbol{d_2^*}}, f_1^{\boldsymbol{d_1}}, ..., f_U^{\boldsymbol{d_1}}, g_1^{a\boldsymbol{d_1}},$$
$$y = e(g_1, g_2)^{\alpha \boldsymbol{d_1} \cdot \boldsymbol{d_1^*}}, y_1 = e(g_1, g_2)^{\boldsymbol{d_2} \cdot \boldsymbol{d_2^*}},$$
$$X = g_2^{x\boldsymbol{d_1^*}}, Y = g_2^{y\boldsymbol{d_1^*}} \rangle$$
$$MSK = \langle g_1^{\alpha \boldsymbol{d_1}} \rangle, \mathcal{K} = \{\kappa_{\lambda_i}\}$$

KeyGen$(MPK, MSK, \mathcal{S}, ID) \rightarrow SK_\mathcal{S}$. First, the algorithm interacts with user ID. Then, the user chooses $k \xleftarrow{R} \mathbb{Z}_p^*$ and calculates the commitment value $R = g_1^{ak\boldsymbol{d_1}}$. It sends R and a proof of knowledge of k without revealing the value of k. Finally, the algorithm chooses $t \xleftarrow{R} \mathbb{Z}_p^*$ and outputs:

$$SK_\mathcal{S} = \langle K_1, K_2, K_3, K_4, \{K_x\}_{x\in\mathcal{S}}, T_1, T_2, T_3 \rangle$$
$$= \langle g_1^{\frac{\alpha \boldsymbol{d_1}}{(x+ID+yd)}} \cdot g_1^{atk\boldsymbol{d_1}}, g_2^{t\boldsymbol{d_1^*}}, g_2^{xt\boldsymbol{d_1^*}}, g_2^{yt\boldsymbol{d_1^*}},$$
$$\{f_x^{t(x+ID+yd)\boldsymbol{d_1}}\}_{x_\epsilon\mathcal{S}}, ID \rangle$$

Encrypt$(M, \Pi, MPK) \rightarrow \overline{CT}$. First, the algorithm generates a secret $s \xleftarrow{R} \mathbb{Z}_p^*$. Next, it calculates a share vector $\vec{v} = (s, v_2, ... v_n) \in \mathbb{Z}_p^*$. Then, it computes $\lambda_i = \vec{v} \cdot \mathbb{A}_i$ and generates $r_1, ..., r_l \xleftarrow{R} \mathbb{Z}_p^*$. It outputs:

$$\overline{CT} = \langle C = M \cdot y^s, \overline{C'} = g_2^{s\boldsymbol{d_1^*}}, C_2 = g_2^{xs\boldsymbol{d_1^*}}, C_3 = g_2^{ys\boldsymbol{d_1^*}},$$
$$\{\overline{C_i} = g_1^{a\lambda_i \boldsymbol{d_1}} \cdot f_{\rho(i)}^{-r_i \boldsymbol{d_1}}, \overline{D_i} = g_2^{r_i \boldsymbol{d_1^*}}\} \rangle$$

ReEncrypt$(\overline{CT}, MPK, \mathcal{K}) \rightarrow CT$. The algorithm first randomly chooses $\eta_1, ... \eta_l$, $\delta \xleftarrow{R} \mathbb{Z}_p^*$. It outputs the final ciphertext:

$$CT = \langle C, \tilde{C} = g_1^{\mathbb{H}_T(M)}, C' = \overline{C'} \cdot g_2^{\delta \boldsymbol{d_2^*}}, C_2, C_3, \{C_i = \{\overline{C_i} \cdot g_1^{(\eta_i + \kappa_{t_i})\boldsymbol{d_2}}, \overline{D_i}\}\rangle$$

GenTK*(MPK, SK_S, k, CT, $\mathcal{K}_I$)* $\rightarrow$ *TK, RK_S*. First, the algorithm computes a set $\{\omega_i \in \mathbb{Z}_p^*\}$, such that $\sum_{i \in I} \omega_i \lambda_i = s$, where $I = \{i : \rho(i) \in \mathcal{S}\}$. Next, it chooses $z, d \xleftarrow{R} \mathbb{Z}_p^*$ and calculates:

$$TK = \langle K_1' = K_1 \cdot g_1^{z\boldsymbol{d_2}}, K_2' = (K_2 \cdot g_2^{d\boldsymbol{d_2^*}})^{T_1} \cdot K_3 \cdot K_4^{T_3}, \{(K_\rho(i), \omega_i)\}_{i \in I}, I\rangle$$

$$RK_S = z\delta ID - dkID\big(\sum_{i \in I} \eta_i + \sum_{i \in I} \kappa_{t_i}\big)$$

GenCT*(CT, SK_S)* $\rightarrow$ *CT'*. The algorithm outputs the transformed ciphertext:

$$CT' = \langle C, \tilde{C}, C' = (C')^{T_1} C_2 C_3^{T_3}, \{C_i, \overline{D_i}\}\rangle$$

OutsourcedDecrypt*(MPK, TK, CT')* $\rightarrow$ *CT''*. The algorithm computes:

$$\begin{aligned} T_0 &= e_n(K_1', C') \\ &= e_n(K_1 \cdot g_1^{z\boldsymbol{d_2}}, (g_2^{s\boldsymbol{d_1^*}} g_2^{\delta \boldsymbol{d_2^*}})^{ID} g_2^{xs\boldsymbol{d_1^*}} g_2^{ysd\boldsymbol{d_1^*}}) \\ &= e(g_1, g_2)^{\alpha s\boldsymbol{d_1} \cdot \boldsymbol{d_1^*}} \cdot e(g_1, g_2)^{atsk(x+ID+yd)\boldsymbol{d_1} \cdot \boldsymbol{d_1^*}} \cdot \\ &\quad \cdot e(g_1, g_2)^{z\delta ID\boldsymbol{d_2} \cdot \boldsymbol{d_2^*}} \end{aligned}$$

$$\begin{aligned} T_1 &= \prod_{i \in I} \big(e_n(C_i, K_2') \cdot e_n(K_\rho(i), D_i)\big)^{\omega_i} \\ &= e(g_1, g_2)^{ats(x+ID+yd)\boldsymbol{d_1} \cdot \boldsymbol{d_1^*}} \cdot \\ &\quad \cdot e(g_1, g_2)^{(\sum_{i \in I} \eta_{t_i} + \sum_{i \in I} \kappa_{t_i}) dID\boldsymbol{d_2} \cdot \boldsymbol{d_2^*}} \end{aligned}$$

Finally the algorithm returns CT'':

$$CT'' = \langle C', \tilde{C}', T_1, T_2\rangle$$

Decrypt*(MPK, CT, CT'', RK_S, T_2)* $\rightarrow$ *M*. First, the algorithm checks if $C = C'$ and $\tilde{C} = \tilde{C}'$. Next, it computes:

$$M = \frac{C \cdot y_1^{RK_S}}{T_0 / T_1^{T_2}}$$

If $g_1^{\mathbb{H}(M)} = \tilde{C}'$ the algorithm returns M.

Decryption Correctness:

$$\begin{aligned} M &= \frac{C \cdot y_1^{RK_S}}{T_0 / T_1^{T_2}} \\ &= \frac{M \cdot y^s \cdot e(g_1, g_2)^{\boldsymbol{d_2} \cdot \boldsymbol{d_2^*}(z\delta ID - dIDk \cdot (\sum_{i \in I} \eta_i + \sum_{i \in I} \kappa_{t_i})}}{\frac{e(g_1,g_2)^{\alpha s\boldsymbol{d_1} \cdot \boldsymbol{d_1^*}} \cdot e(g_1,g_2)^{atsk(x+ID+yd)\boldsymbol{d_1} \cdot \boldsymbol{d_1^*}} \cdot e(g_1,g_2)^{z\delta ID\boldsymbol{d_2} \cdot \boldsymbol{d_2^*}}}{\left(e(g_1,g_2)^{ats(x+ID+yd)\boldsymbol{d_1} \cdot \boldsymbol{d_1^*}} \cdot e(g_1,g_2)^{(\sum_{i \in I} \eta_i + \sum_{i \in I} \kappa_{t_i}) dID\boldsymbol{d_2} \cdot \boldsymbol{d_2^*}}\right)^k}} \\ &= \frac{M \cdot e(g_1, g_2)^{\alpha s\boldsymbol{d_1} \cdot \boldsymbol{d_1^*}}}{e(g_1, g_2)^{\alpha s\boldsymbol{d_1} \cdot \boldsymbol{d_1^*}}} \\ &= M \end{aligned}$$

□

KeySanityCheck*(MPK, SK_S)* $\rightarrow$ 1 *or* 0. The algorithm returns 1 if the secret key SK_S passes the key sanity check. Otherwise, it returns 0. SK_S passes the key sanity check if

$$e(K_1, g_2^{\boldsymbol{d_1^*}^{T_1}} XY^{T_3}) = y \cdot e(g_1^{a\boldsymbol{d_1}^{T_2}}, K_2^{T_1} K_3 K_4^{T_3}) \neq 1 \quad (1)$$

$$\forall x \in \mathcal{S}, s.t.\ e(K_x, g_2^{\boldsymbol{d_1^*}}) = e(f_x^{\boldsymbol{d_1}}, K_2^{T_1} K_3, K_4^{T_3}) \neq 1 \quad (2)$$

Trace*(SK_S)* $\rightarrow$ $\perp$ *or (ID, k)*. If ***KeySanityCheck****(SK_S)* $\rightarrow$ 0 the algorithm returns $\perp$. Otherwise, SK_S is well-formed and the algorithm returns (ID, k). If ID does not exist, then the authority must be dishonest. If ID exists and $g_1^{a\boldsymbol{d_1}^k}$ is equal to the user committed value the user is misbehaving. Otherwise, the attribute authority is dishonest.

Revoke*(ID, U_R)*. First, the algorithm removes the user ID from all attribute groups $U \in U_R$. Second, it generates new attribute group keys for all $\kappa_i \in U_R$. Then, it calls *ReEncrypt* on all affected ciphertexts. The new attribute group key needs to be delivered to all valid users through a secure communication channel.

IV. Security-Proof

We first prove that our construction is secure under the SXDH assumption. Next, we prove the accountability property of the proposed scheme.

A. CPA security of our construction

Theorem 1. *If the SXDH assumption holds, the presented CP-ABE scheme is CPA secure. For any adversary $\mathcal{A}$, there exist probabilistic algorithms $\mathcal{B}_0, \mathcal{B}_1, \ldots, \mathcal{B}_v$, which running times are the same as those of $\mathcal{A}$, such that, given the maximum number of key queries v, the following holds:*

$$Adv_{\mathcal{A}}^{ABE}(\lambda) \leq, Adv_{\mathcal{B}_0}^{DDH2}(\lambda) + \sum_{i=1}^{v} Adv_{\mathcal{B}_i}^{DDH1}(\lambda) + \frac{v}{q}$$

We follow the approach by [11] and adapt the proof given in [14]. We rely on *semi-functional ciphertexts* and *semi-functional keys* and provide algorithms to generate them. The provided algorithms are not part of the final system. They do not need to be efficiently computable from MPK or MSK.

KeyGenSF. The algorithm selects $t, z_3, z_4, z, d \xleftarrow{R} \mathbb{Z}_p^*$ and generates a semi-functional secret key as:

$$\begin{aligned} SK_{\mathcal{S}}^{(SF)} &= \langle K_1, K_2, K_3, K_4, \{K_x\}_{x \in \mathcal{S}}, T_1, T_2, T_3\rangle \\ &= \langle g_1^{\frac{\alpha \boldsymbol{d_1}}{(x+ID+yd)}} \cdot g_1^{atk\boldsymbol{d_1}} \cdot g_1^{z_3\boldsymbol{d_3} + z_4\boldsymbol{d_4}}, g_2^{t\boldsymbol{d_1^*}}, g_2^{xt\boldsymbol{d_1^*}}, g_2^{yt\boldsymbol{d_1^*}}, \\ &\quad \{f_x^{t(x+ID+yd)\boldsymbol{d_1}}\}_{x_\epsilon \mathcal{S}}, ID, k, d\rangle \\ TK^{(SF)} &= \langle K_1' = K_1 \cdot g_1^{z\boldsymbol{d_2}}, K_2' = (K_2 \cdot g_2^{d\boldsymbol{d_2^*}})^{T_1} \cdot K_3 \cdot K_4^{T_3}, \\ &\quad \{(K_\rho(i), \omega_i)\}_{i \in I}, I\rangle \\ RK_{\mathcal{S}}^{(SF)} &= z\delta ID - dkID\big(\sum_{i \in I} \eta_i + \sum_{i \in I} \kappa_{t_i}\big) \end{aligned}$$

EncryptSF. The algorithm selects $s, t_3, t_4, \eta_1, \ldots, \eta_l, \delta \xleftarrow{R} \mathbb{Z}_p^*$ and encrypts a message M as:

$$CT^{(SF)} = \langle C = M \cdot y^s, C' = g_2^{s\boldsymbol{d_1^*}} \cdot g_2^{\delta \boldsymbol{d_2^*}} \cdot g_2^{t_3 \boldsymbol{d_3^*} + t_4 \boldsymbol{d_4^*}},$$
$$C_2 = g_2^{xs\boldsymbol{d_1^*}}, C_3 = g_2^{ys\boldsymbol{d_1^*}},$$
$$C_i = g_1^{a\lambda_i \boldsymbol{d_1}} \cdot f_{\rho(i)}^{-r_i \boldsymbol{d_1}} \cdot g_1^{(\eta_i + \kappa_i)\boldsymbol{d_2}}, D_i = g_2^{r_i \boldsymbol{d_1^*}} \}\rangle$$

Decrypting a normal ciphertext with a semi-functional key will succeed because $\boldsymbol{d_3}, \boldsymbol{d_4}$ are orthogonal to all remaining vectors in C'. Likewise, decrypting a semi-functional ciphertext will succeed for a normal key because $\boldsymbol{d_3^*}, \boldsymbol{d_4^*}$ are orthogonal to all the remaining exponents in K_1. If the ciphertext and the key are semi-functional, decryption will fail. The pairing $e_n(K_1', C')$ will then contain an additional term:

$$e(g_1, g_2)^{t_3 z_3 ID \boldsymbol{d_3} \cdot \boldsymbol{d_3^*} + t_4 z_4 ID \boldsymbol{d_4} \cdot \boldsymbol{d_4^*}} = e(g_1, g_2)^{(t_3 z_3 + t_4 z_4) ID \psi}$$

We define the following games between any polynomial-time adversary $\mathcal{A}$ and a challenger C:

- $Game_{Real}$: is the real security game
- $Game_i$ for $i = 0, 1 \ldots v$: $Game_i$ is like $Game_{Real}$, except that the challenge ciphertext is semi-functional and the first i keys given to the attacker are semi-functional. For $Game_0$ all keys are normal, and for $Game_v$ all keys are semi-functional.
- $Game_{Final}$: Is like $Game_v$, except that the ciphertext is a semi-functional encryption of a random message M in $\mathbb{G}_T$.

We prove the following lemmas to show that for each transition from $Game_{Real}$ to $Game_v$, the attacker's advantage cannot change by a non-negligible amount.

Lemma 1. *Suppose that there exists an adversary $\mathcal{A}$. If the adversary has an advantage $|Adv_{\mathcal{A}}^{Game_{Real}}(\lambda) - Adv_{\mathcal{A}}^{Game_0}(\lambda)| = \epsilon$, there exists an algorithm $\mathcal{B}_0$ such that $Adv_{\mathcal{B}_0}^{DS2}(\lambda) = \epsilon$ with $k = 2$ and $n = 4$.*

Proof. $\mathcal{B}_0$ is given a distribution $\boldsymbol{D}$ and T_1, T_2, as defined in Section II-E:

$$\boldsymbol{D} = \langle G, g_1^{\boldsymbol{b_1}}, g_1^{\boldsymbol{b_2}}, g_2^{\boldsymbol{b_1^*}}, \ldots, g_2^{\boldsymbol{b_4^*}}, U_1, U_2, \mu_2 \rangle$$

$\mathcal{B}_0$ needs to decide whether T_1, T_2 are distributed as $g_2^{\tau_1 \boldsymbol{b_1^*}}, g_2^{\tau_1 \boldsymbol{b_2^*}}$ or $g_2^{\tau_1 \boldsymbol{b_1^*} + \tau_2 \boldsymbol{b_3^*}}, g_2^{\tau_1 \boldsymbol{b_2^*} + \tau_2 \boldsymbol{b_4^*}}$. $\mathcal{B}_0$ starts by simulating $Game_{Real}$ or $Game_0$ with $\mathcal{A}$, depending on the distribution of T_1, T_2. Given a security parameter λ, $\mathcal{B}_0$ first generates a random invertible matrix $A \in \mathbb{Z}_p^{2x2}$. It sets the dual orthonormal bases $\mathbb{D}, \mathbb{D}^*$ to:

$$\boldsymbol{d_1} \coloneqq \boldsymbol{b_1}, \quad \boldsymbol{d_2} \coloneqq \boldsymbol{b_2}, \quad (\boldsymbol{d_3}, \boldsymbol{d_4}) \coloneqq (\boldsymbol{b_3}, \boldsymbol{b_4})A,$$
$$\boldsymbol{d_1^*} \coloneqq \boldsymbol{b_1^*}, \quad \boldsymbol{d_2^*} \coloneqq \boldsymbol{b_2^*}, \quad (\boldsymbol{d_3^*}, \boldsymbol{d_4^*}) \coloneqq (\boldsymbol{b_3^*}, \boldsymbol{b_4^*})(A^{-1})^t$$

Now, $\mathcal{B}_0$ chooses random values $a, \alpha \xleftarrow{R} \mathbb{Z}_p^*$ and computes the MPK and MSK according to the ***Setup*** algorithm. Since MSK is known to $\mathcal{B}_0$, it can respond to key queries for a set $\mathcal{S}$ by calling the key generation algorithm. We can now build the challenge ciphertext. First, the adversary $\mathcal{A}$ defines the challenge access structures $\mathbb{A}_0$, $\mathbb{A}_1$. It then generates two messages M_0 and M_1 and sends them to $\mathcal{B}_0$. $\mathcal{B}_0$ now flips a coin β. $\mathcal{B}_0$ encrypts M_β under the challenge access structure $\mathbb{A}_\beta$ and implicitly sets $s \coloneqq \tau_1$:

$$CT = \langle C = M_\beta \cdot e_n(g1^{\boldsymbol{b_1}}, T_1)^\alpha, C' = T_1 \cdot T_2, C_2 = T_1^x, C_3 = T_1^y,$$
$$C_i = g_1^{a\lambda_i \boldsymbol{b_1}} \cdot f_{\rho(i)}^{-r_i \boldsymbol{b_1}} \cdot g_1^{(\eta_i + \kappa_i)\boldsymbol{b_2}}, D_i = g_2^{r_i \boldsymbol{f_1^*}} \}\rangle$$

Finally, it returns the ciphertext CT. If T_1, T_2 are equal to $g_2^{\tau_1 \boldsymbol{b_1^*}}$, $g_2^{\tau_1 \boldsymbol{b_2^*}}$ then $\mathcal{B}_0$ has successfully simulated $Game_{Real}$. However, if T_1,T_2 are equal to $g_2^{\tau_1 \boldsymbol{b_1^*} + \tau_2 \boldsymbol{b_3^*}}, g_2^{\tau_1 \boldsymbol{b_2^*} + \tau_2 \boldsymbol{b_4^*}}$, the ciphertext element C' has an additional term $\tau_2 \boldsymbol{b_3^*} + \tau_2 \boldsymbol{b_4^*}$. $\mathcal{B}_0$ can obtain the coefficients in the basis $\boldsymbol{d_3^*}, \boldsymbol{d_4^*}$ by multiplying the matrix A^{-1} with the transpose of the vector. Since A is uniformly random, the coefficients are uniformly random, and $\mathcal{B}_0$ has correctly simulated $Game_0$. As a result, $\mathcal{B}_0$ can leverage the adversaries' advantage ϵ between $Game_{Real}$ and $Game_0$ against the subspace assumption in $\mathbb{G}_2$: $Adv_{\mathcal{B}_0}^{DS2}(\lambda) = \epsilon$. □

Lemma 2. *Suppose that there exists an adversary $\mathcal{A}$. If the adversary has an advantage $|Adv_{\mathcal{A}}^{Game_{i-1}}(\lambda) - Adv_{\mathcal{A}}^{Game_i}(\lambda)| = \epsilon$. Then there exists an algorithm $\mathcal{B}_i$ such that $Adv_{\mathcal{B}_i}^{DS1}(\lambda) = \epsilon - 1/q$ with $k = 2$ and $n = 4$.*

Proof. $\mathcal{B}_i$ is given a distribution $\boldsymbol{D}$, as defined in Section II-E and and T_1, T_2:

$$\boldsymbol{D} = \langle G, g_2^{\boldsymbol{b_1}*}, g_2^{\boldsymbol{b_2}*}, g_1^{\boldsymbol{b_1}}, \ldots, g_1^{\boldsymbol{b_4}}, U_1, U_2, \mu_2 \rangle$$

$\mathcal{B}_i$ needs to decide whether T_1, T_2 are distributed as $g_1^{\tau_1 \boldsymbol{b_1}}, g_1^{\tau_1 \boldsymbol{b_2}}$ or $g_1^{\tau_1 \boldsymbol{b_1} + \tau_2 \boldsymbol{b_3}}, g_1^{\tau_1 \boldsymbol{b_2} + \tau_2 \boldsymbol{b_4}}$. It begins by simulating $Game_i$ or $Game_{i-1}$ with $\mathcal{A}$, depending on the distribution of T_1, T_2. Given a security parameter λ, $\mathcal{B}_i$ first generates a random invertible matrix $A \in \mathbb{Z}_q^{2x2}$. Then, $\mathcal{B}_i$ sets the dual orthonormal bases $\mathbb{D}, \mathbb{D}^*$ to:

$$\boldsymbol{d_1} \coloneqq \boldsymbol{b_1}, \quad \boldsymbol{d_2} \coloneqq \boldsymbol{b_2}, \quad (\boldsymbol{d_3}, \boldsymbol{d_4}) \coloneqq (\boldsymbol{b_3}, \boldsymbol{b_4})A,$$
$$\boldsymbol{d_1^*} \coloneqq \boldsymbol{b_1^*}, \quad \boldsymbol{d_2^*} \coloneqq \boldsymbol{b_2^*}, \quad (\boldsymbol{d_3^*}, \boldsymbol{d_4^*}) \coloneqq (\boldsymbol{b_3^*}, \boldsymbol{b_4^*})(A^{-1})^t$$

Now, $\mathcal{B}_0$ chooses random values $a, \alpha \xleftarrow{R} \mathbb{Z}_p^*$ and computes the parameters according to the ***Setup*** algorithm. Since MSK is known to $\mathcal{B}_i$, it can respond to key queries for a set $\mathcal{S}$ by calling the standard key generation algorithm. Furthermore, $\mathcal{B}_i$ knows $g_1^{\boldsymbol{d_3}}$ and $g_1^{\boldsymbol{d_4}}$. It can, therefore, easily generate semi-functional keys. For the first $i - 1$ key queries, $\mathcal{B}_i$ runs the semi-functional key generation algorithm and returns them to $\mathcal{A}$. For the ith key query for a set of attributes $\mathcal{S}$, $\mathcal{B}_i$ responds with:

$$
\begin{aligned}
SK_{\mathcal{S}} &= \langle K_1, K_2, K_3, K_4, \{K_x\}_{x\in\mathcal{S}}, T_1, T_2, T_3\rangle \\
&= \langle (g_1^{\boldsymbol{b_1}})^{\alpha/(x+ID+yd)} \cdot (T_1)^{tk}, g_2^{t\boldsymbol{b_1^*}}, g_2^{xt\boldsymbol{b_1^*}}, g_2^{yt\boldsymbol{b_1^*}}, \\
&\quad \{f_x^{t(x+ID+yd)\boldsymbol{b_1}}\}_{x_\in\mathcal{S}}, ID, k, d\rangle \\
TK_{CT} &= \langle K_1' = K_1 \cdot T_2, K_2' = K_2 \cdot (g_2^{\boldsymbol{b_2^*}})^d K_3 K_4^{T_3}, \\
&\quad \{(K_\rho(i), \omega_i)\}_{i\in I}, I\rangle
\end{aligned}
$$

$\mathcal{B}_0$ implicitly sets $a \coloneqq \tau_1$. If T_1, T_2 are equal to $g_1^{\tau_1\boldsymbol{b_1}}$, $g_1^{\tau_1\boldsymbol{b_2}}$, then the key is properly distributed. However, if T_1,T_2 are equal to $g_1^{\tau_1\boldsymbol{b_1}+\tau_2\boldsymbol{b_3}}$, $g_1^{\tau_1\boldsymbol{b_2}+\tau_2\boldsymbol{b_4}}$, the key is semi-functional. The exponent vector then includes an additional term $t\tau_2\boldsymbol{b_3}+\tau_2\boldsymbol{b_4}$. For the remaining key queries, $\mathcal{B}_k$ simply runs the regular key generation algorithms.

Next, the adversary $\mathcal{A}$ defines the challenge access structures $\mathbb{A}_0$, $\mathbb{A}_1$. It then generates two messages M_0 and M_1 and sends them to $\mathcal{B}_0$. $\mathcal{B}_1$ now flips a coin β. $\mathcal{B}_1$ implicitly sets $s \coloneqq \tau_1$. It encrypts M_β under the challenge access structure $\mathbb{A}_\beta^*$ as follows:

$$
\begin{aligned}
CT =& \langle C = M_\beta \cdot \left(e_n(g_1^{\boldsymbol{b_1}}, U_1)\right)^\alpha = M_\beta \cdot \left(e_n(g_1, g_2)^{\alpha\boldsymbol{d_1}\boldsymbol{d_1^*}}\right)^s, \\
& C' = U_1 \cdot U_2, C_2, C_3, \\
& \{C_i = g_1^{a\lambda_i\boldsymbol{b_1}} \cdot f_{\rho(i)}^{-r_i\boldsymbol{b_1}} \cdot g_1^{(\eta_i+\kappa_{t_i})\boldsymbol{b_2}}, D_i = g_2^{r_i\boldsymbol{b_1^*}}\}\rangle
\end{aligned}
$$

where $\mathcal{B}_1$ implicitly sets $s \coloneqq \mu_1$. The semi-functional part of the ciphertext now contains $\mu_2\boldsymbol{b_3^*} + \mu_2\boldsymbol{b_4^*}$. The distribution for the $i-1$ first keys is independent of the random matrix A. Likewise, the challenge ciphertext is independent of the random matrix A. The coefficients are uniformly random (except for $1/q$ probability from [16]). Summarising, $\mathcal{B}_i$ can properly simulate either $Game_{i-1}$ or $Game_i$, depending on the distribution of T_1, T_2. As a result, $\mathcal{B}_0$ can leverage the adversaries' advantage ϵ between $Game_{i-1}$ and $Game_i$ against the subspace assumption in $\mathbb{G}_1$: $Adv_{\mathcal{B}_i}^{DS1}(\lambda) = \epsilon - 1/q$. □

Lemma 3. *For any adversary $\mathcal{A}$, $Adv_{\mathcal{A}}^{Game_i}(\lambda) = Adv_{\mathcal{A}}^{Game_{Final}}(\lambda)$.*

Proof. To prove this lemma, we show that an adversary $\mathcal{A}$ cannot distinguish between the joint distributions of $Game_\nu$: $(MPK, CT^{(SF)}, \{SK_{\mathcal{S}_l}^{(SF)}\}_{l=1,\dots,i}, \{TK_l^{(SF)}\}_{l=1,\dots,i})$ and $Game_{Final}$: $(MPK, CT^{(R)}, \{SK_{\mathcal{S}_l}^{(SF)}\}_{l=1,\dots i}, \{TK_l^{(SF)}\}_{l=1,\dots i})$, where $CT^{(R)}$ is a semi-functional encryption of a random message in $\mathbb{G}_T$ under a random access structure. We again define a matrix A. This time, we set $A \coloneqq (\xi_{i,j}) \xleftarrow{R} \mathbb{Z}_p^{2x2}$ and define new orthonormal bases $\mathbb{F} \coloneqq (\boldsymbol{f_1}, \dots, \boldsymbol{f_4})$, and $\mathbb{F}^* \coloneqq (\boldsymbol{f_1^*}, \dots, \boldsymbol{f_4^*})$:

$$
\begin{pmatrix} \boldsymbol{f_1} \\ \boldsymbol{f_2} \\ \boldsymbol{f_3} \\ \boldsymbol{f_4} \end{pmatrix} \coloneqq \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \xi_{1,1} & \xi_{1,2} & 1 & 0 \\ \xi_{2,1} & \xi_{2,2} & 0 & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{d_1} \\ \boldsymbol{d_2} \\ \boldsymbol{d_3} \\ \boldsymbol{d_4} \end{pmatrix},
$$

$$
\begin{pmatrix} \boldsymbol{f_1^*} \\ \boldsymbol{f_2^*} \\ \boldsymbol{f_3^*} \\ \boldsymbol{f_4^*} \end{pmatrix} \coloneqq \begin{pmatrix} 1 & 0 & -\xi_{1,1} & -\xi_{2,1} \\ 0 & 1 & -\xi_{1,2} & -\xi_{2,2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{d_1^*} \\ \boldsymbol{d_2^*} \\ \boldsymbol{d_3^*} \\ \boldsymbol{d_4^*} \end{pmatrix}
$$

Next, we express our public parameters, the challenge ciphertext and the secret keys in $Game_\nu$ with the bases of $\mathbb{F}, \mathbb{F}^*$. We change the coefficients (s, δ) in the ciphertext term to random coefficients $(s', s'') \in \mathbb{Z}_p \times \mathbb{Z}_p$ of $\boldsymbol{f_1^*}, \boldsymbol{f_2^*}$. We let:

$$
\begin{aligned}
SK_{\mathcal{S}_l}^{(SF)} &= \langle K_1, K_2, K_3, K_4, \{K_x\}_{x\in\mathcal{S}}\rangle \\
&= \langle g_1^{\alpha/(x+ID+yd)\boldsymbol{f_1}} \cdot g_1^{at_l\boldsymbol{f_1}} \cdot g_1^{z'_{l,3}\boldsymbol{f_3}+z'_{l,4}\boldsymbol{f_4}}, g_2^{t_l\boldsymbol{f_1^*}}, g_2^{xt_l\boldsymbol{f_1^*}}, \\
&\quad g_2^{yt_l\boldsymbol{f_1^*}}, \{f_x^{t_l\boldsymbol{f_1}}\}_{x_\in\mathcal{S}}\rangle \\
TK_{CT}^{(SF)} &= \langle K_1' = K_1 \cdot g_1^{z_l\boldsymbol{f_2}}, K_2' = (K_2 \cdot g_2^{d_l\boldsymbol{f_2^*}})^{T_1} K_3 K_4^{T_3}, \\
&\quad \{(K_\rho(i), \omega_i)\}_{i\in I}, I\rangle \\
RK_{\mathcal{S}_l} &= z_l\delta - d(\sum_{i\in I}\eta_i + \sum_{i\in I})\kappa_{t_i} \\
CT^{(SF)} &= \langle C = M \cdot y^s, C' = g_2^{s'\boldsymbol{f_1^*}} \cdot g_2^{s''\boldsymbol{f_2^*}} \cdot g_2^{t'_3\boldsymbol{f_3^*}+t'_4\boldsymbol{f_4^*}}, \\
&\quad C_i = g_1^{a\lambda_i\boldsymbol{f_1}} \cdot f_{\rho(i)}^{-r_i\boldsymbol{f_1}} \cdot g_1^{(\eta_i+\kappa_i)\boldsymbol{f_2}}, D_i = g_2^{r_i\boldsymbol{f_1^*}}\}\rangle
\end{aligned}
$$

We set $s' = s - t_3\xi_{1,1} - t_4\xi_{2,1}$, $s'' = \delta - t_3\xi_{1,2} - t_4\xi_{2,2}$, $\{z'_{l,3} = z_{l,3} + \xi_{1,1}(\alpha + at_l) + z_l\xi_{1,2}$, $z'_{l,4} = z_{l,4} + \xi_{2,1}(\alpha + at_l) + z_l\xi_{2,2}$, $r_1, \dots, r_l, \lambda_1, \dots, \lambda_l \xleftarrow{R} \mathbb{Z}_p^*\}_{l=1,\dots,\nu}$, . The values $\xi_{1,1}, \dots, \xi_{2,2}$, $t_{1,3}, \dots, t_{\nu,3}$, $t_{1,4}, \dots, t_{\nu,4}$, $r_1, \dots, r_l$, $\lambda_1, \dots, \lambda_l$ are all uniformly distributed in $\mathbb{Z}_p^*$. Thus, the challenge ciphertext can be seen as a semi-functional encryption of a random element in $\mathbb{G}_T$ under a random access structure. From the adversary's perspective, both $(\mathbb{D}, \mathbb{D}^*)$ and $(\mathbb{F}, \mathbb{F}^*)$ are consistent with the same public key. We can therefore express the challenge ciphertext and the queried keys in two ways: In $Game_\nu$ over the bases $(\mathbb{D}, \mathbb{D}^*)$ and in $Game_{Final}$ over the bases $(\mathbb{F}, \mathbb{F}^*)$. Thus, the adversary cannot distinguish between $Game_\nu$ and $Game_{Final}$. □

Lemma 4. *For any adversary $\mathcal{A}$, $Adv_{\mathcal{A}}^{Game_{Final}}(\lambda) = 0$.*

Proof. The value of β is independent from the adversary's view in $Game_{Final}$. Hence, $Adv_{\mathcal{A}}^{Game_{Final}}(\lambda) = 0$. □

In $Game_{Final}$ the challenge ciphertext is a semi-functional encryption of a random element in $\mathbb{G}_T$ under a random access structure. It is independent of the provided messages and access structures. Hence, our CP-ABE scheme provides hidden policies.

B. Proof of accountability

Since k is only known to the user, we rely on the same technology proposed by [17]. We assume that the simulator knows k, relying on some knowledge extractor.

Theorem 2. *If the q-SDH and DLP assumptions hold, the presented CP-ABE scheme provides accountability of dishonest users.*

We follow the approach by Li et al. [15]. We prove the following lemmas to show that the attacker's advantage cannot change by a non-negligible amount.

Lemma 5. *Suppose there exists an adversary $\mathcal{A}$ which can win the first dishonest user game with a non-negligible advantage $Adv_{\mathcal{A}}$. In that case, there exists a probabilistic algorithm $\mathcal{B}$, which can break the q-SDH assumption with an advantage $\epsilon/2$.*

Proof. To prove this lemma, we assume that $\mathcal{A}$ queries q keys. Each query is recorded in the tuple $(ID_i, d_i, c_i = ID_i + yd_i)$. A forged key is described by the tuple $(ID^*, d^*, c^* = ID^* + yd^*)$. Analogous to the proof by Li et al. [15], we consider two types of adversaries, related to the conditions $c^* \in \{c_i\}_{i\in[q]}$ and $c^* \notin \{c_i\}_{i\in[q]}$: The first adversary manages to either query $ID = -x$ in the key query phase or manages to forge a *SK* related to $c^* \notin \{c_i\}_{i\in[q]}$. The second adversary does not query $ID = -x$ in the key query phase but manages to forge a *SK* related to $c^* \in \{c_i\}_{i\in[q]}$. To indicate which adversary we are dealing with, $\mathcal{B}$ randomly chooses $b_{mode} \in \{1, 2\}$.

At first, $\mathcal{B}$ is given a distribution $\boldsymbol{D}$ according to the q-SDH problem, defined in Section II-F. Let $A_i = \{g_1^{x^i}\}_{0\le i\le q}$. Next, $\mathcal{B}$ randomly chooses elements $\{c_i\}_{1\le i<q}$. Then, it defines $f(z) = \prod_{i=1}^{q-1}(z + c_i) = \sum_{i=0}^{q-1}\eta_i z^i$, where $\eta_i \in \mathbb{Z}_p^*$ are the coefficients of the polynomial. $\mathcal{B}$, then, computes $g_1 = \prod_{i=0}^{q-1} A_i^{\eta_i}$, $Q = g_1^x = \prod_{i=1}^{q} A_i^{\eta_{i-1}} = (g')^{xf(x)}$ Like in the original proof, we know that $f(x) \neq 0$ because if $f(x) = 0$, then $x = c_i$. Now, $\mathcal{B}$ computes the parameters according to the ***Setup*** algorithm. However, $\mathcal{B}$ chooses $\theta_1, \dots, \theta_U \xleftarrow{R} \mathbb{Z}_p^*$ and sets $\{f_i\}_{i\in U} = \{g_1^{\theta_i}\}_{i\in U}$. Furthermore, if $b_{mode} = 1$, $\mathcal{B}$ chooses $y \xleftarrow{R} \mathbb{Z}_p^*$ and sets $X = Z = g_2^x, Y = g_2^y$. Else, if $b_{mode} = 2$, $\mathcal{B}$ chooses $x' \xleftarrow{R} \mathbb{Z}_p^*$ and sets $X = g_2^{x'}, Y = Z = g_2^x$.

Next, $\mathcal{A}$ queries keys for $(ID_1, \mathcal{S}_1)$, ..., $(ID_q, \mathcal{S}_q)$. For every key query i, $\mathcal{B}$ computes the hash list $(ID_i, \mathcal{S}_i, d_i)$. Then, $\mathcal{B}$ computes $f_i(z) = f(z)/(z + c_i) = \prod_{j=1, j\neq i}^{q-1}(z + c_j) = \sum_{i=0}^{q-2}\gamma_j z^j$, where $\gamma_j \in \mathbb{Z}_p$ are the coefficients of the polynomial. $\mathcal{B}$ then computes $\lambda_i = \prod_{j=0}^{q-2} A_j^{\gamma_i} = (g')^{f_i(x)} = g_1^{1/(x+c_i)}$. If $b_{mode} = 1$ and $g_1^{-ID} = g_1^x$, $\mathcal{B}$ can directly calculate x of q-SDH directly, since $\mathcal{B}$ can compute $(c, g_1^{1/(x+c)})$ with any c. As a result, $\mathcal{B}$ can break the q-SDH assumption. Else, $\mathcal{B}$ sets $d_i = (c_i - ID_i)/y \in \mathbb{Z}_p^*$. If $d_i = 0$, $\mathcal{B}$ aborts. Else, if $d_i \neq 0$, $\mathcal{B}$ chooses $t \xleftarrow{R} \mathbb{Z}_p^*$, and returns the secret key $SK_{\mathcal{S}_i}$:

$$SK_{\mathcal{S}_i} = \langle K_1, K_2, K_3, K_4, \{K_x\}_{x\in\mathcal{S}}, T_1, T_2, T_3\rangle$$
$$= \langle \lambda_i^{\alpha \boldsymbol{d_1}} \cdot g_1^{atk\boldsymbol{d_1}} = g_1^{\frac{\alpha \boldsymbol{d_1}}{(x+ID_i+yd_i)}} \cdot g_1^{atk\boldsymbol{d_1}}, g_2^{t\boldsymbol{d_1^*}},$$
$$X^{t\boldsymbol{d_1^*}} = g_2^{xt\boldsymbol{d_1^*}}, Y^{t\boldsymbol{d_1^*}} = g_2^{xt\boldsymbol{d_1^*}}, \{f_x^{t(ID+yd)} g_1^{x^{\theta_i - t}\boldsymbol{d_1}}\}_{x_\epsilon\mathcal{S}}, ID, k, d_i\rangle$$

If $b_{mode} = 2$, $\mathcal{B}$ chooses $d_i = (x + ID_i)/c_i \in \mathbb{Z}_p^*$, If $d_i = 0$, $\mathcal{B}$ aborts again. Else, if $d_i \neq 0$, $\mathcal{B}$ computes the key as follows.

$$SK_{\mathcal{S}_i} = \langle K_1, K_2, K_3, K_4, \{K_x\}_{x\in\mathcal{S}}, T_1, T_2, T_3\rangle$$
$$= \langle \lambda_i^{\alpha/d_i \boldsymbol{d_1}} \cdot g_1^{atk\boldsymbol{d_1}} = g_1^{\frac{\alpha \boldsymbol{d_1}}{(x+ID_i+yd_i)}} \cdot g_1^{atk\boldsymbol{d_1}}, g_2^{t\boldsymbol{d_1^*}},$$
$$X^{t\boldsymbol{d_1^*}} = g_2^{xt\boldsymbol{d_1^*}}, Y^{t\boldsymbol{d_1^*}} = g_2^{xt\boldsymbol{d_1^*}}, \{f_x^{t(ID+yd)} g_1^{x^{\theta_i t}\boldsymbol{d_1}}\}_{x_\epsilon\mathcal{S}}, ID, k, d_i\rangle$$

After each key query, $\mathcal{B}$ adds the tuple $(ID_i, d_i, H_i = g_1^{ID_i} Y^{d_i})$ to a hash list.

Next, $\mathcal{A}$ generates a well-formed secret key $SK_{\mathcal{S}}$ related to (ID^*, d^*), which has not been queried before. Let $H^* = g_1^{ID^*} Y^{d^*}$. $\mathcal{B}$ then searches in the stored hash list. If there exists no tuple with the same hash value, $\mathcal{B}$ sets $B_{mode} = 1$. In turn, if there exists a tuple, equal to the hash, $\mathcal{B}$ sets $B_{mode} = 2$. If $b_{mode} \neq B_{mode}$, $\mathcal{B}$ aborts.

Now, if $B_{mode} = b_{mode} = 1$, $\mathcal{B}$ checks whether $g_1^{-ID} = X$. If that is the case, $\mathcal{B}$ can directly compute $(c, g_1^{1/(x+c)})$ from any c. *$\mathcal{B}$ can thus break the q-SDH assumption.* Else, let $c^* = ID^* + yd^*$. $\mathcal{B}$ then computes $\lambda^* = (K_1 g_1^{-atd_1{}^k})^{1/\alpha \boldsymbol{d_1}} = g_1^{1/(x+ID^*+yc^*)} = (g')^{f(x)/(x+ID^*+yc^*)}$. When $b_{mode} = 1$, $c^* \notin \{c_i\}_{i\in[-l-1]}$. Let $f(z) = \gamma(z)(z + c^*) + \gamma_{-1}$, where $\gamma(z) = \sum_{i=0}^{l-2}\gamma_i z^i$ and $\gamma_i \in \mathbb{Z}_p$. We compute $f(z)/(z + c^*) = \sum_{i=0}^{l-2}(\gamma_i z^i + \gamma_{-1})/(z + c^*)$ and $\lambda^* = (g')^{\sum_{i=0}^{l-2}(\gamma_i z^i + \gamma_1)/(z+d^*)}$, where $\gamma_i = 0$. $f(z) = \prod_{i=1}^{l-1}(z + c_i)$. Finally, $\mathcal{B}$ can solve the q-SDH problem by calculating: $(\lambda^* \cdot \prod_{i=0}^{l-2} A_i^{-\gamma_i})_{-1}^{\gamma} = \left((g')^{\gamma_{-1}/(x+c^*)} \cdot (g')^{\sum_{i=0}^{l-2}\gamma_i^{x^i}} \prod_{i=0}^{l_2}(g')^{-\gamma_i x^i}\right)^{\gamma_{-1}} = (g')^{1/(x+c^*)}$. It returns $(c^*, (g')^{1/(x+c^*)})$ as the solution to the q-SDH problem.

In turn, if $B_{mode} = b_{mode} = 2$, $\mathcal{B}$ can compute $x = (ID_i - ID^*)/(d_i - d^*)$. Since, $Y_1 = g_1^x$, $H_i = H^* = g_1^{ID_i} g^{xd_i} = g_1^{ID^*} g_1^{xd^*}$, $ID_i + xd_i = ID^* + xd^*$. As a result, $\mathcal{B}$ can break the q-SDH assumption for $b_{mode} = 2$.

As can be seen, the choice of b_{mode} is independent of $\mathcal{A}$. The distribution of the public key is the same as in the real scheme. $SK_{\mathcal{S}}$ is well-formed. Assuming, $\mathcal{B}$ calculates a well-formed $SK_{\mathcal{S}}$ with probability ϵ, we can analyse the probability of $\mathcal{B}$ breaking the q-SDH assumption. Like in the proof of Li et al. [15], $\mathcal{B}$ aborts with at least probability $(l - 1)/p$, if $B_{mode} = b_{mode} = 1$. For $B_{mode} = b_{mode} = 2$, $\mathcal{B}$ does not abort. As a result, $Pr[B_{mode} = b_{mode} = 1/2]$. We conclude that $\mathcal{B}$ can solve the q-SDH problem with the following probability:

$$\begin{aligned}
&Pr[\mathcal{B}\ not\ abort\ \&\ win | B_{mode} = b_{mode} = 1] \\
&\quad \cdot Pr[B_{mode} = b_{mode} = 1] \\
&\quad + Pr[\mathcal{B}\ not\ abort\ \&\ win | B_{mode} = b_{mode} = 2] \\
&\quad \cdot Pr[B_{mode} = b_{mode} = 2] \\
&= \epsilon(1 - (l - 1)/p) \cdot 1/4 + \epsilon \cdot 1/4 \\
&= \epsilon/(2 - (l - 1)) \cdot \epsilon/4p \\
&\approx \epsilon/2.
\end{aligned}$$

□

Lemma 6. *Suppose an adversary $\mathcal{A}$ which can win the second dishonest user game with a non-negligible advantage $Adv_{\mathcal{A}}$.*

In that case, there exists a probabilistic algorithm $\mathcal{B}$, which can break the DLP with an advantage ϵ.

Proof. $\mathcal{B}$ first generates the public parameters according to the ***Setup*** algorithm. $\mathcal{A}$ then queries q keys. $(ID_i, \mathcal{S}_i)$. For every ith query, $\mathcal{B}$ extracts k_i from $g_1^{ak_i\boldsymbol{d_1}}$. Given the DLP and a tuple (g_1, g_1^z), $\mathcal{B}$ chooses $\gamma_i, t_i \xleftarrow{R} \mathbb{Z}_p$ and sets $t^{(i)} = \gamma_i z$ implicitly by computing $g_1^{ak_i d^{(i)}} = (g_1^z)^{ak_i\gamma_i}$, $K_{i,2} = g^{d^{(i)}t_i} = (g^z)^{\gamma_i t_i}$. Then, $\mathcal{B}$ sends $SK_{\mathcal{S}_i}$ to $\mathcal{A}$. In the next phase, $\mathcal{A}$ outputs a secret key for the tuple (ID, d, k). We assume SK to be well-formed. That means $(ID, d) = (ID_i, d_i)$ and $k \neq k_i$.

$$\begin{aligned} SK_{\mathcal{S}} &= \langle K_1, K_2, K_3, K_4, \{K_x\}_{x\in\mathcal{S}}, T_1, T_2, T_3\rangle \\ &= \langle g_1^{\frac{\alpha\boldsymbol{d_1}}{(x+ID_i+yc)}} \cdot g_1^{atk\boldsymbol{d_1}}, g_2^{t\boldsymbol{d_1^*}}, g_2^{xt\boldsymbol{d_1^*}}, g_2^{yt\boldsymbol{d_1^*}} \\ &\{f_x^{t(x+ID+yc)\boldsymbol{d_1}}\}_{x_\in\mathcal{S}}, ID, k, d\rangle \end{aligned}$$

The adversary $\mathcal{A}$, then generates a forged key as follows:

$$\begin{aligned} SK_{\mathcal{S}} &= \langle K_1', K_2', K_3', K_4', \{K_x\}_{x\in\mathcal{S}}, T_1', T_2', T_3'\rangle \\ &= \langle g_1^{\frac{\alpha\boldsymbol{d_1}}{(x+ID_i+c)}} \cdot g_1^{at'k'\boldsymbol{d_1}}, g_2^{t'\boldsymbol{d_1^*}}, g_2^{xt'\boldsymbol{d_1^*}}, g_2^{yt'\boldsymbol{d_1^*}} \\ &\{f_x^{t'(x+ID+yd)\boldsymbol{d_1}}\}_{x_\in\mathcal{S}}, ID, k', d\rangle \end{aligned}$$

We observe that $\frac{\alpha\boldsymbol{d_1}}{(x+ID_i+c)}$ and d is unknown to $\mathcal{A}$. However, if $\mathcal{A}$ can successfully generate K_1', it can calculate: $K_1' = K_1 \cdot g_1^{a p_1} \Rightarrow kd + p_1 = k'd'$, where $p_1 \in \mathbb{Z}_p$. Likewise, $K_x' = (K_x)^{p_2} \Rightarrow dp_2 = d'$. $\mathcal{A}$ can now solve the equation $kd + p_1 = k'd'$, $dp_2 = d'$ and calculate $d = p_1/(k'p_2 - k)$. We can argue that the probability $k'p_2 = k$ is negligible. Finally, $\mathcal{B}$ can calculate the solution to the DLP problem: We set $d = \gamma_i z$. Then, $\mathcal{A}$ can calculate the solution: $z = d/\gamma_i = p_1/(\gamma_i(k'p_2 - k))$. We conclude that if $\mathcal{A}$ can successfully generate $SK_{\mathcal{S}}'$, where $k \neq k'$, there exists an algorithm which can break the DLP assumption with an advantage ϵ. □

V. Evaluation

We provide an implementation of our solution, and compare it to existing work. This section describes the process and highlights the results.

Implementation details. We implemented two versions of our proposed solution in Kotlin. To do so, we relied on two different libraries for pairing-related functionality: *Java Pairing-Based Cryptography Library* (JPBC) [18] and the *IAIK ECCelerate*^{TM}[1] library. To put our solution in perspective against existing work, we implemented the similar-featured scheme presented in [15] using the JPBC library.

Platform. We evaluated our construction on two different platforms: We used an Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz on Debian 10 - 5.9.0-0.bpo.5-amd64 to represent a powerful cloud server and a Raspberry Pi 3 Model B+ on Raspbian GNU/Linux 10 - 5.4.83-v7+ to represent a more resource constrained device. These two platforms show how the proposed construction copes with different platforms and available computing resources.

Setting. We carried out multiple benchmarks using the *Java Microbenchmark Harness* (JMH)[2]. To initialise the test system, we used 5 warmup forks with 5 warmup iterations. For benchmarking, we used 10 forks with 10 iterations.

Policies. For each test, we generated random access policies from two up to 100 attributes.

Security. We relied on the proposed parameters by Guillevic [19] and Kiraz and Uzunkol [20]. They are equivalent to a security level of AES-128 and provide near-term security (at least ten years) as defined by NIST [21]. In detail, we selected groups of size $\mathbb{G}_1 = 256$, $\mathbb{G}_2 = 512$, $\mathbb{G}_T = 3072$ with an embedding degree $k = 12$, for the proposed construction. For symmetric pairings as used by [22], we set $r = 160$ and $q = 3000$.

Comparison. We now compare our construction in terms of flexibility and overhead against related work. Table I highlights our findings. We describe the overhead by the number of pairing (P) and exponentiation (E) operations. Furthermore, we estimate the ciphertext and key size by the number of group elements of size $|\mathbb{G}|$ or $|\mathbb{G}_T|$. The value $|PK|$ describes the length of the public parameters. $|SK|$ denotes the length of the user key. We use $|CT|$ to describe the ciphertext size.

From the table, we can see that our construction can compete with existing work. Indeed, *Encrypt* operation only introduces a nominal overhead. In turn, *Decrypt* operation clearly outperforms existing work due to it being a single exponentiation. Turning to storage requirements, we find a similar pattern. The only outlier can be observed when looking at the public key. The size of the public parameters for our construction grows linearly with the number of attributes. In contrast, all the analysed related-work supports the large attribute universe. This means that the public parameters are independent of the number of attributes used. The table also highlights that our construction manages to combine hidden policies, accountability and revocability. Finally, the table outlines the supported access policies. Like ours, the construction by Ning et al. [23] supports LSSS, which is generally speaking more flexible than, e.g. access structures based on access trees or AND-gates.

Timing benchmarks. Figure 1 shows the timing benchmarks conducted on the Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz. From the graph, we can see that our construction (in red and green) outperforms the construction proposed by Li et al. (in blue) for almost all operations. We attribute this to the fact that our construction is designed around Type III pairings instead of Type I pairings. Li et al.'s scheme, however, outperforms our scheme in the setup phase, as it accounts for a large attribute universe. The graph, furthermore, confirms our assumption that the *Decrypt* operation is constant and independent of the number of used attributes. Li et al.'s scheme does not provide outsourced decryption. Hence, the operations

[1] https://jce.iaik.tugraz.at

[2] https://openjdk.java.net/projects/code-tools/jmh/

TABLE I: Comparison of the proposed scheme with existing constructions. n is the number of attributes in the system. l symbolises the number of user attributes. We use i to specify the complexity (e.g. the number of attributes) of a policy. We denote P as the time for a pairing operation and E as the time for an exponentiation. Cells in **bold** highlight the best value for each column.

	Li et al. [24]	Ning et al. [23]	Qiao et al. [25]	Li et al. [15]	Ours
$\|PK\|$	$\mathbf{2\|\mathbb{G}\| + \|\mathbb{G}_T\|}$	$(n+4)\|\mathbb{G}\| + \|\mathbb{G}_T\|$	$\mathbf{2\|\mathbb{G}\| + \|\mathbb{G}_T\|}$	$7\|\mathbb{G}_1\| + \|\mathbb{G}_T\|$	$(2n+4)\|\mathbb{G}_1\| + 8\|\mathbb{G}_2\| + 2\|\mathbb{G}_T\|$
$\|SK\|$	$(4n+4\|ID\|)\|\mathbb{G}\|$	$(l+6)\|\mathbb{G}\|$	$\mathbf{(l+4)\|\mathbb{G}\|}$	$(2l+4)\|\mathbb{G}\|$	$(2l+2)\|\mathbb{G}_1\| + 6\|\mathbb{G}_2\|$
$\|CT\|$	$(4i+8\|ID\|)\|\mathbb{G}\| + \|\mathbb{G}_T\|$	$(2i+3)\|\mathbb{G}\| + \|\mathbb{G}_T\|$	$(4i+1)\|\mathbb{G}\| + \|\mathbb{G}_T\|$	$\mathbf{(2l+4)\|\mathbb{G}\|}$	$2i\|\mathbb{G}_1\| + (2i+6)\|\mathbb{G}_2\| + \|\mathbb{G}_T\|$
Encrypt	$(4i+8\|ID\|+1)E$	$\mathbf{(3i+4)E}$	$(5i+2)E$	$(5i+3)E$	$(6i+4)E$
Decrypt	$(4n+4\|ID\|)P$	$(l+3)E + (2n+2)P$	$2iE + (2i+1)P$	$(l+4)E + (3l+1)P$	$\mathbf{1E}$
Trace	$\mathbf{(4i+4\|ID\|)E}$	$\mathbf{3E + (2l+8)P}$	$(5i+2)E + (2i+1)P + uE$	$(l+6)E + (4l+2)P$	$(2l+12)E + (2l+2)P$
Hidden policies	✓	✗	✗	✓	✓
Revocation	✗	✓	✗	✗	✓
Policy	AND	LSSS	Access tree	AND^*_m	LSSS
Group	Prime	Composite	Prime	Prime	Prime
Model	Black-Box	White-Box	Black-Box	White-Box	White-Box

TABLE II: Execution Time of Encrypt and Decrypt operations on a Raspberry Pi 3 Model B+.

Attribute count	Encrypt	Decrypt
2	326 ms	162 ms
10	890 ms	162 ms
20	1 638 ms	162 ms
30	2 429 ms	162 ms
40	3 274 ms	163 ms
50	4 135 ms	163 ms
60	5 116 ms	162 ms
70	6 064 ms	162 ms
80	7 084 ms	162 ms
90	8 157 ms	162 ms
100	9 983 ms	162 ms

are missing from the graph.

In turn, Table II shows the benchmark results of operations performed on the Raspberry Pi 3 Model B+ for the JPBC library. We intentionally evaluated *Encrypt* and *Decrypt* operations on this platform, as these operations are typically not carried out by a cloud server. From the table, we can see that the time for *Encrypt* operation increases linearly with the number of used attributes. For a policy with two attributes, the execution time is around 300ms. For a policy with 100 attributes, the execution takes around 10s. At first glance, this result looks like a significant overhead. However, ABE provides one-to-many encryption. That means that a ciphertext only needs to be encrypted once, regardless of the number of recipients. In turn, the *Decrypt* operation takes around 160ms, independent of the number of attributes.

VI. Related-Work

We present the major concepts of privacy-aware accountable ABE and discuss their relevance to our work.

Li et al. [24] first presented a privacy aware CP-ABE scheme with user accountability to prevent illegal key sharing. User accountability is achieved in a black-box model by embedding user specific information in attribute private key. In contrast to our construction, expensive pairing operations need to be run on the client. Xhafa et al. [26] propose a multi-authority CP-ABE scheme with user accountability and hidden access policies. The solution targets a patient-centric model of health information exchange. The scheme allows identifying a misbehaving user who shared their decryption key. It embeds a user global identity into the user's private key. The scheme does not account for resource-constrained devices. In particular, expensive bilinear pairing operations are performed on the client device. Liu et al. presented traceable white-box [27] and black-box [28] CP-ABE schemes. Both schemes rely on composite order bilinear groups. As a result, concrete implementations of the schemes result in less efficient pairing operations and are thus less efficient for resource-constrained environments. Ning et al. [29, 30] propose two white-box traceable, large universe CP-ABE systems. Both systems rely on prime order bilinear groups, like ours does. Their constructions support any monotone access structure and guarantee constant storage for tracing. The decryption operation, however, still relies on pairing operations, and might therefore not be applicable in resource-constrained environments. They, later [31, 32], improved their construction. It allows identifying malicious authorities, i.e. authorities that illegally distribute keys, and malicious users, i.e. users that share their key. The construction is designed in the white-box model and supports arbitrary monotone access structures. However, the construction depends on composite-order groups and is, therefore, limited to more powerful devices. Qiao et al. [25] propose a novel black-box compulsory traceable CP-ABE approach for fog systems. Compulsory means that the tracing ciphertext must not be distinguishable form a normal ciphertext. The approach scales well and is efficient. In contrast to our scheme, decryption requires expensive bilinear pairing operations to be run on the client device. In our construction, the client is left with only a single exponentiation. Ning et al. [23] propose a CP-ABE system which relies on novel non-interactive commitments for traitor tracing. The resulting scheme relies on composite order groups with three distinct primes. Our construction, in contrast, relies on more efficient prime order bilinear groups and does not require pairing operations on client devices. Most recently,

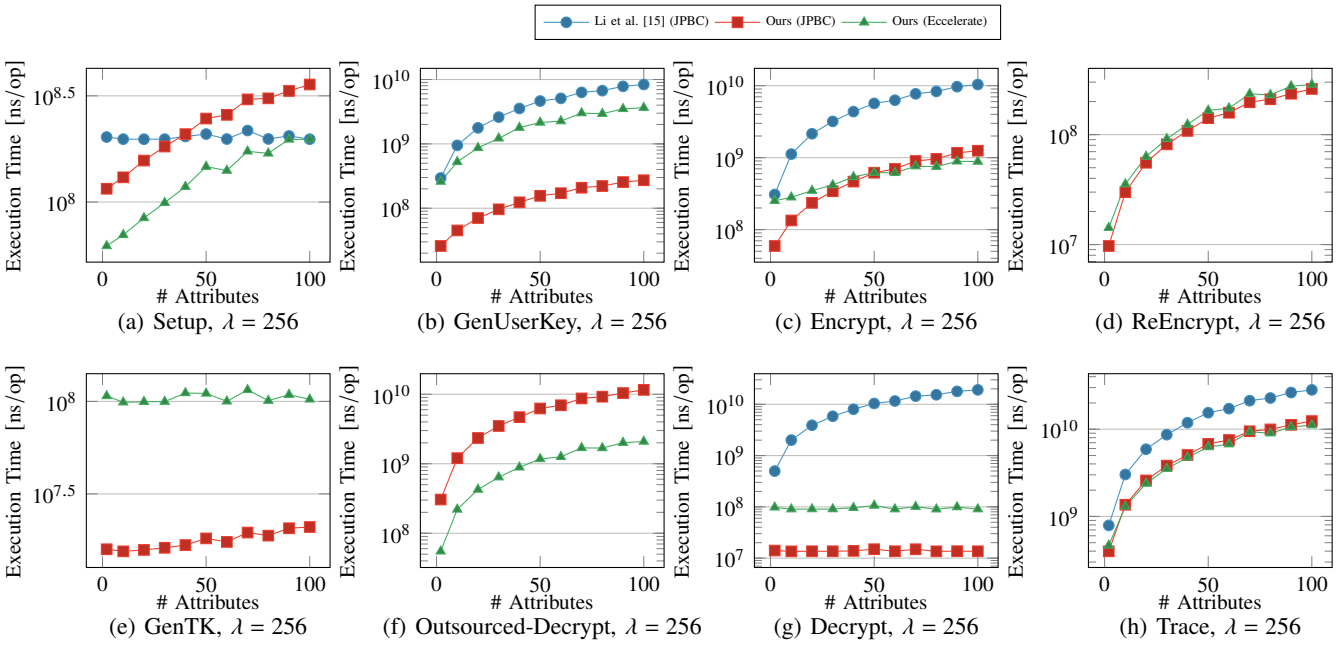

Fig. 1: Performance comparison of the proposed scheme with the schemes presented in [15]. The graphs show the timing results for different operations and different implementations. Tests were executed on an Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz.

Li et al. [15] published one of the first works to combine hidden access structures and accountability. They present a CP-ABE scheme with hidden-policies for *Internet of Things* (IoT) systems which offers white-box accountability for users and authorities. The performance of encryption and decryption scales linearly with the number of attributes. Thus, the scheme is limited to more powerful devices as expensive pairing operations are performed by the client. Arguably, a client can outsource decryption to some external proxy. This, however, entails that the proxy learns the plaintext message $\mathcal{M}$. In our construction, the decryption proxy does not learn the plaintext message or the access policy. Additionally, our construction provides revocable attributes.

Summarising, we improve state-of-the-art by combining hidden-access policies with malicious user tracing, all while reducing the computational burden on client devices.

VII. Conclusion

ABE is an emerging technology providing fine-grained access control in highly dynamic environments. On a closer look, however, the practical adoption of ABE is often limited due to open questions such as *performance*, *privacy* and *accountability*. This paper tackles these limitations and provides a novel CP-ABE approach which improves the state-of-the-art from two angles: First, our construction combines numerous features such as hidden-policies, outsourced decryption and white-box accountability. Second, the provided solution still outperforms similar constructions by an order of magnitude. Summarising, the presented approach narrows the gap between heterogeneous environments and ABE-based access control. It, thus, paves the way towards a holistic ABE solution and shows the real potential of ABE in such cloud environments. As the research has demonstrated, however, some questions still need to be answered. For example, in *Time-Sensitive Networking* (TSN) networks, even a seemingly small overhead such as the one in our construction might disrupt regular operations. A full discussion on this topic lies beyond the scope of this paper. In the future, we will therefore explore additional techniques to improve the construction. Possible techniques include, but are not limited to: outsourcing the encryption operation or removing bilinear pairings altogether.

References

[1] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-Based Encryption for Fine-grained Access Control of Encrypted Data," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, ser. CCS '06. New York, NY, USA: ACM, 2006, pp. 89–98.

[2] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-Policy Attribute-Based Encryption," in *2007 IEEE Symposium on Security and Privacy (SP '07)*, May 2007, pp. 321–334.

[3] S. Ding, C. Li, and H. Li, "A Novel Efficient Pairing-Free CP-ABE Based on Elliptic Curve Cryptography for IoT," *IEEE Access*, vol. 6, pp. 27 336–27 345, 2018.

[4] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the Decryption of ABE Ciphertexts," in *Proceedings of the 20th USENIX Conference on Security*, ser. SEC'11. Berkeley, CA, USA: USENIX Association, 2011, p. 34.

[5] S. D. Galbraith, K. G. Paterson, and N. P. Smart, "Pairings for Cryptographers," *Discrete Applied Mathematics*, vol. 156, no. 16, pp. 3113–3121, 2008.

[6] T. Okamoto and K. Takashima, "Homomorphic Encryption and Signatures from Vector Decomposition," in *Pairing-Based Cryptography – Pairing 2008*, S. D. Galbraith and K. G. Paterson, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 57–74.

[7] V. Goyal, "Reducing Trust in the PKG in Identity Based Cryptosystems," in *Advances in Cryptology - CRYPTO 2007*, A. Menezes, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 430–447.

[8] D. Boneh and H. Shacham, "Group Signatures with Verifier-Local Revocation," in *Proceedings of the 11th ACM conference on Computer and communications security - CCS '04*. New York, New York, USA: ACM Press, 2004, p. 168.

[9] A. Sahai, H. Seyalioglu, and B. Waters, "Dynamic Credentials and Ciphertext Delegation for Attribute-Based Encryption," in *Advances in Cryptology – CRYPTO 2012*, R. Safavi-Naini and R. Canetti, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 199–217.

[10] J. Chen, H. W. Lim, S. Ling, H. Wang, and H. Wee, "Shorter IBE and Signatures via Asymmetric Pairings," in *Pairing-Based Cryptography – Pairing 2012*, M. Abdalla and T. Lange, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 122–140.

[11] A. Lewko, "Tools for Simulating Features of Composite Order Bilinear Groups in the Prime Order Setting," in *Advances in Cryptology – EUROCRYPT 2012*, D. Pointcheval and T. Johansson, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 318–335.

[12] D. Boneh and X. Boyen, "Short Signatures Without Random Oracles," in *Advances in Cryptology - EUROCRYPT 2004*, C. Cachin and J. L. Camenisch, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 56–73.

[13] ——, "Short Signatures Without Random Oracles and the SDH Assumption in Bilinear Groups," *Journal of Cryptology*, vol. 21, no. 2, pp. 149–177, 2008.

[14] D. Ziegler and A. Marsalek, "Efficient Revocable Attribute-Based Encryption with Hidden Policies," in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2020, pp. 1638–1645.

[15] J. Li, Y. Zhang, J. Ning, X. Huang, G. S. Poh, and D. Wang, "Attribute Based Encryption with Privacy Protection and Accountability for CloudIoT," *IEEE Transactions on Cloud Computing*, p. 1, 2020.

[16] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, "Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption," in *Advances in Cryptology – EUROCRYPT 2010*, H. Gilbert, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 62–91.

[17] Y. Rouselakis and B. Waters, "Practical Constructions and New Proof Methods for Large Universe Attribute-Based Encryption," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, ser. CCS '13. New York, NY, USA: Association for Computing Machinery, 2013, pp. 463–474.

[18] A. De Caro and V. Iovino, "jPBC: Java pairing based cryptography," in *Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011*, Kerkyra, Corfu, Greece, June 28 - July 1, 2011, pp. 850–855.

[19] A. Guillevic, "Comparing the Pairing Efficiency over Composite-Order and Prime-Order Elliptic Curves," in *Applied Cryptography and Network Security*, M. Jacobson, M. Locasto, P. Mohassel, and R. Safavi-Naini, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 357–372.

[20] M. S. Kiraz and O. Uzunkol, "Still Wrong Use of Pairings in Cryptography," *CoRR*, vol. abs/1603.0, 2016.

[21] E. Barker, "Recommendation for Key Management Part 1: General," National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep., Jan. 2016.

[22] J. Li, Y. Wang, Y. Zhang, and J. Han, "Full Verifiability for Outsourced Decryption in Attribute Based Encryption," *IEEE Transactions on Services Computing*, vol. 13, no. 3, pp. 478–487, May 2020.

[23] J. Ning, Z. Cao, X. Dong, and L. Wei, "White-Box Traceable CP-ABE for Cloud Storage Service: How to Catch People Leaking Their Access Credentials Effectively," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 5, pp. 883–897, 2018.

[24] J. Li, K. Ren, B. Zhu, and Z. Wan, "Privacy-Aware Attribute-Based Encryption with User Accountability," in *Information Security*, P. Samarati, M. Yung, F. Martinelli, and C. A. Ardagna, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 347–362.

[25] H. Qiao, J. Ren, Z. Wang, H. Ba, and H. Zhou, "Compulsory traceable ciphertext-policy attribute-based encryption against privilege abuse in fog computing," *Future Generation Computer Systems*, vol. 88, pp. 107–116, 2018.

[26] F. Xhafa, J. Feng, Y. Zhang, X. Chen, and J. Li, "Privacy-aware attribute-based PHR sharing with user accountability in cloud computing," *The Journal of Supercomputing*, vol. 71, no. 5, pp. 1607–1619, 2015.

[27] Z. Liu, Z. Cao, and D. S. Wong, "White-Box Traceable Ciphertext-Policy Attribute-Based Encryption Supporting Any Monotone Access Structures," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 1, pp. 76–88, Jan. 2013.

[28] ——, "Blackbox Traceable CP-ABE: How to Catch People Leaking Their Keys by Selling Decryption Devices on Ebay," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, ser. CCS '13. New York, NY, USA: Association for Computing Machinery, 2013, pp. 475–486.

[29] J. Ning, Z. Cao, X. Dong, L. Wei, and X. Lin, "Large Universe Ciphertext-Policy Attribute-Based Encryption with White-Box Traceability," in *Computer Security - ESORICS 2014*, M. Kutyłowski and J. Vaidya, Eds. Cham: Springer International Publishing, 2014, pp. 55–72.

[30] J. Ning, X. Dong, Z. Cao, L. Wei, and X. Lin, "White-Box Traceable Ciphertext-Policy Attribute-Based Encryption Supporting Flexible Attributes," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 6, pp. 1274–1288, Jun. 2015.

[31] J. Ning, X. Dong, Z. Cao, and L. Wei, "Accountable Authority Ciphertext-Policy Attribute-Based Encryption with White-Box Traceability and Public Auditing in the Cloud," in *Computer Security – ESORICS 2015*, G. Pernul, P. Y A Ryan, and E. Weippl, Eds. Cham: Springer International Publishing, 2015, pp. 270–289.

[32] J. Ning, Z. Cao, X. Dong, K. Liang, L. Wei, and K. R. Choo, "CryptCloud+: Secure and Expressive Data Access Control for Cloud Storage," *IEEE Transactions on Services Computing*, p. 1, 2018.